Dew: Transparent Constant-sized zkSNARKs

Arasu Arun*[1], Chaya Ganesh[2], Satya Lokam[3], Tushar Mopuri*[2], and Sriram Sridhar†[3]

[1]New York University
arasu@nyu.edu
[2]Indian Institute of Science
{chaya,tusharmopuri}@iisc.ac.in
[3]Microsoft Research India
satya@microsoft.com, srirams@berkeley.edu

Abstract

We construct polynomial commitment schemes with constant sized evaluation proofs and logarithmic verification time in the transparent setting. To the best of our knowledge, this is the first result achieving this combination of properties.

Our starting point is a transparent inner product commitment scheme with constant-sized proofs and linear verification. We build on this to construct a polynomial commitment scheme with constant size evaluation proofs and logarithmic (in the degree of the polynomial) verification time. Our constructions make use of groups of unknown order instantiated by class groups. We prove security of our construction in the Generic Group Model (GGM). Using our polynomial commitment scheme to compile an information-theoretic proof system yields Dew – a transparent and constant-sized zkSNARK (Zero-knowledge Succinct Non-interactive ARguments of Knowledge) with logarithmic verification.

Finally, we show how to recover the result of DARK (Bünz et al., Eurocrypt 2020). DARK presented a succinct transparent polynomial commitment scheme with logarithmic proof size and verification. However, it was recently discovered to have a gap in its security proof (Block et al, CRYPTO 2021). We recover its extractability based on our polynomial commitment construction, thus obtaining a transparent polynomial commitment scheme with logarithmic proof size and verification under the same assumptions as DARK, but with a prover time that is quadratic.

*Work partially done while at Microsoft Research India.
†Work partially done while at Indian Institute of Science.

Contents

1 Introduction

Proof systems have a rich history in cryptography and theory of computation starting from interactive proofs [GMR89], zero-knowledge proofs [GMW86], and probabilistically checkable proofs [BFLS91, ALM+92]. They are now a fundamental building block in public-key encryption [NY90], signature schemes, anonymous credential systems [CL01] and various other cryptographic constructions. In recent times, proof systems are gaining widespread deployment for privacy-preserving applications on the Blockchain, for instance in cryptocurrencies like ZCash [Zca].

Succinct proofs. In real-world applications, crucial parameters of a proof system that are of interest are: the amount of interaction, proof size and efficiency of proving and verification. It is known that statistically sound proofs are unlikely to allow for significant improvements in proof size, [GH98, GVW02, Wee05], and therefore one way to beat the bound is by considering *computationally sound* proofs. Computationally sound proofs are called *arguments*. Succinct arguments, where the proof size is logarithmic in the size of the statement were first considered by Kilian [Kil92a], who gave a construction based on probabilistically checkable proofs (PCP). This was made non-interactive by Micali [Mic94a] in the random oracle model. In the plain model, non-interactivity is achieved by generating a Common Reference String (CRS) during a setup phase. There has been a series of works on constructing (zero-knowledge) Succinct Non-interactive ARguments of Knowledge (zk-SNARKs) [Gro10, Lip12, BCI+13, GGPR13, PHGR13, Lip13, BCTV14, Gro16], which have very short proofs that can be verified efficiently. These are all either in the Structured Reference String (SRS) model or in idealized models (like ROM,GGM,AGM). The constructions that achieve concretely better proof sizes are in the SRS model and require a preprocessing phase. Unfortunately, this one-time setup or preprocessing needs to be *trusted*.

Transparent proof systems. Proof systems that do not involve a trusted setup phase and the verifier randomness consists of only *public coins* are called *transparent*. The GKR protocol [GKR08], and a recent line of works building on that protocol give a proof with communication complexity $O(d \log n)$ for circuits of size n and depth d. For uniform computations, STARKs [BBHR19] achieve communication complexity of $O(\log^2 T)$ where T is a bound on the running time of the program. Another line of work attempts to reduce the degree of trust in the setup phase by constructing SNARKS with a universal and updatable SRS, where the one-time setup can be used to prove statements about any computation, as opposed to a circuit-dependent setup required in preprocessing based SNARKs. This universal SRS is, in addition, *updatable*, meaning parties can continuously contribute to the randomness of the SRS, and an SRS is trusted as long as at least one of the updates was honest. A recent line of work [CHM+20, RZ21a, CFF+21, GWC19] follows a modular approach to construct SNARKs: first, an information-theoretic component is constructed; then this is compiled into an argument using cryptographic tools. Finally, this is made non-interactive to obtain a SNARK in the random oracle model (ROM). The resulting SNARK inherits the trusted setup assumption or the transparency property from the cryptographic tools used in the compilation process.

Polynomial commitment scheme (PCS). At a high level, a PCS allows a prover to commit to a polynomial P of bounded degree so that later a verifier can ask for evaluations $P(x)$ along with proofs that the provided values are indeed consistent with the commitment. A PCS is a central cryptographic tool used to obtain a SNARK in a modular way. Any resulting SNARK from compiling an information-theoretic protocol inherits the complexity of the PCS, that is, the proof size depends on the commitment size and evaluation proof size of the PCS. Unfortunately, all existing succinct PCS schemes either require the trusted-setup assumption [KZG10], or they are transparent, but only achieve logarithmic proof size [BFS20, BHR+21][1].

1.1 Our Contributions

In this work, we present the first PCS with constant size commitment and constant evaluation proof size in the transparent setting. Our starting point is a construction of a transparent inner product commitment (IPC) scheme (which is a more general object than PCS) that allows a prover to open a committed vector to inner products with query vectors. This is succinct – that is, the size of the

[1] A flaw in the proof of security of DARK scheme [BFS20] was discovered by Block et al [BHR+21], who propose a different PCS with logarithmic proof size. See §6 for a discussion.

commitment and the proof of correct opening is independent of the size of the vector. We then specialize our IPC construction to a succinct PCS, resulting in a *transparent constant-sized PCS* with logarithmic verification – the *first* such construction to achieve the above combination of properties for a PCS, to the best of our knowledge.

From a technical point of view, our contributions are summarized below.

Inner Product Commitment (IPC) and Polynomial Commitment Scheme (PCS). We construct a constant size transparent IPC scheme in §3. In §5, we present our transparent PCS construction that achieves constant sized proofs, constant sized public parameters, and verification in $O(\log n)$ field operations and a constant number of group operations for polynomials of degree n. Both the above constructions are in the GGM.
We also show hiding and zero knowledge variants of our constructions. Using the now standard compilation process from information-theoretic proofs in idealized models to zkSNARKs via PCS [BFS20, CHM+20], we immediately obtain a *transparent constant-sized zkSNARK with constant-sized public parameters* (§7). The resulting zkSNARKs achieve $O_\kappa(1)$ communication, $O(\log n)$ "online" verification, $O(n^2)$ prover time, where n is the complexity of the NP relation (number of constraints of a Rank 1 Constraint System, or the number of gates in an arithmetic circuit). The only other transparent zkSNARKs with constant-sized proofs and public parameters are obtained by compiling constant-query PCPs using transparent vector commitment schemes with constant-sized opening proofs and public parameters. We compare our zkSNARK Dew to existing schemes in the literature in Fig.14.

A New Combinatorial Lemma. For our PCS with log-verification, we rely on some extremal combinatorial bounds. Informally, how many points can we choose in the discrete cube $[n]^d$ such that they do not contain among them the corners of a d-dimensional hyper-rectangle (box)? When $d = 2$, this bound also follows from the Zarankiewicz problem [Bol04] in extremal graph theory, and for dimension d, it is given by a Box theorem due to Rosenfeld [Ros16]. We can use these bounds to obtain sublinear (l^ϵ) verification time for constant $0 < \epsilon < 1$. But they do not suffice to obtain logarithmic verification. We achieve logarithmic verification by generalizing boxes to "d-structures" (details in §5.2.1) and obtaining significant asymptotic improvements over the Box theorem [Ros16] and tight bounds on the number of points that do not contain a d-structure. Extremal combinatorics results like this have found applications in complexity theory and theoretical computer science in general. To the best of our knowledge, our result is the first application of an extremal combinatorics theorem in the construction of SNARKs and we believe this to be of independent interest.

Recovering the DARK [BFS20] result. We show that our PCS can be adapted to obtain logarithmic proof size and verification by employing the recursive evaluation protocol from DARK on our new commitment scheme. This recovers the flawed Lemmas 8, 9 from DARK thus recovering a transparent PCS with logarithmic proof size and logarithmic verification, *but* at the expense of an increased quadratic prover time. We note that the DARK recovery does not require GGM; we achieve this result under the same assumptions made in DARK, i.e., the Adaptive Root Assumption and the Strong RSA Assumption. We also note that [BHR+21] also give a construction that achieves similar results as DARK by modifying DARK's evaluation protocol. In contrast, our construction is a commitment scheme that is syntactically close to DARK, and has a similar evaluation protocol. We present this in §6.

1.2 Related Work

Functional commitments were introduced by [LRY16] as a generalization of a vector commitment scheme, where, a prover can commit to a vector, and later respond to inner product queries by revealing the inner product together with a proof that the answer is consistent with the committed vector. In their definition, they only ask for function binding, without any extractability guarantee. Lai and Malavolta [LM19] put forth the notion of Linear Map Commitments (LMC) that capture a more general class of functions on a committed vector allowing a prover to open a commitment to the output of a linear map. The construction from [LM19] achieves succinctness — constant commitment and proof size, but requires trusted setup and is therefore, not transparent.

In a recent concurrent work, [CFKS22] presents transparent inner product commitment schemes with constant size openings and constant size public parameters. Their scheme is also in groups of unknown

order, however, the techniques they use are completely different. Their result relies on proofs of cardinality of RSA accumulated sets, whereas we rely on integer encoding of vectors and combinatorial techniques to show extraction.

Polynomial commitment schemes were introduced in [KZG10], and have since led to several variants being used in recent SNARKs. The KZG scheme [KZG10] gives constant-sized commitments and proofs, but require a trusted setup. In the transparent setting, Wahby et al. [WTs+18] constructed a polynomial commitment scheme for multilinear polynomials that has commitment size and evaluation proof size $O(\sqrt{d})$ for degree d polynomials. Zhang et al. [ZXZS20] construct a polynomial commitment from FRI (Fast Reed Solomon IOPP) that is transparent, has constant size commitments, but evaluation proofs have size $O(\log^2 d)$. Transparent instantiations of the schemes in Spartan [Set20] yield SNARKs with polylogarithmic proof size and verification.

As mentioned earlier, Bünz et al [BFS20] used a Diophantine Argument of Knowledge (DARK), and constructed a polynomial commitment scheme for m-variate polynomials of total degree d with proof size $O(m \log d)$ and $O(m \log d)$ verification time. However, Block et al [BHR+21] identified a gap in the proof of security of the DARK scheme that breaks the extraction procedure. The work of [BHR+21] also propose a modification to the DARK scheme that sidesteps the gap in extraction, with polylogarithmic proof size and verification time.

Classical CS proofs approach [Kil92b, Mic94b] based on PCP also yields a transparent SNARK. The resulting properties are inherited from the vector commitment scheme (VC) used to commit to the PCP. Using Merkle trees as VC yield proofs of logarithmic size, and VC constructions with constant size openings (like the constructions in [LM19, BBF19]) give constant-sized SNARKs and logarithmic verification.

1.3 Technical Overview

We now give an informal description of the technical ideas behind our constructions.

The intuitive starting point of our commitment schemes is the following basic ideas of converting vectors into integers. Given a vector[2] $\mathbf{c}$, define $\mathsf{int}_\alpha(\mathbf{c}) := \langle \mathbf{c}, \boldsymbol{\alpha} \rangle := \sum_0^{l-1} c_i \alpha^i$, where $\boldsymbol{\alpha} := (1, \alpha, \alpha^2, \ldots, \alpha^{l-1})$, for $\alpha \gg \max_i c_i$ (bounds on α are defined in terms of p and l in later sections). Given a random element $g \in \mathbb{G}$ from a group $\mathbb{G}$ of unknown order, our commitment to vector $\mathbf{c}$ is then defined as $C := g^{\mathsf{int}_\alpha(\mathbf{c})}$. Conversely, given $C \in \mathbb{G}$, and parameters g and α, using a proof of knowledge of exponent protocol (see PoKPE that builds on Wesolowski's PoE protocol [Wes19] and GGM in Section 2.5), we can compute a such that $C = g^a$ and derive a vector $\mathbf{a}$ from the α-base representation of $a = a_0 + a_1\alpha + a_2\alpha^2 \ldots$.

Now, given a query vector $\mathbf{q}$, to evaluate the inner product $\langle \mathbf{c}, \mathbf{q} \rangle$, we consider the integer product

$$\mathsf{int}_\alpha(\mathbf{c}) \cdot \mathsf{int}_\alpha(\mathsf{reverse}(\mathbf{q})) = (\sum_{i=0}^{l-1} c_i \alpha^i) \cdot (\sum_{i=0}^{l-1} q_{l-i} \alpha^i) = L + \alpha^l \cdot \langle \mathbf{c}, \mathbf{q} \rangle + H, \tag{1}$$

where L and H are polynomials in α of degree less than l and more than l respectively. Raising g to both sides of (1), we obtain

$$C^{\mathsf{int}_\alpha(\mathsf{reverse}(\mathbf{q}))} = (g^L) \cdot (g^{\alpha^l \langle \mathbf{c}, \mathbf{q} \rangle}) \cdot (g^H). \tag{2}$$

It follows that if a prover sends to a verifier a commitment C to $\mathbf{c}$, and given a query vector $\mathbf{q}$, claims that inner product $\langle \mathbf{c}, \mathbf{q} \rangle$ evaluates to v, and *also* sends g^L and g^H, then the verifier can check consistency of the prover's claim using (2) (with v in place of $\langle \mathbf{c}, \mathbf{q} \rangle$). While this intuition suffices for a completeness proof, it is by no means sufficient for a soundness proof. Our main challenge is to show that a check somewhat analogous to (2) with some additional machinery *suffices* for a verifier to catch a cheating prover with high probability.

The above argument is reminiscent of approaches in [LM19, BFS20]; however, we have to tackle a number of technical challenges that arise since our goal is *constant sized* proofs (unlike in [BFS20]), and in the

[2] Our vectors are in $\mathbb{Z}_p^l$, which we often view as vectors with integer coordinates in $\{0, \ldots, p-1\}$.

transparent setting where α is not secret (unlike the trusted setup in [LM19]). We describe below the hurdles and our ideas to overcome them at a high level.

A cheating prover could use large values (violating $\alpha \gg \max_i c_i$) in the committed vector (extracted by computing the α-base representation of exponent of g in C) that could cause "overflow" in the coordinates of the computation expressed in (1). In that case, we can no longer guarantee the correctness of the inner product.

To control the issues caused by overflow, our idea is to double the length of the vector to $2l$ and place the vector $\mathbf{c}$ to be committed in even positions $(0, 2, \ldots, 2l-2)$ and 0's in odd positions (in the honest case). More generally, let $\mathbf{d}$ (resp. $\mathbf{c}$) denote the subvector in the odd (even) positions and let $\mathbf{c}\|\mathbf{d}$ denote the combined vector of length $2l$. Then, in the honest case, $\mathbf{d} = \mathbf{0}$ and $\mathsf{int}_\alpha(\mathbf{c}\|\mathbf{d}) = \mathsf{int}_{\alpha^2}(\mathbf{c})$ and hence completeness goes through as before (by replacing α with α^2). From now on, we will consider equations (1) and (2) generalized to length $2l$ vectors and commitment C to the vector $\mathbf{c}\|\mathbf{d}$. Suppose now, a cheating prover creates non-zero entries in the $\mathbf{d}$-coordinates (but uses a $\mathbf{c} \in \mathbb{Z}_p^l$). Then, the inner product of $\mathbf{c}\|\mathbf{d}$ with a uniformly chosen random vector $\mathbf{z} \in \mathbb{Z}_p^l$ in *odd* positions ($\mathbf{d}$-coordinates) and 0's in even positions will be nonzero with high probability by Schwartz-Zippel. Define *test equations* to be the (generalizations of) (1) and (2) but without the middle terms on the right hand side (corresponding to inner product being 0) and $0^l\|\mathbf{z}$ for a random $\mathbf{z} \in \mathbb{Z}_p^l$ as the (test) query vector[3]. We use these test equations to define a **TEST** protocol (cf. Fig. 2) that allows a verifier to immediately catch a cheating prover or derive structural conditions on the $\mathbf{d}$-vector that are exploited in a subsequent inner product evaluation protocol **IPP** (cf. Fig 3). By the Schwartz-Zippel argument above, if a cheating prover succeeds in **TEST**, the middle term is no longer the inner product $\langle \mathbf{d}, \mathbf{z} \rangle$, but is corrupted by an error term overflowing from the $\mathbf{c}$-vector: it is in fact, $\langle \mathbf{d}, \mathbf{z} \rangle \mod \alpha + \mathsf{err}$, where $\mathsf{err} = \lfloor \frac{\langle \mathbf{c}, \mathbf{z} \rangle}{\alpha} \rfloor$. Vanishing of this middle term combined with bounds on $\mathbf{c}$, $\mathbf{z}$ in relation to α and hardness assumptions in GGM, allows us to show that $\mathbf{d}$ must have a certain structure, captured by Theorem 3.4; namely, the coordinates of $\frac{\mathbf{d}}{\alpha}$ are close to rationals with small ($\ll \alpha$) denominators. This structure theorem is a crucial technical ingredient of our results.

Armed with the structure theorem, we prove (Theorem 3.5) that if an instance of the inner product evaluation protocol **IPP** succeeds in satisfying (generalizations) of (1) and (2), with query vector $\mathbf{q}\|0^l$ (i.e. real query vector in *even* positions and 0's in odd positions), then we can extract a vector $\tilde{\mathbf{c}}$ that, while fractional over the integers, has invertible denominators modulo p. Using this $\tilde{\mathbf{c}}$ as the "opening hint" (cf. **Open()** in §3.1), we can then extract a unique $\mathbf{c}$. This completes the technical overview for our IPC scheme.

PCS with logarithmic verification. Our IPC scheme above immediately yields a Polynomial Commitment Scheme (PCS), noting that, for a polynomial f given by its vector of coefficients $\mathbf{f} = (f_0, \ldots, f_{l-1})$, $f(x) = \langle \mathbf{f}, \mathbf{x} \rangle$, where $\mathbf{x} = (1, x, \ldots, x^{l-1})$. However, the verification complexity of the resulting PCS is much worse than what we want to achieve. While linear verification seems inherent for inner products (since the query vectors can be arbitrary and the verifier needs to at least read the statement), in a PCS, the query vector $\mathbf{x}$ parameterized by single variable x, we can hope to achieve logarithmic verification time. In particular, we can delegate the verifier's expensive exponentiations to the prover using standard Proof of Exponentiation (PoE). This makes the verifier in **IPP** protocol (specialized to a PCS) logarithmic. However, the bottleneck for verifier computation arises from the **TEST** protocol of our IPS. Note that while the query vector $\mathbf{x}$ is parameterized by a single variable x, the *test vector* z is not and hence computing $\mathsf{int}_\alpha(\mathsf{reverse}(\mathbf{z}))$ in checking (2) seems to require linear time.

To reduce verifier's computation, we use the idea of *Kronecker products*: instead of choosing $\mathbf{z} \in \mathbb{Z}_p^l$, we choose $\log l$ vectors $\mathbf{z_1}, \ldots, \mathbf{z}_{\log \mathbf{l}}$ from $\mathbb{Z}_p^2$ and define $\mathbf{z} = \mathbf{z_1} \otimes \cdots \otimes \mathbf{z}_{\log \mathbf{l}}$. To illustrate how this helps, consider the following computation needed on the right hand side of (2), where $\mathbf{z}$ is as above and $i = (i_0, \ldots, i_{\log l - 1})$ the binary expansion of index $i \in [l]$.

$$\mathsf{int}_\alpha(\mathsf{reverse}(\mathbf{z})) = \sum_{i=0}^{l-1} \alpha^{l-i} \prod_{j=0}^{\log l - 1} z_{j,i_j} = \alpha^l \cdot \prod_{j=0}^{\log l - 1} (z_{j,0} + z_{j,1}\alpha^{-2^j}),$$

[3] To make the communication complexity in this step to be constant, verifier can send the seed to a PRG that generates $\mathbf{z}$.

and note that the big product on the right hand side can be computed in logarithmic time.

While Kronecker helps improve verifier efficiency of **TEST**, it breaks soundness! Note that the extractability proof of **TEST** in IPC relies on uniform randomness of the test vector $\mathbf{z} \in \mathbb{Z}_p^l$. So, to benefit from Kronecker products, we must *improve the extractability proof to work with exponentially smaller randomness* in the $\log l$ vectors $\mathbf{z}_j$ of length 2 as compared to the single length-l vector z from before. Specifically, it is crucial to recover analog of the structure theorem from this vastly reduced space of randomness. We do so by proving a *new result in extremal combinatorics*. Informally, this theorem (Theorem 5.2) gives a tight upper bound on the number of points in the hypercube $[n]^d$ such that no subset of 2^d points in that set form a configuration called a *d-structure*. A d-structure is a generalization of the corners of a *box* or a hyper-rectangle. When $d = 2$ and the "forbidden" configuration is a rectangle, this is the well-known Zarankiewicz problem [Bol04] from extremal graph theory and the bound is asymptotically $n^{3/2}$. We need to generalize this in two ways: first, we need to consider high dimensions with growing d (but no more than $\log n$) and second, we need to generalize rectangles/boxes to d-structures; when $d = 2$, a d-structure generalizes a rectangle to a parallelogram and for larger d, a recursive definition is given in §5.2.2. For $d > 2$, Rosenfeld [Ros16] proved an upper bound of $n^{d-2^{-d+1}}$ on the number of points that do not contain the corners of a box. This bound, however, is insufficient for us to get logarithmic verification since as d grows it tends to n^d almost entirely filling the space. By generalizing boxes to d-structures, we succeed in obtaining a tight upper bound of $(n^d - (n-1)^d) \leq dn^{d-1}$, which is a vanishingly small fraction of n^d.

How do we use this combinatorial bound? Intuitively, each successful run of **TEST** by the prover corresponds to choosing a point in $[n]^d$ (in our application, this will be space of randomness, n will be p and d will be like $\log l$). Hence, a prover that succeeds with a non-negligible probability gives rise to many such points and then the combinatorial bound implies the existence of a d-structure in that space. Each point in that structure gives rise to equations involving the coordinates of the $\mathbf{d}$-vector (from $\mathbf{c}\|\mathbf{d}$) and random variable z_{ij}. These equations are derived as consequences of satisfaction of (2) (corresponding to prover's success). The recursive nature of the structure can then be exploited to combine (fold) these equations by recursively subtracting them along various dimensions and finally obtain conditions only on the coordinates of the $\mathbf{d}$-vector. These conditions help us recover an analog of the structure theorem (Theorem 3.4) for **logTEST** (Theorem 5.4).

The cost for recovering the structure theorem for **logTEST** in the reduced randomness space is increased value of α. Whereas we could do with an α that is polynomial in p in the IPC, for the PCS with log verification as described above, we require an α that is as big as $p^{l \log l}$. To reduce the value of α, we use a very recent result by Bünz and Fisch [BF22] who prove a Schwartz-Zippel lemma for multilinear polynomials over mod n for *composite* n. We use this lemma to prove an upper bound on the number of d-structures that can arise from a prover's successful execution of **logTEST**. On the other hand, our combinatorial theorem not only shows the *existence* but also gives a lower bound on the *number* of d-structures that must exist when we choose many points in the randomness space. Combing these helps us achieve extractability with a much smaller value of α that is only $p^{O(\log l)}$.

Recovering DARK. In DARK, the evaluation protocol is recursive, where at each step, the parties recurse on a random linear combination of an instance of reduced size. The magnitude of the digits in the base-q representation (q is the large integer at which the polynomial is evaluated) grows as the protocol proceeds. In the DARK extraction procedure, the extractor takes a tree of accepting transcripts, recursively runs an extractor twice at each level, and combines two outputs at level $i+1$ into an output at level i. Block et al [BHR+21] note that this recombination is not sound. In particular, the proof of the Lemma that argues that the digits remain small (Lemma 8 in DARK) after recombination is flawed. Thus, there is no valid extractor that shows that DARK PCS satisfies witness extended emulation.

We show a PCS with logarithmic proof size and verification. The way we commit to a polynomial is syntactically similar to the DARK PCS: encoding the evaluation of the polynomial at a large integer. We construct a PCS that is similar to the one defined above in the commit function, but the evaluation protocol is recursive as in DARK. Since our PCS allows rationals as opening hints, by setting our parameter α carefully, we recover the flawed lemma and prove extraction based on the adaptive root and strong RSA assumptions. This prover, however, takes quadratic time. This is due to an increased value of α that is an artifact of our encoding methods.

2 Preliminaries

Notation. A finite field is denoted by $\mathbb{F}$. We denote by κ a security parameter. When we explicitly specify the random tape for a randomized algorithm A, then we write $a \leftarrow A(\mathsf{pp}; \rho)$ to indicate that A outputs a given input pp and random tape ρ. We consider interactive arguments for relations, where a prover P convinces the verifier that it knows a witness w such that for a public statement $x, (x, w) \in \mathcal{R}$. For a pair of PPT interactive algorithms P, V, we denote by $\langle P(w), V\rangle(x)$, the output of V on its interaction with P where w is P's private input and x is a common input.

Fiat-Shamir transform. In this work, we consider *public coin* interactive arguments where the verifier's messages are uniformly random strings. Public coin protocols can heuristically be made non-interactive by applying the Fiat-Shamir [FS87] transform (FS) in the Random Oracle Model (ROM).

2.1 Inner Product Commitments

We define Inner Product Commitments (IPC) which is an extension of functional commitments introduced in [LRY16]. IPC allows a prover to prove that the committed vector $\mathbf{f}$ satisfies $\langle \mathbf{f}, \mathbf{q}\rangle = v$, for some vector $\mathbf{q}$ and v.

An Inner Product Commitment scheme over $\mathbb{F}$ is a tuple
$\mathsf{IPC} = (\mathbf{Setup}, \mathbf{Com}, \mathbf{Open}, \mathbf{Eval})$ where:

- $\mathbf{Setup}(1^\kappa, D) \to \mathsf{pp}$. On input security parameter κ, and an upper bound on the lengths of vectors accepted as inputs $D \in \mathbb{N}$, **Setup** generates public parameters pp.
- $\mathbf{Com}(\mathsf{pp}, f_0, \ldots, f_{l-1}, l) \to (C, \tilde{\mathbf{c}})$. On input the public parameters pp, the length of the vector $l \leq D$ and a vector of length l, given as $f_0, \ldots, f_{l-1} \in \mathbb{F}$, **Com** outputs a commitment C, and additionally an opening hint $\tilde{\mathbf{c}} \equiv (f_0 \ldots, f_{l-1})$.
- $\mathbf{Open}(\mathsf{pp}, \mathbf{f}, l, C, \tilde{\mathbf{c}}) \to b$. On input the public parameters pp, the opening hint $\tilde{\mathbf{c}}$, the length of the vector in the commitment l and the commitment C, the claimed committed vector $\mathbf{f}$, **Open** outputs a bit indicating accept or reject.
- $\mathbf{Eval}(\mathsf{pp}, C, l, \mathbf{q}, v; \mathbf{f}) \to b$. A public coin interactive protocol $\langle P_{\mathbf{Eval}}(\mathbf{f}), V_{\mathbf{Eval}}\rangle(\mathsf{pp}, C, l, \mathbf{q}, v)$ between a PPT prover and a PPT verifier. The parties have as common input public parameters pp, commitment C, the length of the vector in the commitment l, query vector $\mathbf{q} \in \mathbb{F}^l$, and claimed inner product v. The prover has, in addition, the vector committed to in C, $\mathbf{f}$. At the end of the protocol, the verifier outputs 1 indicating accepting the proof that $\langle \mathbf{f}, \mathbf{q}\rangle = v$, or outputs 0 indicating rejecting the proof.

Definition 2.1 (Completeness)**.** *For all $l \leq D$, for all inputs $f_0, \ldots, f_{l-1} \in \mathbb{F}$, for query vectors $\mathbf{q} \in \mathbb{F}^l$,*

$$\Pr\left(b = 1 \;:\; \begin{array}{c} \mathsf{pp} \leftarrow \mathbf{Setup}(1^\kappa, D) \\ (C, \tilde{\mathbf{c}}) \leftarrow \mathbf{Com}(\mathsf{pp}, f_0, \ldots, f_{l-1}, l) \\ v \leftarrow \langle (f_0, \ldots, f_{l-1}), \mathbf{q}\rangle \\ b \leftarrow \mathbf{Eval}(\mathsf{pp}, C, l, \mathbf{q}, v; \mathbf{f}) \end{array}\right) = 1.$$

Definition 2.2 (Binding)**.** *An Inner Product Commitment scheme PC is binding if for all PPT $\mathcal{A}$, the following probability is negligible in κ.*

$$\Pr\left(\begin{array}{c} \mathbf{Open}(\mathsf{pp}, \mathbf{f_0}, l, C, \tilde{\mathbf{c}}_\mathbf{0}) = 1 \wedge \\ \mathbf{Open}(\mathsf{pp}, \mathbf{f_1}, l, C, \tilde{\mathbf{c}}_\mathbf{1}) = 1 \wedge \\ \tilde{\mathbf{c}}_\mathbf{0} \neq \tilde{\mathbf{c}}_\mathbf{1} \end{array} \;:\; \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{setup}(1^\kappa, D) \\ (C, \mathbf{f_0}, \mathbf{f_1}, \tilde{\mathbf{c}}_\mathbf{0}, \tilde{\mathbf{c}}_\mathbf{1}, l) \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right).$$

Definition 2.3 (Succinctness)**.** *We require the commitments and the evaluation proofs to be of size independent of the length of the vector, that is the scheme is* proof succinct *if $|C|$ is $\mathsf{poly}(\kappa)$ and $|\pi|$ is $\mathsf{poly}(\kappa)$, where π is the transcript obtained by applying FS to* **Eval**.

Definition 2.4 (Extractability). *For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT algorithm* Ext *such that the following probability is negligible in κ:*

$$\Pr\left[b = 1 \wedge \mathcal{R}_{\mathbf{Eval}}(\mathsf{pp}, C, l, \mathbf{q}, v; \mathbf{f}, \tilde{\mathbf{c}}) = 0 \ : \begin{array}{c} \mathsf{pp} \leftarrow \mathbf{Setup}(1^\kappa, D) \\ (C, l, \mathbf{q}, v, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ (\mathbf{f}, \tilde{\mathbf{c}}) = \mathsf{Ext}^{\mathcal{A}_2}(\mathsf{pp}) \\ b \leftarrow \langle \mathcal{A}_2(\mathsf{st}), V_{\mathbf{Eval}} \rangle(\mathsf{pp}, C, l, \mathbf{q}, v) \end{array} \right].$$

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

$$\mathcal{R}_{\mathbf{Eval}} = \{ \big((\mathsf{pp}, C \in \mathbb{G},\ l \in \mathbb{N},\ \mathbf{q} \in \mathbb{F}^l,\ v \in \mathbb{F});\ (\mathbf{f}, \tilde{\mathbf{c}}) \big) : \\ (\mathbf{Open}(\mathsf{pp}, \mathbf{f}, l, C, \tilde{\mathbf{c}}) = 1) \wedge v = \langle \mathbf{f}, \mathbf{q} \rangle \mod p \}$$

2.2 Polynomial Commitment Scheme

The notion of a polynomial commitment scheme that allows the prover to open evaluations of the committed polynomial succinctly was introduced in [KZG10] who gave a construction under the trusted setup assumption. A polynomial commitment scheme over $\mathbb{F}$ is a tuple $\mathsf{PC} = (\mathsf{setup}, \mathsf{commit}, \mathsf{open}, \mathsf{eval})$ where:

- $\mathsf{setup}(1^\kappa, D) \to \mathsf{pp}$. On input security parameter κ, and an upper bound $D \in \mathbb{N}$ on the degree, setup generates public parameters pp.
- $\mathsf{commit}(\mathsf{pp}, f(X), d) \to (C, \tilde{\mathbf{c}})$. On input the public parameters pp, and a univariate polynomial $f(X) \in \mathbb{F}[X]$ with degree at most $d \leq D$, commit outputs a commitment to the polynomial C, and additionally an opening hint $\tilde{\mathbf{c}}$.
- $\mathsf{open}(\mathsf{pp}, f(X), d, C, \tilde{\mathbf{c}}) \to b$. On input the public parameters pp, the commitment C and the opening hint $\tilde{\mathbf{c}}$, a polynomial $f(X)$ of degree $d \leq D$, open outputs a bit indicating accept or reject.
- $\mathsf{eval}(\mathsf{pp}, C, d, x, v; f(X)) \to b$. A public coin interactive protocol $\langle P_{\mathsf{eval}}(f(X)), V_{\mathsf{eval}} \rangle(\mathsf{pp}, C, d, z, v)$ between a PPT prover and a PPT verifier. The parties have as common input public parameters pp, commitment C, degree d, evaluation point x, and claimed evaluation v. The prover has, in addition, the opening $f(X)$ of C, with $\deg(f) \leq d$. At the end of the protocol, the verifier outputs 1 indicating accepting the proof that $f(x) = v$, or outputs 0 indicating rejecting the proof.

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

Definition 2.5 (Completeness). *For all polynomials $f(X) \in \mathbb{F}[X]$ of degree $d \leq D$, for all $x \in \mathbb{F}$,*

$$\Pr\left[b = 1 \ : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{setup}(1^\kappa, D) \\ (C, \tilde{\mathbf{c}}) \leftarrow \mathsf{commit}(\mathsf{pp}, f(X), d) \\ v \leftarrow f(x) \\ b \leftarrow \mathsf{eval}(\mathsf{pp}, C, d, x, v; f(X)) \end{array} \right] = 1.$$

Definition 2.6 (Binding). *A polynomial commitment scheme* PC *is binding if for all PPT $\mathcal{A}$, the following probability is negligible in κ:*

$$\Pr\left[\begin{array}{c} \mathsf{open}(\mathsf{pp}, f_0, d, C, \tilde{\mathbf{c}}_{\mathbf{0}}) = 1 \wedge \\ \mathsf{open}(\mathsf{pp}, f_1, d, C, \tilde{\mathbf{c}}_{\mathbf{1}}) = 1 \wedge \\ f_0 \neq f_1 \end{array} : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{setup}(1^\kappa, D) \\ (C, f_0, f_1, \tilde{\mathbf{c}}_{\mathbf{0}}, \tilde{\mathbf{c}}_{\mathbf{1}}, d) \leftarrow \mathcal{A}(\mathsf{pp}) \end{array} \right].$$

Definition 2.7 (Extractability). *For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT algorithm* Ext *such that the following probability is negligible in κ:*

$$\Pr\left[b = 1 \wedge \mathcal{R}_{\mathsf{eval}}(\mathsf{pp}, C, x, v; \tilde{f}, \tilde{\mathbf{c}}) = 0 \ : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{setup}(1^\kappa, D) \\ (C, d, x, v, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ (\tilde{f}, \tilde{\mathbf{c}}) \leftarrow \mathsf{Ext}^{\mathcal{A}_2}(\mathsf{pp}) \\ b \leftarrow \langle \mathcal{A}_2(\mathsf{st}), V_{\mathsf{eval}} \rangle(\mathsf{pp}, C, d, x, v) \end{array} \right].$$

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

$$\mathcal{R}_{\mathsf{eval}} = \{((\mathsf{pp}, C \in \mathbb{G},\ x \in \mathbb{F},\ v \in \mathbb{F});\ (f(X), \tilde{\mathbf{c}})) : \\ (\mathsf{open}(\mathsf{pp}, f, d, C, \tilde{\mathbf{c}}) = 1) \wedge v = f(x)\}$$

Definition 2.8 (Succinctness)**.** *We require the commitments and the evaluation proofs to be of size independent of the degree of the polynomial, that is the scheme is* proof succinct *if* $|C|$ *is* $\mathsf{poly}(\kappa)$*,* $|\pi|$ *is* $\mathsf{poly}(\kappa)$ *where* π *is the transcript obtained by applying FS to* eval*. Additionally, the scheme is* verifier succinct *if* eval *runs in time* $\mathsf{poly}(\kappa) \cdot \mathsf{log}(d)$ *for the verifier.*

2.3 Assumptions

Groups of unknown order and GGM. Our constructions make use of groups of unknown order. A class group is a candidate group of unknown order. The *class group* of an imaginary quadratic order [BS96, BH01] is the quotient group of fractional ideals by principal ideals of an order of a number field with ideal multiplication. It is completely defined by its discriminant, which can be generated using only public randomness.

We use the generic group model (GGM) for groups of unknown order as defined by Damgård and Koprowski [DK02], and used in [BBF19]. In this model, the group is parameterized by two integer public parameters A, B and the order of the group is sampled uniformly from $[A, B]$. The group $\mathbb{G}$ description consists of a random injective function $\sigma : \mathbb{Z}_{|\mathbb{G}|} \to \{0,1\}^{\ell}$, for some ℓ where $2^{\ell} \gg |\mathbb{G}|$. The elements of the group are $\sigma(0), \sigma(1), \ldots, \sigma(|\mathbb{G}| - 1)$.

A generic group algorithm $\mathcal{A}$ is a probabilistic algorithm with the following properties. Let $\mathcal{L}$ be a list that is initialized with the encodings (group elements) given to $\mathcal{A}$ as inputs. $\mathcal{A}$ can query two generic group oracles, $\mathcal{O}_1$ and $\mathcal{O}_2$. $\mathcal{O}_1$ samples a random $r \in \mathbb{Z}_{|\mathbb{G}|}$ and returns $\sigma(r)$ which is appended to $\mathcal{L}$. The second oracle $\mathcal{O}_2(i, j, \pm)$ takes two indices $i, j \in \{1, \ldots, q\}$, where q is the size of $\mathcal{L}$, and a sign bit and returns $\sigma(x_i \pm x_j)$, which is appended to $\mathcal{L}$. It should be noted that $\mathcal{A}$ is not given $|\mathbb{G}|$. We use a group sampler GGen that on input a security parameter κ, samples a description of the group $\mathbb{G}$ of size $2^{\mathsf{poly}(\kappa)}$. Note that GGen is public-coin.

Assumption 1 (Order assumption – [BBF19])**.** *For a group of unknown order* $\mathbb{G} \leftarrow \mathsf{GGen}(\kappa)$*, the* Order assumption *holds if for any adversary* $\mathcal{A}$

$$\Pr\left[\mathsf{w} \neq 1 \wedge \mathsf{w}^{\alpha} = 1 \ : \begin{array}{c} \mathsf{GGen}(\kappa) \to \mathbb{G} \\ \mathcal{A}(\mathbb{G}) \to (\mathsf{w}, \alpha) \\ \text{where } |\alpha| < 2^{poly(\kappa)} \in \mathbb{Z} \text{ and } \mathsf{w} \in \mathbb{G} \end{array}\right] \leq negl(\kappa)$$

Assumption 2 (Adaptive root assumption – [Wes19])**.** *For a group of unknown order* $\mathbb{G} \leftarrow \mathsf{GGen}$*, we say that the* Adaptive root assumption *holds for* GGen *if there is no efficient adversary* $(\mathcal{A}_0, \mathcal{A}_1)$ *that succeeds in the following task:*

- $\mathcal{A}_0$ *outputs an element* $w \in \mathbb{G}$ *and some* state*.*
- $\mathcal{A}_1$ *is given a uniformly sampled* $\ell \leftarrow\!\$\ \boldsymbol{Primes}(\kappa)$ *and the output* state *of* $\mathcal{A}_0$*.*
- $\mathcal{A}_1$ *outputs* $z = w^{1/\ell} \in \mathbb{G}$*.*

More precisely, there exists a negligible function $negl(\kappa)$ *such that:*

$$\Pr\left[z^{\ell} = w \neq 1 \ : \begin{array}{c} \mathsf{GGen}(\kappa) \to \mathbb{G} \\ \mathcal{A}_0(\mathbb{G}) \to (w, \mathsf{state}) \\ \ell \leftarrow\!\$\ \boldsymbol{Primes}(\kappa) \\ \mathcal{A}_1(\ell, \mathsf{state}) \to z \end{array}\right] \leq negl(\kappa)$$

Assumption 3 (r-Fractional Root Assumption – [BFS20])**.** *For a group of unknown order, the* r-Fractional Root Assumption *holds for* GGen *if for any efficient adversary* $\mathcal{A}$*:*

$$\Pr\left[u^\beta = g^\alpha \wedge \frac{\beta}{\gcd(\alpha,\beta)} \neq r^k, k \in \mathbb{N}. \; : \; \begin{array}{c} \mathsf{GGen}(\kappa) \to \mathbb{G} \\ \mathbb{G} \to g \\ \mathcal{A}(\mathbb{G}, g) \to (\alpha, \beta, u) \\ \textit{where } |\alpha| < 2^{poly(\kappa)}, \\ |\beta| < 2^{poly(\kappa)} \in \mathbb{Z}, \\ \textit{and } u \in \mathbb{G} \end{array}\right] \leq negl(\kappa)$$

Note that the adaptive root assumption implies the order assumption. In addition, these assumptions are indeed intractable in the Generic Group Model as shown in [BBF19].

2.4 Arguments of knowledge and SNARKs

Definition 2.9 (Indexed relation [CHM+20])**.** *An* indexed relation $\mathcal{R}$ *is a set of triples* $(\mathsf{i}, \mathsf{x}, \mathsf{w})$ *where* i *is the index,* x *is the instance, and* w *is the witness. The corresponding* indexed language $\mathcal{L}_\mathcal{R}$ *is the set of pairs* (i, x) *for which there exists a witness* w *such that* $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$*. Given a size bound* $\mathsf{n} \in \mathbb{N}$*, we denote by* $\mathcal{R}_\mathsf{n}$ *the restriction of* $\mathcal{R}$ *to triples* $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$ *such that* $|\mathsf{i}| \leq \mathsf{n}$*.*

A proof (or argument) for a language $\mathcal{L}$ allows a prover P to convince a verifier V that $x \in L$ for a common input x. A proof of knowledge[4] intuitively captures not only the truth of a statement $x \in L$, but also the fact that the prover is in "possession" of a witness w.

Definition 2.10 (Preprocessing Argument of Knowledge with Universal SRS)**.** *A preprocessing argument of knowledge with universal SRS is a tuple of four algorithms* $\mathsf{AoK} = (\mathsf{S}, \mathsf{I}, \mathsf{P}, \mathsf{V})$*.* S *is a probabilistic polynomial-time setup algorithm that given a bound* $\mathsf{n} \in \mathbb{N}$ *samples a structured reference string* srs *supporting indices of size up to* n*. The indexer algorithm* I *is deterministic and, given oracle access to* srs *produces a proving index key and a verifier index key, used by* P *and* V *respectively.* P *and* V *are probabilistic polynomial-time interactive algorithms.*

1. *Completeness. For all size bounds* $\mathsf{n} \in \mathbb{N}$ *and PPT* $\mathcal{A}$*, the following probability is* 1*.*

$$\Pr\left(\begin{array}{c} (\mathsf{i}, \mathsf{x}, \mathsf{w}) \notin \mathcal{R}_\mathsf{n} \vee \\ \langle \mathsf{P}(\mathsf{ipk}, \mathsf{x}, \mathsf{w}), \mathsf{V}(\mathsf{ivk}, \mathsf{x}) \rangle = 1 \end{array} : \begin{array}{c} \mathsf{srs} \leftarrow \mathsf{S}(1^\kappa, \mathsf{n}) \\ (\mathsf{i}, \mathsf{x}, \mathsf{w}) \leftarrow \mathcal{A}(\mathsf{srs}) \\ (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathsf{I}^{\mathsf{srs}}(\mathsf{i}) \end{array}\right)$$

2. *Knowledge Soundness. For every* $\mathsf{n} \in \mathbb{N}$ *and PPT adversary* $\tilde{\mathsf{P}} = \left(\tilde{\mathsf{P}}_1, \tilde{\mathsf{P}}_2\right)$ *there exists an efficient extractor* Ext *such that the following probability is* $\mathsf{negl}(\kappa)$*.*

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

3. *Zero-knowledge. There exists an efficient simulator* $\mathsf{Sim} = (\mathsf{Setup}, \mathsf{Prove})$ *such that for every efficient adversary* $\tilde{\mathsf{V}} = (\tilde{\mathsf{V}}_1, \tilde{\mathsf{V}}_2)$ *it holds that*

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

$$\Pr\left(\begin{array}{c} (\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}_\mathsf{n} \wedge \\ \left\langle \mathsf{Sim.Prove}(\mathsf{trap}, \mathsf{i}, \mathsf{x}), \tilde{\mathsf{V}}_2(\mathsf{st}) \right\rangle = 1 \end{array} : \begin{array}{c} (\mathsf{srs}, \mathsf{trap}) \leftarrow \mathsf{Sim.Setup}(1^\kappa, \mathsf{n}) \\ (\mathsf{i}, \mathsf{x}, \mathsf{w}, \mathsf{st}) \leftarrow \tilde{\mathsf{V}}_1(\mathsf{srs}) \end{array}\right)$$

4. *Succinctness. An argument system is* proof succinct *if the communication complexity between prover and verifier is bounded by* $\mathsf{poly}(\kappa)$*, and* verifier succinct *if the running time of* V *is bounded by* $\mathsf{poly}(\kappa + |\mathsf{x}|)$ *and independent of the size of the index* i*.*

[4] We use proof and argument as synonymous in this paper, as we are only interested in computational soundness.

An argument is public-coin *if all the messages from the verifier are uniformly random strings of a bounded length. Furthermore, it is* transparent*, if the* srs *is not trusted, that is, the setup algorithm* S *uses only public random coins.*

A public-coin succinct preprocessing argument of knowledge can be compiled using the Fiat-Shamir transformation [FS87] to obtain their non-interactive analogue: preprocessing zkSNARKs with universal SRS.

2.5 Proofs about Exponents

PoE: Proof of Exponentiation We use Wesolowski's proof of exponentiation (PoE) protocol [Wes19] in it's slightly more generalized form as presented in [BBF19] for the relation

$$\mathcal{R}_{PoE} = \{((u, w \in \mathbb{G}, x \in \mathbb{Z}); \perp) : w = u^x \in \mathbb{G})\}$$

PoE (Proof of exponentiation)

Params : $\mathbb{G} \xleftarrow{\$} \mathsf{GGen}(\kappa)$; Inputs : $u, w \in \mathbb{G}, x \in \mathbb{Z}$; Claim : $u^x = w$

1. Verifier sends $l \xleftarrow{\$} \mathbf{Primes}(\kappa)$ to the prover.
2. Prover finds the quotient $q = \lceil x/l \rceil$ and residue $r \in [l]$ such that $x = ql + r$.
   Prover sends $Q := u^q$, to the Verifer.
3. Verifier computes $r = x \bmod l$ and $Q^l u^r = w$.

Theorem 2.11. ***([BBF19] Theorem 1)*** *Protocol* PoE *is an argument system for the relation* $\mathcal{R}_{PoE}$ *with negligible soundness error, assuming the adaptive root assumption holds for* GGen*.*

PoKE: Proof of Knowledge of Exponent For completeness, we have the PoKE protocol from [BBF19] below

$$\mathcal{R}_{PoKE} = \{((u, w \in \mathbb{G}); x \in \mathbb{Z} : w = u^x \in \mathbb{G})\}$$

PoKE (Proof of knowledge of exponent)

Params : $(\mathbb{G}, g) \xleftarrow{\$} \mathsf{GGen}(\kappa)$; Inputs : $u, w \in \mathbb{G}$; Witness : $x \in \mathbb{Z}$;
Claim : $u^x = w$

1. Prover sends $z = g^x \in \mathbb{G}$ to the verifier.
2. Verifer sends $l \xleftarrow{\$} \mathbf{Primes}(\kappa)$.
3. Prover finds the quotient q and residue $r \in [l]$ such that $x = ql + r$.
   Prover sends $Q := u^q, Q' := g^q$ and r to the Verifer.
4. Verifier accepts if $r \in [l], Q^l u^r = w$ and accepts if $(Q')^l g^r = z$.

Theorem 2.12. ***([BBF19] Theorem 3)*** *Protocol PoKE is an argument of knowledge for the relation* $\mathcal{R}_{PoKE}$ *in the generic group model.*

Proof of Knowledge of Squared Exponent (PoKSE) First, we describe a auxiliary protocol PoKSE (Proof of Knowledge of Squared exponent) as a special case of the PoKE from the previous section.

$$\mathcal{R}_{PoKSE} = \{((w \in \mathbb{G}); x \in \mathbb{Z} : w = g^{x^2} \in \mathbb{G})\}$$

PoKSE (Proof of Knowledge of Squared exponent)

Params : $(\mathbb{G}, g) \xleftarrow{\$} \mathsf{GGen}(\kappa)$; Inputs : $w \in \mathbb{G}$; Witness : $x \in \mathbb{Z}$;

Claim : $g^{x^2} = w$

1. Prover sends $z = g^x \in \mathbb{G}$ to the verifier.
2. Verifer sends $l \xleftarrow{\$} \mathbf{Primes}(\kappa)$.
3. Prover finds the quotient q and residue $r \in [l]$ such that $x = ql + r$.
   Prover sends $Q := z^q, Q' := g^q$ and r to the Verifer.
4. Verifier accepts if $r \in [l], Q^l z^r = w$ and $Q'^l g^r = z$.

Theorem 2.13. *Protocol PoKSE is an argument of knowledge for the relation $\mathcal{R}_{PoKSE}$ in the generic group model.*

This follows directly from Theorem 2.12, as a special case.

2.6 Arguing positivity

Recall Lagrange's four-square theorem: *Every natural number can be written as the sum of four integer squares.* Using this theorem and 4 invocations of the PoKSE protocol, we construct an argument of knowledge for the relation $\mathcal{R}_{PoKPE}$. (Note that we can use a trick noted in [Gro05] to reduce the number of calls to three – $(4y + 1)$ is positive iff y is non-negative, and $(4y + 1)$ can always be written as a sum of three squares by Legendre's three-square theorem).

Define the relation $\mathcal{R}_{PoKPE}$ as

$$\mathcal{R}_{PoKPE} = \{((w \in \mathbb{G}); x \in \mathbb{Z} : (w = g^x) \wedge (x > 0))\}$$

Now, consider the following protocol:

PoKPE (Proof of knowledge of positivity of exponent)

Params : $(\mathbb{G}, g) \xleftarrow{\$} \mathsf{GGen}(\kappa)$; Inputs : $w \in \mathbb{G}$; Witness : $x \in \mathbb{Z}$;

Claim : $(g^x = w) \wedge (x > 0)$

1. Prover sets $x = a_1^2 + a_2^2 + a_3^2 + a_4^2 \in \mathbb{Z}$ and $P_i := g^{a_i^2}$.
2. Prover and Verifer engage in PoKSE for each P_i.
3. Verifier accepts if all the PoKSE's output `accept` and at least one $P_i \neq 1$ and if $w = P_1 P_2 P_3 P_4$.

Theorem 2.14. *Protocol PoKPE is an argument of knowledge for the relation $\mathcal{R}_{PoKPE}$ in the generic group model.*

Proof. Since all the PoKSEs pass, there exists an extractor (from PoKSE) that outputs exponents $p_i \in \mathbb{Z}$ such that $P_i = g^{p_i^2}$ for all i. At least one is a positive square due to the check that at least one $P_i \neq 1$.

WLOG let $P_1 \neq 1$, which implies that $p_1 > 0$. Since $w = P_1 P_2 P_3 P_4$, we know that $w = g^{p_1+p_2+p_3+p_4} \wedge (p_1 + p_2 + p_3 + p_4 > 0)$. The extractor for PoKPE simply calls the PoKSE extractors for each P_i to get p_i and outputs $p_1 + p_2 + p_3 + p_4$. □

We also use the notation $\mathsf{PoKPE}\{A, B, \dots\}$ to denote the execution of protocols $\mathsf{PoKPE}(A), \mathsf{PoKPE}(B), \dots$, where the verifier outputs 1 iff the PoKPE checks pass for all elements in the set $\{A, B, \dots\}$.

3 Constant-size Inner Product Commitment Scheme

In this section, we construct an inner product commitment (IPC) scheme. Our focus here is to get a constant size commitment. We construct polynomial commitment schemes in the next two sections building on this IPC.

3.1 Construction

$\mathsf{IPC} = (\mathbf{Setup}, \mathbf{Com}, \mathbf{Open}, \mathbf{Eval})$ are as defined below:

- $\mathbf{Setup}(1^\kappa, D)$: Here, κ is the security parameter and D is an upper bound on the length of the committed vectors. Sample a group of unknown order (we use class groups) $\mathbb{G} \leftarrow \mathsf{GGen}(\kappa)$ and $g \leftarrow\$ \mathbb{G}$. An integer α such that $\alpha > 12Dp^4$ and $\alpha = 0 \mod p$ is selected (p is a large prime such that $\mathsf{len}(p) = \mathsf{poly}(\kappa)$). Return $\mathsf{pp} = (\kappa, \mathbb{G}, g, p, \alpha)$.
- $\mathbf{Com}(\mathsf{pp}, D, f_0, \ldots, f_{l-1}, l)$: Define commitment $C := g^{\sum_{i=0}^{l-1} f_i \alpha^{2i}}$, considering $f_i \in \mathbb{Z}_p$ as integers in $[0, p-1]$. If $l \leq D$, return $(C, \mathbf{f})$, where $\mathbf{f} = (f_0, \ldots, f_{l-1})$ else return error.
- $\mathbf{Open}(\mathsf{pp}, \mathbf{f}, l, C, \tilde{\mathbf{c}})$: Return 1 if all the below conditions hold, else return 0.
  - $l \leq D$
  - $\sum_{i=0}^{d} \tilde{c}_i \alpha^{2i} \in \mathbb{Z}$, $C = g^{\sum_{i=0}^{l-1} \tilde{c}_i \alpha^{2i}}$
  - $\tilde{\mathbf{c}} \in \mathbb{Q}(2,3)^l$, where

$$\mathbb{Q}(\beta_1, \beta_2) := \left\{ \frac{a}{b} : \ \gcd(a,b) = 1, \ \gcd(b,p) = 1, \ 0 < b < p^{\beta_1}, \ |a/b| \leq \beta_2 \alpha \right\},$$

    where $|a|$ denotes the absolute value of $a \in \mathbb{Q}$. (*Note that $\mathbb{Q}(\beta_1, \beta_2)$ is a subset of $\mathbb{Q}(\beta_1', \beta_2')$ if $\beta_1 \leq \beta_1', \beta_2 \leq \beta_2'$*)
  - $\tilde{\mathbf{c}} = \mathbf{f} \mod p$
- $\mathbf{Eval}(\mathsf{pp}, C, l, \mathbf{q}, v; \mathbf{f})$: The **Eval** protocol consists of two sub-protocols **TEST** and **IPP** as described in Fig. 2 and 3 below.
  - $b_1 \leftarrow \mathbf{TEST}(C, l; \mathbf{f})$, $b_2 \leftarrow \mathbf{IPP}(C, l, \mathbf{q}, v; \mathbf{f})$, and
  - If $l \leq D$ return $b = b_1 \wedge b_2$, return 0 otherwise.

To describe the "Computations in **TEST**" and **IPP**, we define a common algorithm $\mathsf{CoeffSplit}$ in Fig. 1 that takes as input α, two integers, and an index of interest i and returns 3 outputs (v, γ, λ).

"Computations in **TEST**" calls

$$\mathsf{CoeffSplit}\left(\alpha, \sum_{j=0}^{l-1} f_j \alpha^{2j}, \sum_{j=0}^{l-1} \alpha^{2l-2-2j} z_j, 2l-1\right)$$

to obtain (v, γ, λ) and defines $\Lambda := g^\lambda$, $\Gamma := g^\gamma$ and outputs (Λ, Γ).
"Computations in **IPP**" calls

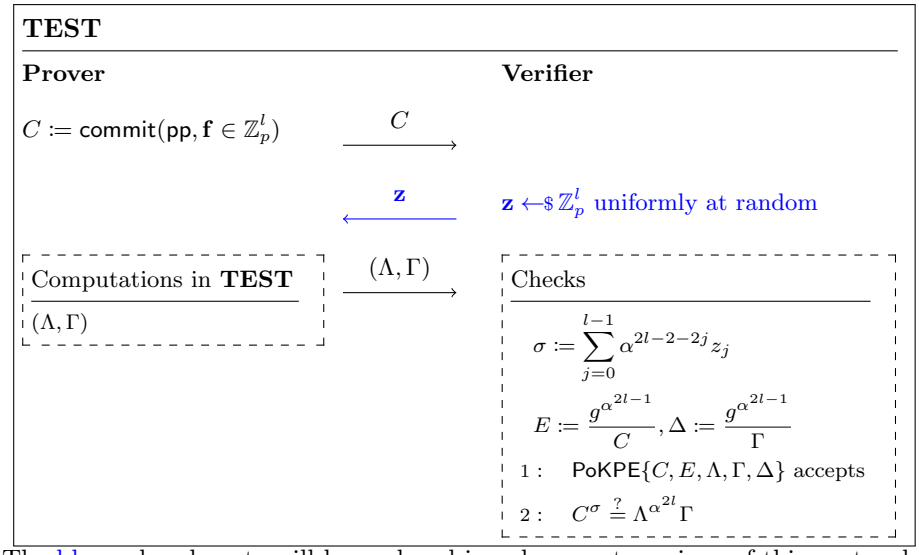
$$\mathsf{CoeffSplit}\left(\alpha, \sum_{j=0}^{l-1} f_j \alpha^{2j}, \sum_{j=0}^{l-1} \alpha^{2l-1-2j} q_j, 2l-1\right)$$

to obtain (v, γ, λ) and defines $\Lambda := g^\lambda$, $\Gamma := g^\gamma$ and outputs $(v \mod p, \left\lfloor \frac{v}{p} \right\rfloor, \Lambda, \Gamma)$.

$\mathsf{CoeffSplit}(\alpha, a, b, i)$

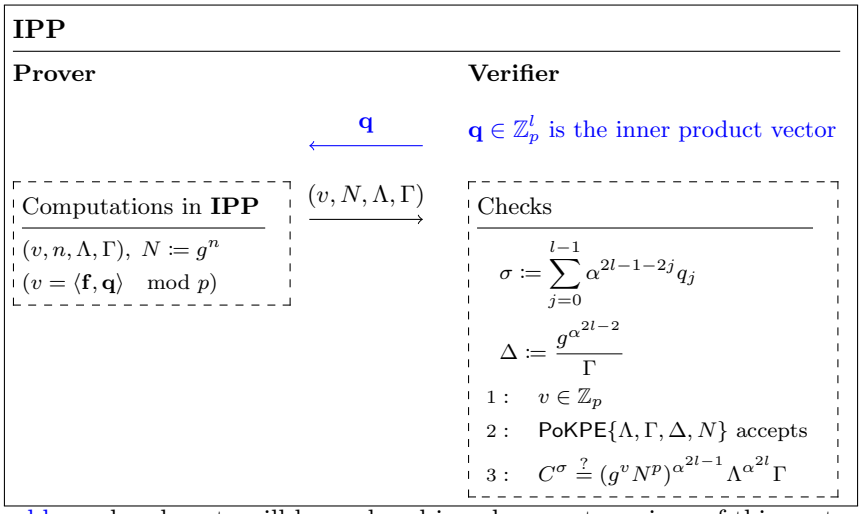
1. Write $a \cdot b$ in base α, call the resulting representation vector $\mathbf{c}$.
2. Set $v := c_i$, $\gamma := \sum_{j=0}^{i-1} c_j \alpha^j$, $\lambda := \sum_{j=i+1}^{\lceil \log_\alpha ab \rceil} c_j \alpha^j$
3. Output the tuple (v, γ, λ).

Figure 1: $\mathsf{CoeffSplit}$



The blue colored parts will be replaced in subsequent versions of this protocol.

Figure 2: The **TEST** Protocol



The blue colored parts will be replaced in subsequent versions of this protocol.

Figure 3: The **IPP** Protocol

3.2 Proofs of Security

In this section, we prove that our construction IPC satisfies the requirements of an inner product scheme as defined in §2.1.

Theorem 3.1 (Completeness)**.** *The inner product commitment scheme* IPC *satisfies Completeness (Definition 2.1).*

Proof. Note that by definition of CoeffSplit and completeness of PoKPE, all the PoKPE checks will accept.

To show that the last checks in **TEST** and **IPP** hold, it suffices to show that $v = 0$ in **TEST** and $v = \langle \mathbf{f}, \mathbf{q} \rangle \mod p$ in **IPP**. We will show this by expanding the computations done in CoeffSplit.

In **TEST**, direct manipulation shows

$$\sum_{j=0}^{l-1} f_j \alpha^{2j} \times \sum_{j=0}^{l-1} \alpha^{2l-2-2j} z_j$$
$$= \alpha^{2l} \underbrace{\left(\sum_{j'>j} \alpha^{2(j'-j)-2} f_{j'} z_j \right)}_{\lambda} + \underbrace{\left(\sum_{j'<j} \alpha^{2l-2-2(j-j')} f_{j'} z_j + \sum_{j'=j} \alpha^{2l-2} f_j z_j \right)}_{\gamma}$$

and notice that since $\alpha > lp^2$, these are indeed the γ, λ returned by $\mathsf{CoeffSplit}$, and $v = 0$.

And, in **IPP**,

$$\sum_{j=0}^{l-1} f_j \alpha^{2j} \times \sum_{j=0}^{l-1} \alpha^{2l-1-2j} q_j = \alpha^{2l-1} \left(\underbrace{\sum_{i=0}^{l-1} f_j q_j \mod p}_{v} + p \underbrace{\left\lfloor \frac{\sum_{i=0}^{l-1} f_j q_j}{p} \right\rfloor}_{n} \right)$$
$$+ \alpha^{2l-2}(0) + \alpha^{2l} \underbrace{\left(\sum_{j'>j} \alpha^{2(j'-j)-1} f_{j'} q_j \right)}_{\lambda} + \underbrace{\left(\sum_{j'<j} \alpha^{2l-1-2(j-j')} f_{j'} q_j \right)}_{\gamma}$$

Here, again since $\alpha > lp^2$, $(v + np, \lambda, \gamma)$ above coincide with the output of $\mathsf{CoeffSplit}$, and $v = \langle \mathbf{f}, \mathbf{q} \rangle \mod p$. □

Theorem 3.2 (Binding)**.** *The inner product commitment scheme* IPC *Construction in §3.1 is binding (Definition 2.2) for opening hint vectors in* $\mathbb{Q}(\beta_1, \beta_2)$ *if* $\alpha > 4\beta_2 p^{2\beta_1}$ *and if the Order assumption holds for* GGen*.*

Proof. Suppose there exists an adversary $\mathcal{A}$ which breaks binding as defined in Definition 2.2, i.e., $\mathcal{A}(\mathsf{pp})$ outputs $(C, \mathbf{f}, \mathbf{f}', \mathbf{c}, \mathbf{c}', d)$ such that $\mathsf{open}(\mathsf{pp}, \mathbf{f}, d, C, \mathbf{c}) = 1$ and $\mathsf{open}(\mathsf{pp}, \mathbf{f}', d, C, \mathbf{c}') = 1$ but $\mathbf{f} \neq \mathbf{f}'$ (which also implies that $\mathbf{c} \neq \mathbf{c}'$ – we will use this condition to show a contradiction). Then, since open outputs 1 for both $\mathbf{f}, \mathbf{f}'$ we know that the opening hints $\mathbf{c}, \mathbf{c}' \in \mathbb{Q}(\beta_1, \beta_2)^l$ and that

$$g^{\sum_{i=0}^{l-1} c_i \alpha^{2i}} = g^{\sum_{i=0}^{l-1} c'_i \alpha^{2i}} \iff g^{\sum_{i=0}^{l-1} (c_i - c'_i) \alpha^{2i}} = 1.$$

If the exponent of g above were not zero, we could construct an adversary $\mathcal{A}_{Ord}$ that uses the above exponent to break the Order assumption (Assumption 1). Now, let the exponents be equal, and consider the largest index j such that $c'_i \neq c_i$ (WLOG, let $c'_i > c_i$). This implies that $\sum_{i=0}^{j-1} (c_i - c'_i) \alpha^{2i} = (c'_j - c_j) \alpha^{2j}$.

We can now show that this equality is impossible given the conditions on α, β_1, β_2.
Notice that any difference $|c'_i - c_i|$ (if non-zero) can be bounded by $\frac{1}{p^{2\beta_1}} < |c'_i - c_i| < 2\beta_2 \alpha$,

since $c_i, c'_i \in \mathbb{Q}(\beta_1, \beta_2)$. This implies that

$$\sum_{i=0}^{j-1} (c_i - c'_i) \alpha^{2i} < 2\beta_2 \alpha \sum_{i=0}^{j-1} \alpha^{2i} < 2\beta_2 \alpha \cdot 2\alpha^{2j-2} < \frac{\alpha^{2j}}{p^{2\beta_1}} < (c'_j - c_j) \alpha^{2j},$$

which is a contradiction. □

Before proving extractability, we need a few definitions. Define the set

$$S \coloneqq \left\{ \frac{m\alpha - n}{k} : m, n, k \in \mathbb{Z},\ gcd(m, k) = 1,\ 0 < m \leq k < p,\ 0 \leq n < k + 2 \right\}$$

and functions $\chi_m, \chi_n : S^q \to \mathbb{Q}^q$ which isolates the vector of fractions m/k and n/k from the elements of S^q:

$$\mathbf{v} \in S^q \implies \mathbf{v} = \left(\frac{m_i\alpha - n_i}{k_i}\right)_i, \quad \chi_m(\mathbf{v}) := \left(\frac{m_i}{k_i}\right)_i, \quad \text{and} \quad \chi_n(\mathbf{v}) := \left(\frac{n_i}{k_i}\right)_i.$$

These functions can be made well-defined by fixing a representation of elements of S: for any $d \in S$, consider the representation (m, n, k) as the one with the smallest denominator k and if there are multiple such representations, we pick the one with the smallest m.

For a (multi-)set of rational numbers F in reduced form (a rational $q = a/b$ is in reduced form if), define DenLCM(F) as the *LCM of the denominators* of elements of F, $\text{lcm}(\{b \ : \ a/b \in F\})$. For a vector $\mathbf{f}$, DenLCM($\mathbf{f}$) is defined to be the DenLCM of the set of its coefficients.

Theorem 3.3 (Extractability)**.** *The inner product commitment scheme* IPC *satisfies Extractability for* $(\beta_1, \beta_2) = (2, 3)$ *(Def. 2.4) in the Generic Group Model.*

Proof. We split the proof into two theorems; the first theorem (concerning **TEST**) will define a partial extractor and obtain conditions on the extracted objects, while the second theorem uses the results of the first one and proves the required conditions on the final extracted object.
Suppose there exists a *generic* adversary $\mathcal{A}$ that makes the Verifier in eval accept with non-negligible probability (and hence both the **TEST** verifier and **IPP** verifier). We will construct a polynomial time extractor Ext that outputs $(\tilde{\mathbf{f}}, \tilde{\mathbf{c}})$ satisfying $\mathcal{R}_{\mathsf{eval}}$ (in Def 2.7) with overwhelming probability.

Theorem 3.4 (**TEST** Extractor)**.** *If the Verifier in* **TEST** *outputs accept with non-negligible probability over the choice of the random* $\mathbf{z} \in \mathbb{Z}_p^l$*, there exists an efficient extractor* Ext_T *in the GGM that outputs vectors* $\mathbf{c}, \mathbf{d} \in \mathbb{Z}_p^l$ *satisfying the following properties:*

1. $C = g^{\sum_{i=0}^{l-1}(c_i + \alpha d_i)\alpha^{2i}}$ *– proved in Lemma 3.10*
2. $\mathbf{d} \in S^l$ *– proved in Lemma 3.11*
3. $\text{DenLCM}(\chi_m(\mathbf{d})) < p$ *– proved in Lemma 3.12*

Theorem 3.5 (**IPP** Extractor)**.** *If the Verifier in* **IPP** *outputs accept with non-negligible probability for some query vector* $\mathbf{q}$*, and given that the Verifier of* **TEST** *also did so, there exists an efficient extractor* Ext *in the GGM that outputs an opening* $\tilde{\mathbf{f}} \in \mathbb{Z}_p^l$ *and a valid opening hint* $\tilde{\mathbf{c}}$ *in* $\mathbb{Q}(2, 3)$ *for* C *such that* $v = \langle \tilde{\mathbf{f}}, \mathbf{q} \rangle \mod p$*.*

These theorems are proved in the next section (§ 3.3). □

3.3 Proofs of TEST and IPP Extractors

Auxiliary Lemmas. Before we prove Theorems 3.4 and 3.5, we state four auxiliary lemmas (Lemmas 3.6, 3.7, 3.8, and 3.9) that go into their proofs. Proofs of the auxiliary lemmas appear in Appendix §A.

Lemma 3.6. *Given a group* G *and* $g \in G$*, if there exist group elements* $W, X_1, \ldots, X_m$ *such that* $W = \prod_{i=1}^m X_i$ *and a prover succeeds in convincing a* PoKPE *verifier for all of these elements w.r.t. the base* g*, either* 1 *or* 2 *below hold:*

1. *the prover knows exponents* $w, x_1, \ldots, x_m$ *for* $W, X_1, \ldots, X_m$ *respectively such that* $w, x_i > 0$ *and* $w = \sum_{i=1}^m x_i$*.*
2. *the prover knows (a multiple of) the order of* g*, and can break the Order Assumption (Assumption 1).*

Lemma 3.7. *Suppose* $K = \sum_{i=0}^{k} M_i \alpha^i$ *where* M_i*'s are not necessarily* $< \alpha$*, but we have a bound* $M_i < \alpha(\alpha - 1)\ \forall\, i$*. Then, we can write* $K = \sum_{i=0}^{k+1} U_i \alpha^i$ *where each* $U_i < \alpha$*, and*

$$U_i := \begin{cases} (M_0 \mod \alpha + u_0) \mod \alpha & \text{if } i = 0 \\ (M_i \mod \alpha + \left\lfloor \frac{M_{i-1}}{\alpha} \right\rfloor + u_i) \mod \alpha & \text{if } 1 \le i \le k \\ (\left\lfloor \frac{M_k}{\alpha} \right\rfloor + u_{k+1}) & \text{if } i = k+1 \end{cases}$$

$$u_i := \begin{cases} 0 & \text{if } i = 0 \\ \left\lfloor \frac{M_{i-1} \mod \alpha + \left\lfloor \frac{M_{i-2}}{\alpha} \right\rfloor + u_{i-1}}{\alpha} \right\rfloor & \text{if } 1 \le i \le k+1 \end{cases}$$

Also, $\forall\, i \;\; u_i \in \{0, 1\}$.

Lemma 3.8. *Suppose for some* α*,* $M'\alpha - N' = M\alpha - N$*, where* $M', N' \in \mathbb{Q}$ *and* $M, N \in \mathbb{Z}$*.*
If $|N|, |N'| < B$*, and* $M' = \frac{x}{y}$ *for the smallest possible* y *and* $y < \frac{\alpha}{2B}$*, then* $M' = M$ *and* $N' = N$*.*

Lemma 3.9. *Given* $k, p \in \mathbb{Z}$ *and* $n < \frac{p}{2}$*, then any polynomial* $f(x_1, \ldots, x_n) \coloneqq \sum_{i=1}^{n} m_i x_i$ *is such that for all* $c \in \mathbb{Z}_p$ *and* $b = GCD(m_1, \ldots, m_n, k)$*,*

$$\Pr_{x_i \leftarrow\$ \mathbb{Z}_p} \left(\sum_{i=1}^{n} m_i x_i = c \mod k \right) \le \frac{b}{k} + \frac{n}{p}.$$

We now prove Theorems 3.4 and 3.5 (proof of Extractability).

Lemma 3.10 (Part (1) of **TEST** extraction)**.** *If the* **TEST** *Verifier accepts with non-negligible probability over the choice of the random query* $\mathbf{z}$*, there exists an extractor that outputs vectors* $\mathbf{c}, \mathbf{d} \in \mathbb{Z}_p^l$ *satisfying* $C = g^{\sum_{i=0}^{l-1} (c_i + \alpha d_i) \alpha^{2i}}$.

The extractor Ext_T invokes the PoKPE extractor for C, which outputs an exponent $c > 0$ such that $C = g^c$. Since E also passes the PoKPE protocol and $C \cdot E = g^{\alpha^{2l-1}}$, we can infer that $c < \alpha^{2l-1}$. Consider the base-α representation of c, which is a $2l$-length vector. Ext_T outputs the even indexed coordinates as $\mathbf{c}$ and the odd indexed coordinates as $\mathbf{d}$.

Note that by definition, the first condition in the theorem is satisfied: $C = g^{\sum_{i=0}^{l-1} (c_i + \alpha d_i) \alpha^{2i}}$. An honest prover would clearly choose $d_i = 0$ and $c_i = x_i$, $0 \le c_i \le p - 1$ for $0 \le i \le l - 1$ to commit to a vector $\mathbf{x} \in \mathbb{Z}_p^l$. However, with a cheating prover, we are only guaranteed that $0 \le c_i, d_i \le \alpha - 1$.

Lemma 3.11 (Part (2) of **TEST** extraction)**.** *For all* $0 \le i \le l - 1$*,* $d_i \in S$*.*

Suppose the prover succeeds with non-negligible probability over the choice of random $\mathbf{z} \in \mathbb{Z}_p^l$. Fix some arbitrary index $0 \le i \le l - 1$.
Consider a partition of $\mathbb{Z}_p^l$ by sets of the form

$$T_{\mathbf{q}} := \{(z_i, \mathbf{q}) \;:\; z_i \in \mathbb{Z}_p\}$$

for every $\mathbf{q} \in \mathbb{Z}_p^{l-1}$ (each set contains p points, each of which differ only in the i^{th} coordinate). Since $\frac{1}{p}$ is negligible, there exists at least one $T_{\mathbf{q}}$ which contains more than one point for which the prover is able to succeed (we will refer to these as accepting points). Call this set T^*, and the two accepting points $\mathbf{z_1}, \mathbf{z_2}$ (for simplicity, we will use z_1, z_2 to denote the i^{th} coordinate that differs in both these vectors).

Then, we can look at the final check in **TEST** (Fig. 2, Check 2) and using Lemma 3.6, we get an integer equality and Lemma 3.7 gives us two equations for the $(2l-1)^{th}$ index with $M_{2l-1} = \langle \mathbf{d}, \mathbf{z} \rangle$ and $M_{2l-2} = \langle \mathbf{c}, \mathbf{z} \rangle$. Since each M_i is an inner product of two vectors, one of which has coordinates $< \alpha$ and the other $< p$, $\implies M_i < \alpha(\alpha - 1)$.

$$\langle \mathbf{d}, \mathbf{z_1} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_1} \rangle}{\alpha} \right\rfloor + u_1 \mod \alpha = 0 \mod \alpha \tag{3}$$

$$\langle \mathbf{d}, \mathbf{z_2} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_2} \rangle}{\alpha} \right\rfloor + u_2 \mod \alpha = 0 \mod \alpha \tag{4}$$

where $u_1, u_2 \in \{0,1\}$. We can subtract the two equations to obtain bounds on d_i (WLOG, let $z_2 > z_1$):

$$(z_2 - z_1)d_i = -\left(\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_2}\rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_1}\rangle}{\alpha} \right\rfloor + u_2 - u_1\right) \mod \alpha \tag{5}$$

Let the term in the brackets on the RHS be n. Using the fact that $x - 1 \leq \lfloor x \rfloor < x$ and $u_i \in \{0,1\}$ for all i gives us trivial bounds $-1 < n < (z_2 - z_1) + 2$. We know that $0 < z_2 - z_1 < p$, hence $d_i \in S$. Since i was an arbitrary index, $\mathbf{d} \in S^l$.

Lemma 3.12 (Part (3) of **TEST** extraction)**.** $\text{DenLCM}(\chi_m(\mathbf{d})) < p$.

Suppose $\text{DenLCM}(\chi_m(\mathbf{d})) > p$ (this is well-defined as $\mathbf{d} \in S^l$ from Lemma 3.11).
Then, we can always find a subset J of the indices $[0, l-1]$ such that $p < \text{DenLCM}(\chi_m(\mathbf{d}|_J)) < p^2$. J is non-empty because each d_i contributes at most a factor of $(p-1)$ to $\text{DenLCM}(\chi_m(\mathbf{d}))$. For notational simplicity, assume WLOG that J consists of the first $|J|$ indices.

Now consider a partition of $\mathbb{Z}_p^l$ by sets of the form

$$T_{\mathbf{q}} \coloneqq \{(z_0, \ldots, z_{|J|-1}, \mathbf{q})\}$$

for every $\mathbf{q} \in \mathbb{Z}_p^{l-|J|}$ – each set has $p^{|J|}$ points. We will show that no set can contain more than 2 accepting points, which contradicts the fact that the verifier in **TEST** outputs accept with non-negligible probability.

Consider any set T^* and pick any two accepting points in the set. We can derive a condition on the difference of any two accepting points, which in turn gives us a bound on the maximum number of accepting points.
Again, take any two accepting points in the set T^*, call them $\mathbf{z}, \mathbf{z}'$ such that they differ only in the first $|J|$ coordinates. Lemmas 3.6, 3.7 give us two equations similar to (3) and (4) with $\mathbf{z}, \mathbf{z}'$ instead of $\mathbf{z_1}, \mathbf{z_2}$: Subtracting the two equations retains only the $|J|$ coordinates of the first term (WLOG let the LHS be positive, if not, relabel $\mathbf{z}, \mathbf{z}'$):

$$\sum_{i=0}^{|J|-1} d_i(z_i' - z_i) = -\left(\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}'\rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}\rangle}{\alpha} \right\rfloor + u' - u\right) \mod \alpha$$

Note that the negative term on the RHS (call it N) is bounded in absolute value by $p|J| + 2$ since $\mathbf{z}$ and $\mathbf{z}'$ differ in at most $|J|$ indices. The equation can be now written as $\sum_{i=0}^{|J|-1} d_i(z_i' - z_i) = M\alpha - N$. From Lemma 3.11, we know that every d_i can be written in the form $\frac{m_i\alpha - n_i}{k_i}$, and we can define two additional quantities M', N'. Let $M' \coloneqq \sum_{i=0}^{|J|-1} \frac{m_i(z_i'-z_i)}{k_i}$ and $N' \coloneqq \sum_{i=0}^{|J|-1} \frac{n_i(z_i'-z_i)}{k_i}$. Due to the conditions on m_i, n_i, k_i, z_i we know that $M' < p|J|, N' < 2p|J|$ and that the smallest denominator of M' when written as a rational is the LCM of all the k_i, which is at most p^2.

Since $\alpha > 4lp^3$, using Lemma 3.8 we can conclude that $M, N \in \mathbb{Z}$.
However, $M \in \mathbb{Z}$ can only happen for certain choices of $(z_0, \ldots, z_{|J|-1}) \in \mathbb{Z}_p^{|J|}$. We can re-write the condition $M \in \mathbb{Z}$ as:

$$\sum_{i=0}^{|J|-1} \frac{m_i L}{k_i} y_i = 0 \mod L$$

where $L \coloneqq \text{DenLCM}(k_0, \ldots, k_{|J|-1})$ and y_i is uniform over $\mathbb{Z}_p$ and is representative of candidate differences. Note that the GCD of all the (integer) coefficients is 1 (since $\gcd(m_i, k_i) = 1$ for all i).

Using Lemma 3.9, this probability is negligible when $L > p$.
Note that we have proved that there is only a negligible fraction of *differences of* accepting points in any partitioning set T^*. However, given a bound on differences of a set of points, the maximum number of points that can exist in the set is simply bounded by the same number plus one – precisely when all the points are arranged in a line (arithmetic progression) at a constant distance from each other (if there are B distinct numbers in a set, there are at least $B-1$ unique elements in the set $\{|b_1 - b_2| : b_1 \neq b_2 \in B\}$). Hence, we have shown that the verifier in **TEST** will output accept only with negligible probability, which is a contradiction to our initial assumption.

3.4 Proof of Theorem 3.5 – IPP extraction

We are now ready to prove Theorem 3.5. We define the extractor Ext for eval using the extractor Ext_T of **TEST**. Since the **TEST** verifier outputs accept with non-negligible probability, there exists Ext_T that outputs vectors $\mathbf{c}, \mathbf{d}$ such that they satisfy the conditions in Theorem 3.4. Ext invokes Ext_T and performs the following computations on $\mathbf{c}, \mathbf{d}$:

1. Compute m_i, n_i, k_i for every i such that $d_i = \frac{m_i\alpha - n_i}{k_i}$.
2. Let $m_{-1} \coloneqq 0$, $k_{-1} \coloneqq 1$ and define vectors $\mathbf{c}', \mathbf{d}' \in \mathbb{Z}_p^l$ as $c'_i \coloneqq c_i + \frac{m_{i-1}}{k_{i-1}}$ and $d'_i \coloneqq -\frac{n_i}{k_i}$.
3. Output $\mathbf{c}' + \alpha\mathbf{d}'$ as the opening hint, and $(\mathbf{c}' + \alpha\mathbf{d}') \mod p$ as the opening to the commitment.

Note that this extractor is indeed efficient, since Ext_T is efficient and the only non-trivial computations done are in Step 1 above. This can be done efficiently, details are given in Lemma 4.9.

By construction, this is a valid opening hint, as $\mathbf{d} \in S^l \implies \mathbf{c}' + \alpha\mathbf{d}' \in \mathbb{Q}(2,3)$ and since it is just a rearrangement of the coordinates of $\mathbf{c}$ and $\mathbf{d}$ while keeping the sum $\sum_{i=0}^{l-1}(c'_i + \alpha d'_i)\alpha^{2i}$ equal to the previous sum $\sum_{i=0}^{l-1}(c_i + \alpha d_i)\alpha^{2i}$ (since we just move the coefficient of α in d_i to c_{i+1}), we have $C = g^{\sum_{i=0}^{l-1}(c'_i + \alpha d'_i)\alpha^{2i}}$, which is one of the required conditions.

Now, since the verifier accepts in **IPP**, Lemmas 3.6 and 3.7 along with the final check of the verifier (check 3) gives us equations corresponding to the coefficients of α^{2l-2} and α^{2l-1}, which given that $M_{2l-3} = \langle \mathbf{c}, \mathbf{q}^+ \rangle$, $M_{2l-2} = \langle \mathbf{d}, \mathbf{q}^+ \rangle$ and $M_{2l-1} = \langle \mathbf{c}, \mathbf{q} \rangle$ are:

$$\langle \mathbf{d}, \mathbf{q}^+ \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u' = 0 \mod \alpha \tag{6}$$

$$\langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u = v + np \mod \alpha \tag{7}$$

where $\mathbf{q}^+$ is defined as the vector with elements $q_i^+ \coloneqq q_{i+1} \, \forall \, i \in \{0, \ldots, l-2\}$, $q_{l-1}^+ \coloneqq 0$ and $u, u' \in \{0, 1\}$. Due to the bounds on coefficients of $\mathbf{q}$, we know that Equation 6's LHS must be either 0 or α. Also define

$$M' \coloneqq \sum_{i=0}^{l-1} \frac{m_i}{k_i} q_i^+, \quad \text{and} \quad N' \coloneqq \sum_{i=0}^{l-1} \frac{n_i}{k_i} q_i^+$$

1. If the LHS is 0, then so are each of the terms in the LHS, as they are all non-negative. Hence, $\langle \mathbf{d}, \mathbf{q}^+ \rangle = 0 \mod \alpha$, and by definition, $u = 0$ (Lemma 3.7).
   Hence, we can simplify Equation 7

$$\begin{aligned} v + np = \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u \mod \alpha \\ = \langle \mathbf{c}, \mathbf{q} \rangle + \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \mod \alpha = \langle \mathbf{c}, \mathbf{q} \rangle + M' - \frac{N'}{\alpha} \mod \alpha \end{aligned}$$

   $M' - N'/\alpha$ is an integer and $N' < \alpha$ by choice of α and $\mathbf{q} \implies N' = 0$.

$$\begin{aligned} v + np = \langle \mathbf{c}, \mathbf{q} \rangle + M' \mod \alpha = \langle \mathbf{c}', \mathbf{q} \rangle \mod \alpha \\ \implies v = \langle \mathbf{c}', \mathbf{q} \rangle \mod p = \langle \mathbf{c}', \mathbf{q} \rangle + \alpha \langle \mathbf{d}', \mathbf{q} \rangle \mod p, \end{aligned}$$

   as $\alpha = 0 \mod p$ and $\langle \mathbf{d}', \mathbf{q} \rangle$ is invertible modulo p (or simply $0 \mod p$). In either case, $v = \langle \mathbf{c}' + \alpha\mathbf{d}', \mathbf{q} \rangle \mod p$.

2. If the LHS is α, we get that $u = 1$ by definition. From the properties of $\mathbf{c}$ and $\mathbf{d}$ as shown in Lemma 3.12, we know that $M' \in \mathbb{Q}$ has a denominator of at most p (in the reduced form). Write Equation 6 in the form $\langle \mathbf{d}, \mathbf{q}^+ \rangle = M\alpha - N$, by moving all the terms but the inner product to the

RHS and calling it N. Now, Lemma 3.8 implies that $M', N' \in \mathbb{Z}$ (since $\alpha > 4lp^2$). Then,

$$\begin{aligned} v + np = \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u \mod \alpha \\ = \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor M' - \frac{N'}{\alpha} \right\rfloor + 1 \mod \alpha \\ = \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + M' - 1 + 1 \mod \alpha \end{aligned}$$

as $N' < \alpha$. Hence, as before, we get that $v = \langle \mathbf{c}' + \alpha \mathbf{d}', \mathbf{q} \rangle \mod p$.

Thus, the extracted opening equals the claimed inner product v in both cases.

3.5 Optimizations

3.5.1 Using a PRG to generate the TEST query vector

The communication (and hence the proof size) in the **TEST** protocol is linear in the size of the committed vector, due to the random vector $\mathbf{z} \in \mathbb{Z}_p^l$ being sent by the verifier. Instead of the Verifier sending the vector, the prover and verifier can agree on a PRG that outputs vectors in $\mathbb{Z}_p^l$ beforehand, and the verifier instead sends a short seed to the prover, and both the parties locally compute the vector.

To show that this does not affect the soundness of the **TEST** protocol, consider two hybrids H_0 and H_1 where the difference between the two hybrids is the query vector, which is generated and sent to the prover using true randomness in H_0 and using the PRG in H_1.
Suppose there was exists some prover $\mathcal{A}$ that succeeds in cheating in H_0 with probability p_0 and probability p_1 in H_1. Consider a distinguisher $\mathcal{D}$ for the PRG, which runs the prover in the two hybrids and outputs "PRG" if the prover succeeds and "random" if the prover fails. The distinguishing probability is precisely $\frac{1}{2} + \frac{p_1 - p_0}{2}$, which has non-negligible advantage over $\frac{1}{2}$ iff $p_1 - p_0$ is non-negligible. Hence, no prover can succeed in breaking soundness with non-negligible probability when the random vector $\mathbf{z}$ is replaced by the output of a PRG on a random seed.

3.5.2 Using Proofs of Exponentiation

The **TEST** protocol contains checks on group elements which involve raising the group elements to large powers like α^{2l}. This computation, if done locally by the verifier is very inefficient. Hence, we use a Proof of Exponentiation (PoE) [Wes19, Pie18] to outsource the computation to the prover.
For completeness, the protocol is described in Section 2.5. This protocol is public-coin, has constant communication and verifier time.

3.5.4 Batching PoKE proofs

We perform a (constant) number of PoKE proofs in our protocol, which can all be batched using techniques from [BBF19] (the PoKCR protocol) to concretely reduce the communication cost of our protocol. This can be combined with an optimisation in the PoKPE protocol (Def 2.6) where we can represent a number as a sum of three squares instead of four [Gro05], and hence only require 3 calls to PoKE.

3.5.5 Evaluating a single commitment at multiple points

The protocols **TEST** and **IPP** can be extended to the case of evaluating a single committed polynomial at a vector of evaluation points $\mathbf{x}$, and claimed evaluations $\mathbf{y}$ in $\mathbb{Z}_p^n$, say.
Given a commitment C, the Prover and Verifier first engage in **TEST** for C. If the Verifier accepts, the Prover then sends the claimed evaluations of the committed polynomial at $\mathbf{x}$ to the Verifier. Then, they run the **IPP** protocol with the query $\sum_{i=0}^{n} z_i \mathbf{q_i}$, where $\mathbf{q_i}$ are the queries corresponding to each x_i as defined in the **IPP** protocol in Section 3.1, and $z_i \in \mathbb{Z}_p$ are chosen uniformly at random. The claimed output (used for the checks) is naturally taken as $\sum_i z_i y_i \mod p$.

When $n \ll p$ and the Verifier accepts, the **Eval** extractor works the same as in the soundness proof (Theorem 3.3), and outputs an opening $f(X) \in \mathbb{Z}_p[X]$ to C such that

$$\sum_{i=1}^{n} z_i f(x_i) = \sum_{i=1}^{n} z_i y_i \mod p$$

A single application of the Schwartz-Zippel lemma gives us the desired property.

3.5.6 Aggregating multiple commitments

When multiple inner products need to be computed with the same vector, the Prover first commits to each polynomial in a separate commitment. Given commitments $C_1, C_2, \ldots, C_m$, and claimed inner products $\mathbf{y} \in \mathbb{Z}_p^m$ with some fixed vector $\mathbf{q}$, we would like a aggregated proof that is more efficient (succinct) that running the evaluation protocol m times.

This can be done in a standard way; both the Prover and Verifier compute a randomly combined commitment $C := C_1 C_2^z \cdots C_m^{z^{m-1}}$, and run **Eval** on the combined commitment C with the query vector $\mathbf{q}$ and with the claimed output $\sum_{i=1}^{m} z^{i-1} y_i \mod p$. Note that we can also use random powers for each commitment instead of powers of a single element. The only difference in the proof is that a Vandermonde matrix is always invertible, as opposed to a random matrix.

We also need to slightly relax the opening requirements of our scheme, by allowing opening hints in the set $\mathbb{Q}(m\beta_1 + m^2 - m, mp^m\beta_2)^l$ as valid opening hints - this in turn gives a lower bound on α according to Theorem 3.2 since the extracted openings hints need to be within the binding space.

Theorem 3.13. *When the verifier accepts in* **Eval** *performed as above with non-negligible probability over the choice of $z \in \mathbb{Z}_p$, there exists an extractor that outputs with high probability for all $i \in [m]$ openings (and opening hints) to each commitment C_i such that $\langle \mathbf{f_i}, \mathbf{q} \rangle = y_i \mod p$ for $C_i = \mathbf{Com}(\mathbf{f_i})$.*

Proof. Since the Verifier accepts, we get an extractor that outputs an opening hint $\mathbf{f} \in \mathbb{Q}(2,3)^l$ for the commitment C.

Consider m different accepting transcripts, which differ in the random choice of z, call them $z_1, \ldots, z_m$ (and the combined commitments as D_i, $i \in [m]$). The extractor can obtain this efficiently by rewinding, because we assume that the Prover succeeds with non-negligible probability.
For this paragraph, write the group equalities in additive notation, we get equations of the form

$$D_i = \sum_{j=1}^{m} z_i^{j-1} C_j$$

We can think of these as m different linear equations, which are always guaranteed a solution, as the coefficient matrix is a Vandermonde matrix. The solutions are of the form $C_j = \sum_{i=1}^{m} b_{ij} D_i$ for all commitments C_j, where b_{ij} are the entries of the j^{th} row of the inverse matrix.

$$b_{ij} = \frac{(-1)^{n-i} e_{n-i}(\{z_1, z_2, \ldots, z_m\}\{z_j\})}{\prod_{h=1, h\neq j}^{m} (z_j - z_m)}$$

where $e_\cdot(\cdot)$ is the elementary symmetric function. Note that $|b_{ij}| < p^m$ and the smallest non-negative value it can take is $> \frac{1}{p^{m-1}}$.

We can then check that since $D_i = g^{\sum_{k=0}^{l-1} f_{ik}\alpha^{2k}}$ for $f_{ik} \in \mathbb{Q}(2,3)$ (as **TEST** passes with high probability), the coefficients $g_{jk} := \sum_{i=1}^{m} b_{ij} f_{ik}$ in the exponent of C_j lie in the larger set $\mathbb{Q}(m^2+m, 3mp^m)$. Since $\alpha > 12mp^{2m^2+3m}$, these vectors are valid opening hints, and are within the binding space of the commitment scheme. The openings can be computed as $\mathbf{f_i} \mod p$.
To check that all these vectors satisfy $\langle \mathbf{f_i}, \mathbf{q} \rangle = y_i \mod p$, we can use the Schwartz-Zippel lemma on the statement (which is true since **Eval** succeeds with high probability)

$$\sum_{i=1}^{m} z^{i-1} \langle \mathbf{f_i}, \mathbf{q} \rangle = \sum_{i=1}^{m} z^{i-1} y_i \mod p$$

for $z \in \mathbb{Z}_p$ and since $m \ll p$, the above polynomial is the zero polynomial with high probability. □

4 Constant-size PCS with $\sqrt{l}$-time verifier

For the inner product scheme in Construction 3.1, note that the verifier has to do work proportional to the length of the query vector. This seems to be the best we can do for inner products when the query vector is allowed to be arbitrary. However, for a polynomial commitment scheme, since the query vector is a function of a single evaluation point, we can hope to achieve significantly better verifier efficiency.

The bottleneck for verifier computation is computing σ in the **TEST** and **IPP** protocols. In this section, we redesign these protocols – specialising them to polynomial commitment schemes – such that σ can be computed efficiently in both cases and the verifier's work is now sublinear – concretely, $\sqrt{l}$ – in the length l of the vector (here, the degree of the polynomial). In Appendix B, we show how to generalize these ideas to get complexity l^ϵ for any constant $0 < \epsilon \leq 1$. Intuitively, in both these constructions, we replace the random length-l query vector in **TEST** with a Kronecker product of random vectors of length $\sqrt{l}$ or $l^{1/e}$, respectively. The technical arguments in this section and § B to prove correctness of these modifications to **TEST**, however, work only for a *constant* e. These techniques can in fact be further generalized to achieve a $\log l$ verification complexity (described in §5).

4.1 Construction of Polynomial Commitment Scheme sqPC

We define a PCS with constant sized proofs and sublinear verification by adapting our inner product scheme from Section 3.1. The main differences are in the $\mathbf{TEST}_2$ and $\mathbf{IPP}_{sub}$ protocols (which form eval for sqPC), which are the same as **TEST** and **IPP** in Section 3.1 except for the query vectors as described below.

1. $\mathbf{TEST}_2$: The random query vector $\mathbf{z} \in \mathbb{Z}_p^l$ in $\mathbf{TEST}_2$ is now defined[5] as a function of $2\sqrt{l}$ uniformly random elements $(x_1, \ldots, x_{\sqrt{l}}, y_1, \ldots, y_{\sqrt{l}}) \in \mathbb{Z}_p^{2\sqrt{l}}$:

$$\text{For } i, j \in [\sqrt{l}] \;\; z_{i\sqrt{l}+j} \coloneqq x_i y_j. \tag{8}$$

   Specifically, $\mathbf{TEST}_2$ is obtained by replacing the blue part of **TEST** in Figure 2 with the following:

$$\xleftarrow{\mathbf{z} \text{ defined by (8)}} \qquad \mathbf{x}, \mathbf{y} \leftarrow\!\$\, \mathbb{Z}_p^{\sqrt{l}}$$

   In the final protocol a PRG is used to generate these elements locally, and the Verifier simply sends the seed to the Prover.

2. $\mathbf{IPP}_{sub}$: The query vector $\mathbf{q}$ in $\mathbf{IPP}_{sub}$ is now defined by the single evaluation point $x \in \mathbb{Z}_p$. For an index k, $0 \leq k \leq l-1$, (assume l is a power of 2), let $k = (k_0, \ldots, k_{\log l - 1})$ denote the bit vector from base-2 representation of k. Then, define

$$q_k \coloneqq \prod_{0 \leq j \leq \log l - 1} (x^{k_j 2^j} \mod p). \tag{9}$$

   Note that q_k is an integer that is at most $p^{\log l}$.
   Specifically, $\mathbf{IPP}_{sub}$ is obtained by replacing the blue part of **IPP** in Figure 3 with the following:

$$\xleftarrow{\mathbf{q} \text{ defined by (9)}} \qquad x \leftarrow\!\$\, \mathbb{Z}_p$$

These changes are made mainly to decrease the verifier's computation complexity, more details are in Theorem 4.5.

[5] Here and below, we can use Kronecker product notation to express such vectors.

Polynomial Commitment Scheme sqPC. Our polynomial commitment scheme is essentially the vector commitment scheme: we interpret a polynomial $f(X) \in \mathbb{F}_p[X]$ of degree $d \leq D$ as a vector of coefficients of length $l := d+1$ and commit to the vector. Let $f(X) = \sum_{i=0}^{d} f_i X^i$. We denote by $\mathbf{f}$ the vector $(f_0, \ldots, f_d)$, and define $\mathsf{sqPC} = (\mathsf{setup}, \mathsf{commit}, \mathsf{open}, \mathsf{eval})$ as follows:

- $\mathsf{setup}(1^\kappa, D)$: $\mathsf{pp} \leftarrow \mathbf{Setup}(1^\kappa, D+1)$
- $\mathsf{commit}(\mathsf{pp}, f(X), d)$: $(C, \tilde{\mathbf{c}}) \leftarrow \mathbf{Com}(\mathsf{pp}, \mathbf{f}, d+1)$.
- $\mathsf{open}(\mathsf{pp}, f, d, C, \tilde{\mathbf{c}})$: $b \leftarrow \mathbf{Open}(\mathsf{pp}, \mathbf{f}, d+1, C, \tilde{\mathbf{c}})$
- $\mathsf{eval}(\mathsf{pp}, C, d, x, v; f(X))$: $b = \mathbf{Eval}(\mathsf{pp}, C, d+1, \mathbf{q}, v; \mathbf{f})$, where $\mathbf{q}$ is defined in 9 as a function of x and **Eval** is modified to use $\mathbf{TEST}_2$ and $\mathbf{IPP}_{sub}$ described above instead of **TEST** and **IPP**.

4.2 Proofs of Security and Succinctness for sqPC

In this section, we prove completeness, soundness, and succinctness of our polynomial commitment scheme sqPC. The most challenging is the proof of soundness, which we do by constructing extractors for $\mathbf{TEST}_2$ and **IPP** as before. However, the structure of the extracted (rational, in general) vector is lot more complex here than in the case of IPC since the query vectors have dependencies among their coordinates. We are able to exploit this structure while still keeping the size of the commitment and the number of class group elements communicated to be an absolute constant. We pay a price for the structural complexity with a larger α.

For this section, we choose $\alpha > c_1 l p^{c_2\sqrt{l}}$ for some absolute constants $c_1, c_2 > 1$ and $\alpha \equiv 0 \bmod p$.

For notational simplicity we also parametrise the definition of the set S to $S(\beta_1, \beta_2)$ and extend the domain of definition of χ_m, χ_n (defined before Theorem 3.3) to $S(\beta_1, \beta_2)$.

$$S(\beta_1, \beta_2) := \left\{ \frac{m\alpha - n}{k} : m, n, k \in \mathbb{Z},\ \gcd(m,k) = \gcd(k,p) = 1,\ 0 < m \leq k < p^{\beta_1}, -\beta_2 < n < k + \beta_2 \right\}$$

Theorem 4.1 (Completeness)**.** *The polynomial commitment scheme* sqPC *(Def 4.1) satisfies Completeness (Def 2.5) if $\alpha > l p^{1+logl}$.*

This proof is similar to the proof of completeness of IPC (Theorem 3.1) and is omitted.

Theorem 4.2 (Extractability)**.** *The polynomial commitment scheme* sqPC *satisfies Extractability (Def 2.7) for $(\beta_1, \beta_2) = (3, 5)$ in the Generic Group Model.*

Proof. We again split the proof into two theorems; the first theorem (concerning $\mathbf{TEST}_2$) constructs a partial extractor Ext_T and proves some conditions on the extracted objects and the second theorem (concerning $\mathbf{IPP}_{sub}$) using the results of the first theorem constructs the extractor Ext for eval.
More precisely, given a *generic* adversary $\mathcal{A}$ that makes the Verifier in eval (Def 4.1) accept with non-negligible probability, we will construct a polynomial time extractor Ext that outputs $(f(X), \tilde{\mathbf{c}})$ satisfying $\mathcal{R}_{\mathsf{eval}}$ (in Def 2.7) with overwhelming probability.

Theorem 4.3 ($\mathbf{TEST}_2$ Extractor)**.** *Suppose the Verifier in $\mathbf{TEST}_2$ instantiated with $\alpha > c_1 l p^{c_2\sqrt{l}}$ accepts with probability $> \frac{c_3}{\sqrt{p}}$ over uniformly random $(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}_p^{2\sqrt{l}}$, where $c_1, c_2, c_3 > 1$ are some fixed constants.*

Then, there exists an efficient extractor Ext_T in the GGM that outputs vectors $\mathbf{c}, \mathbf{d} \in \mathbb{Z}_p^l$ satisfying the following properties:

1. $C = g^{\sum_{i=0}^{l-1}(c_i + \alpha d_i)\alpha^{2i}}$ – *proved in Lemma 3.10*
2. $\mathbf{d} \in S(2,4)^l$ – *proved in Lemma 4.7*
3. $\mathrm{DenLCM}(\chi_m(\mathbf{d})) < p^{2\sqrt{l}}$ – *proved in Lemma 4.8*

Theorem 4.4 (**IPP**$_{sub}$ Extractor)**.** *If the Verifier in* **IPP**$_{sub}$ *outputs accept with non-negligible probability for some evaluation point x, and given that the Verifier of* **TEST**$_2$ *also did so, there exists an efficient extractor* Ext *in the GGM that outputs an opening $f(X) \in \mathbb{Z}_p[X]$ and an opening hint $\tilde{\mathbf{c}}$ in $\mathbb{Q}(3,5)^l$ for C such that $v = f(x) \mod p$.*

□

Theorem 4.5 (Proof Succinctness and $\sqrt{l}$ Verification)**.** *The polynomial commitment scheme* sqPC *satisfies* proof succinctness *as defined in Definition 2.8 and the Verifier in* eval *performs $O(1)$ group operations and $O(\sqrt{l})$ field operations (where $l-1$ is the degree of the polynomial).*

Proof. Proof succinctness is direct; the commitment is a single group element and the evaluation protocol only communicates a constant number of group elements. Using the optimisations in § 3.5, the *random* field elements x_i, y_j for $0 \leq i, j < \sqrt{l}$ that form the query vector can be sent succinctly by using a Pseudorandom Generator (PRG) with a short seed to generate the vector locally – this brings the communication back to constant.

Similarly, we can see that the verifier still only does a constant number of group operations (interacting with the Prover via PoEs for expensive exponentiations as argued in § 3.5) and the bottleneck for field operations is the computation of $\sigma \mod q'$ for a random prime q' during PoE verification.
Here, we can analyse **TEST**$_2$ and **IPP**$_{sub}$ separately. In **TEST**$_2$, the definition of the query vector implies that we can rewrite the computation of $\sigma \mod q'$ in the following way:

$$\sigma \mod q' = \sum_{k=0}^{l-1} \alpha^{2l-2-2j} z_j \mod q' = \alpha^{2l-2} \sum_{i=0}^{\sqrt{l}-1} \frac{x_i}{\alpha^{2i\sqrt{l}}} \sum_{j=0}^{\sqrt{l}-1} \frac{y_j}{\alpha^{2j}} \mod q'$$

This can be computed in $O(\sqrt{l})$ time. In **IPP**$_{sub}$, the query vector is formatted in such a way that the computation can be done in $O(\log l)$ time once $\alpha \mod q'$ is calculated.

$$\sigma \mod q' = \sum_{k=0}^{l-1} \alpha^{2l-1-2j} q_k \mod q' = \alpha^{2l-1} \prod_{i=0}^{\log l - 1} \left(1 + \frac{x^{2^i} \mod p}{\alpha^{2^{i+1}}}\right) \mod q'$$

Also note that for efficient computation, we need to compute $\alpha \mod q'$ and $\alpha^{-1} \mod q'$ (If $\alpha^{-1} \mod q'$ does not exist, then $\alpha = 0 \mod q'$ and computing these values becomes trivial). In this case, since $\alpha = p^{O(\sqrt{l})}$, these quantities can be computed in $O(\sqrt{l})$ as well.

Note: If faster computation of $\alpha \mod q'$ is desired, we can fix $\alpha = p^L$ (such that the bounds on α still hold) for some integer L, which implies that $\alpha \mod q' = p^L \mod q'$ can be computed efficiently - in $O(\log L)$ time. This technique is necessary when we consider the sublinear and logarithmic versions of our protocols (Theorem 5.8). □

4.3 Proofs for TEST$_2$ Extractor

In addition to the auxiliary lemmas from Section 3.3, we will make use of a combinatorial theorem in our proofs in this section.

4.3.1 An Auxiliary Theorem on Forbidden Boxes:

Consider the set of (standard) integer lattice of points inside $[n]^d$. How many points can we choose in this cube such that no subset of 2^d among them form the corners of a d-dimensional hyper-rectangle (box)? Upper bounds on the number of such points are used in the extractability proofs in this section and the next. A precise statement due to Rosenfeld [Ros16] of such a bound is stated below. In this case of $d = 2$, such a bound also follows from the well-known Zarankiewicz problem in extremal graph theory.

Theorem 4.6 ([Ros16] - Theorem 4.1, Restated)**.** *Consider a d-dimensional 0-1 matrix $M : [n]^d \to \{0,1\}$ such that*

$$\prod_{b_1=0}^{1} \cdots \prod_{b_d=0}^{1} M\left(i_1^{b_1}, \ldots i_d^{b_d}\right) = 0 \quad \forall\, 1 \le i_j^0 < i_j^1 \le n,\ 1 \le j \le d.$$

The number of ones in such a matrix M is less than $Cn^{d-2^{-d+1}}$ where $C = C_d$ is a constant that depends only on d.

Now we can prove theorems 4.3 and 4.4 on $\mathbf{TEST}_2$ and $\mathbf{IPP}_{sub}$ (proof of Extractabilty).

Lemma 4.7 (Part (2) of $\mathbf{TEST}_2$ extraction)**.** *If the prover succeeds with probability at least $\sqrt{2/p}$, then for all $0 \le k \le l-1$, $d_k \in S(2,4)$.*

Proof. Fix an arbitrary index k in $[0, l-1]$ and represent it by the pair (i,j), where $k = i\sqrt{l} + j$ – equivalently the representation in base-$\sqrt{l}$ of k is (i,j). Consider the partition of the space $\mathbb{Z}_p^{2\sqrt{l}}$ by sets of the form

$$T_{\mathbf{q}} \coloneqq \{(x_i, y_j, \mathbf{q}) : x_i, y_j \in \mathbb{Z}_p\} \text{ for } \mathbf{q} \in \mathbb{Z}_p^{2\sqrt{l}-2}.$$

Since the success probability of the prover is at least $\frac{\sqrt{2}}{\sqrt{p}}$, at least one of these sets, call it $T_{\mathbf{q_0}}$, (which are 2-dimensional spaces) must have more than $\sqrt{2}\cdot p\sqrt{p}$ accepting points. Lemma 4.6 then implies that there exists a rectangle of accepting points in this space, say $(x_i, y_j), (x_i + h, y_j), (x_i, y_j + t), (x_i + h, y_j + t)$ for $0 < h, t < p$; the rest of the coordinates are identical given by c_0. Call the query vectors in $\mathbb{Z}_p^l$ generated by these points according to Equation (8) as $\mathbf{z_1}, \mathbf{z_2}, \mathbf{z_3}$ and $\mathbf{z_4}$.

Now, Lemmas 3.6 and 3.7 give us equations corresponding to each of these accepting points $\mathbf{z_i}$, relating $\mathbf{c}, \mathbf{d}$ and $\mathbf{z_i}$, as below.

$$\langle \mathbf{d}, \mathbf{z_1} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_1} \rangle}{\alpha} \right\rfloor + u_1 = 0 \mod \alpha \tag{10}$$

$$\langle \mathbf{d}, \mathbf{z_2} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_2} \rangle}{\alpha} \right\rfloor + u_2 = 0 \mod \alpha \tag{11}$$

$$\langle \mathbf{d}, \mathbf{z_3} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_3} \rangle}{\alpha} \right\rfloor + u_3 = 0 \mod \alpha \tag{12}$$

$$\langle \mathbf{d}, \mathbf{z_4} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_4} \rangle}{\alpha} \right\rfloor + u_4 = 0 \mod \alpha \tag{13}$$

Consider the equation $(13-12)-(11-10)$. Due to the choice of $\mathbf{z_i}$, $i \in \{1,2,3,4\}$ we get

$$d_{i\sqrt{l}+j} \cdot ht = -\left[\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_4} \rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_3} \rangle}{\alpha} \right\rfloor + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_1} \rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_2} \rangle}{\alpha} \right\rfloor + (u_4 - u_3 + u_1 - u_2)\right] \mod \alpha \tag{14}$$

Call the term in the brackets on the RHS as n. Since for all x, $x - 1 \le \lfloor x \rfloor < x$ and $u \in \{0,1\}$, we can see that $-2 < n < ht + 4$. Now we can write

$$d_{i\sqrt{l}+j} = \frac{m\alpha - n}{ht}$$

where $m \le ht < p^2$ (since $d_{i\sqrt{l}+j} < \alpha$ and α is large) and $-2 < n < ht + 4$. Hence, $d_{i\sqrt{l}+j} \in S(2,4)$. Since the foregoing was proved for arbitrary $k = (i,j)$, we must have $\mathbf{d} \in S(2,4)^l$. □

Lemma 4.8 (Part (3) of $\mathbf{TEST}_2$ extraction)**.** *If the prover accepts with probability $\ge 3\sqrt{l}/p$, we must have* $\mathrm{DenLCM}(\chi_m(\mathbf{d})) \le p^{2\sqrt{l}}$.

Proof. Suppose this was not the case, and $\mathrm{DenLCM}(\chi_m(\mathbf{d})) > p^{2\sqrt{l}}$. Divide the l coordinates $\{d_{i\sqrt{l}+j}\}_{i,j}$ into $\sqrt{l}$ contiguous blocks of size $\sqrt{l}$, where each block corresponds to a single index i.

Since each denominator (in the representation given by $S(2,4)$) is smaller than p^2 and there are $\sqrt{l}$ elements in every block, there must exist at least one block such that the DenLCM of the set $\{\chi_m(\mathbf{d_{i\sqrt{l}+j}})\}_j$ the block is between p^2 and $p^{2\sqrt{l}}$. WLOG, let this be the first block (corresponding to the index 0).

Now, consider a partition of the space $\mathbb{Z}_p^{2\sqrt{l}}$ by sets of the form

$$T_{\mathbf{q}} := \{(x_0, \mathbf{q}) : x_0 \in \mathbb{Z}_p\}$$

for every $\mathbf{q} \in \mathbb{Z}_p^{2\sqrt{l}-1}$. Now, we can bound the number of c such that $T_{\mathbf{q}}$ has more than two accepting points. Denote the number of accepting points in $T_{\mathbf{q}}$ by $||T_{\mathbf{q}}||$.
Consider any set $T_{\mathbf{q}}$ such that $||T_{\mathbf{q}}|| \geq 2$. Then, using Lemmas 3.6 and 3.7 we get two equations for $\mathbf{z} \neq \mathbf{z}'$ (WLOG let $x_0 > x_0'$):

$$\langle \mathbf{d}, \mathbf{z} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z} \rangle}{\alpha} \right\rfloor + u = 0 \mod \alpha \tag{15}$$

$$\langle \mathbf{d}, \mathbf{z}' \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}' \rangle}{\alpha} \right\rfloor + u' = 0 \mod \alpha \tag{16}$$

$$\implies (x_0 - x_0') \sum_{i=0}^{\sqrt{l}-1} d_i y_i = -\left[\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z} \rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}' \rangle}{\alpha} \right\rfloor + (u - u') \right] \mod \alpha \tag{17}$$

Call the quantity in the square brackets on the RHS as N'. Then, we can use the characterisation of $\mathbf{d} \in S(2,4)^l$ (each $d_i = \frac{m_i \alpha - n_i}{k_i}$) to write Equation (17) as:

$$M\alpha - N = M'\alpha - N'$$

$$M := \sum_{i=0}^{\sqrt{l}-1} \frac{m_i}{k_i}(x_0 - x_0')y_i, \quad N := \sum_{i=0}^{\sqrt{l}-1} \frac{n_i}{k_i}(x_0 - x_0')y_i$$

where $|N|, |N'| < 2p^2\sqrt{l}$ and M has maximum denominator $p^{2\sqrt{l}}$. Using Lemma 3.8 and since $\alpha > 8lp^{2\sqrt{l}+2}$, we get that $M = M'$ and $N = N'$. More importantly, this implies that $M \in \mathbb{Z}$. This implies that

$$(x_0 - x_0') \sum_{i=0}^{\sqrt{l}-1} \frac{m_i}{k_i} y_i \in \mathbb{Z} \tag{18}$$

$$\implies (x_0 - x_0') \sum_{i=0}^{\sqrt{l}-1} \frac{m_i L}{k_i} y_i = 0 \mod L \tag{19}$$

$$\implies \sum_{i=0}^{\sqrt{l}-1} \frac{m_i L}{k_i} y_i = 0 \mod L', \tag{20}$$

where $L = LCM(\{k_i\}_i) > p^2$, $L' \mid L$ and $L' > p$. Since for all i, $GCD(m_i, k_i) = 1$, the GCD of all the coefficients of the polynomial on the LHS with L' is 1.

From Lemma 3.9, the probability that this event happens is at most $\frac{\sqrt{l}}{p} + \frac{1}{L'} \leq \frac{2\sqrt{l}}{p}$ since $L' > p$. In other words, the fraction of solutions $\{y_i\}_i$ to (20) is at most $2\sqrt{l}/p$. However, notice that each set $T_{\mathbf{q}}$ corresponds to only one choice of $\mathbf{y}$. Furthermore, the same number ($p^{\sqrt{l}-1}$, to be precise) of $T_{\mathbf{q}}$ would arise from a given y. Hence, the fraction of sets $T_{\mathbf{q}}$ which contain more than 1 accepting value of $\mathbf{z}$ at most $2\sqrt{l}/p$. Since the size of each $T_{\mathbf{q}}$ is exactly p, the probability of acceptance by the prover is at most $(1 - 2\sqrt{l}/p) \cdot (1/p) + 2\sqrt{l}/p \cdot 1 < 3\sqrt{l}/p$. This contradicts the hypothesis that the prover accepts with probability $\geq 3\sqrt{l}/p$. It follows that we must have $\text{DenLCM}(\chi_m(\mathbf{d})) \leq p^{2\sqrt{l}}$. □

4.4 Proof of Theorem 4.4 – $\mathbf{IPP}_{sub}$ extractor

We define the extractor Ext for eval (of sqPC defined in Section 4.1) using the extractor Ext_T of $\mathbf{TEST}_2$. Since the $\mathbf{TEST}_2$ verifier outputs accept with non-negligible probability, there exists Ext_T that outputs vectors $\mathbf{c}, \mathbf{d}$ such that they satisfy the conditions in Theorem 4.3. Ext invokes Ext_T and performs the following computations on $\mathbf{c}, \mathbf{d}$:

1. Compute m_i, n_i, k_i for every i such that $d_i = \frac{m_i\alpha - n_i}{k_i}$.
2. Define vectors $\mathbf{c}', \mathbf{d}' \in \mathbb{Z}_p^l$ as

$$c'_i \coloneqq c_i + \frac{m_{i-1}}{k_{i-1}}$$
$$d'_i \coloneqq -\frac{n_i}{k_i}$$

   $(m_{-1} \coloneqq 0, k_{-1} \coloneqq 1)$
3. Output $\mathbf{c}' + \alpha\mathbf{d}'$ as the opening hint, and define a polynomial $f(X) \in \mathbb{Z}_p[X]$ using the coefficient vector $(\mathbf{c}' + \alpha\mathbf{d}') \mod p$ as the opening to the commitment.

Note that this extractor is indeed efficient, details are given in Lemma 4.9.

Lemma 4.9. *The extractor* Ext *defined in Theorem 4.4 is efficient.*

Proof. The extractor calls Ext_T, which is efficient since the only computation done in Ext_T is a constant number of calls to the extractor of PoKPE, which is efficient and terminates in polynomial time [BBF19, Theorem 7] when the adversary is assumed to be generic. In Ext above, the only non-trivial computations done are in Step 1 above.

This can be done efficiently using a binary search algorithm on rationals with bounded denominator, since we already know that there exists a representation for d_i of the form $\frac{m_i\alpha - n_i}{k_i}$ with $k_i < p^2$ from Theorem 4.3. For every i,

1. Compute $q := \frac{d_i}{\alpha}$.
2. Find (m_i, k_i) as the rational representation equal to q (within a distance of $\frac{2}{\alpha}$ - since this is negligible compared to the rationals we're searching for, we can use existing algorithms for exact rational search to obtain (m_i, k_i)) with denominator at most p^2. (This can be done in $\Theta(2\log p)$ [KM03, Theorem 1])
3. Compute $n_i := m_i\alpha - kd_i$.
4. Output (m_i, n_i, k_i).

□

By construction, this is a valid opening hint, as $\mathbf{d} \in S(2,4)^l \implies \mathbf{c}' + \alpha\mathbf{d}' \in \mathbb{Q}(3,5)$ and since it is just a rearrangement of the coordinates of $\mathbf{c}$ and $\mathbf{d}$,

$$C = g^{\sum_{i=0}^{l-1}(c'_i + \alpha d'_i)\alpha^{2i}}$$

which is one of the required conditions.

Now, since the verifier accepts in $\mathbf{IPP}_{sub}$, Lemmas 3.6 and 3.7 along with the final check of the verifier (Figure 3, Check 3) gives us equations corresponding to the coefficients of α^{2l-1} and α^{2l-2}:

$$\langle \mathbf{d}, \mathbf{q}^+ \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u' = 0 \mod \alpha \tag{21}$$

$$\langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u = v + np \mod \alpha \tag{22}$$

where $u, u' \in \{0, 1\}$, and the query vector $\mathbf{q}$ is as defined (as a function of the evaluation point x) in Equation (24).

Due to the bounds on coefficients of $\mathbf{q}$, we know that Equation 21's LHS must be either 0 or α. Also define

$$M' \coloneqq \sum_{i=0}^{l-1} \frac{m_i}{k_i} q_i^+ \,,\; N' \coloneqq \sum_{i=0}^{l-1} \frac{n_i}{k_i} q_i^+$$

1. If the LHS is 0, then so are each of the terms in the LHS, as they are all non-negative. Hence, $\langle \mathbf{d}, \mathbf{q}^+ \rangle = 0 \mod \alpha$, and by definition, $u = 0$.
   Hence, we can simplify Equation 22

$$\begin{aligned} v + np &= \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u \mod \alpha \\ &= \langle \mathbf{c}, \mathbf{q} \rangle + \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \mod \alpha \\ &= \langle \mathbf{c}, \mathbf{q} \rangle + M' - \frac{N'}{\alpha} \mod \alpha \end{aligned}$$

   $M' - N'/\alpha$ is an integer and since $N' < \alpha$ by choice of α and the query vector $\mathbf{q} \implies N' = 0$.

$$\begin{aligned} v + np &= \langle \mathbf{c}, \mathbf{q} \rangle + M' \mod \alpha \\ &= \langle \mathbf{c}', \mathbf{q} \rangle \mod \alpha \\ \implies v &= \langle \mathbf{c}', \mathbf{q} \rangle \mod p \\ &= \langle \mathbf{c}', \mathbf{q} \rangle + \alpha \langle \mathbf{d}', \mathbf{q} \rangle \mod p \end{aligned}$$

   as $\alpha = 0 \mod p$ and $\langle \mathbf{d}', \mathbf{q} \rangle$ is invertible modulo p (or simply $0 \mod p$). In either case,

$$v = \langle \mathbf{c}' + \alpha \mathbf{d}', \mathbf{q} \rangle \mod p$$

2. If the LHS is α, we get that $u = 1$ by definition. From the properties of $\mathbf{c}$ and $\mathbf{d}$, we know that $M' \in \mathbb{Q}$ has a denominator of at most $p^{2\sqrt{l}}$ (in the reduced form). Write the equation 21 in the form

$$\langle \mathbf{d}, \mathbf{q}^+ \rangle = M\alpha - N$$

   by moving all the other (integer) terms to the RHS and calling it N. Since $M, N \in \mathbb{Z}$ Lemma 3.8 implies that $M', N' \in \mathbb{Z}$ (since $\alpha > 8lp^{2\sqrt{l}+1+\log l}$).
   Now,

$$\begin{aligned} v + np &= \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u \mod \alpha \\ &= \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor M' - \frac{N'}{\alpha} \right\rfloor + 1 \mod \alpha \\ &= \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + M' - 1 + 1 \end{aligned}$$

   as $N' < \alpha$. Hence, as in the previous step, we get that

$$v = \langle \mathbf{c}' + \alpha \mathbf{d}', \mathbf{q} \rangle \mod p$$

Our query vector $\mathbf{q}$ was chosen specifically to satisfy the property $q_k \mod p = x^k \mod p$ for the evaluation point $x \in \mathbb{Z}_p$. Since the vector $\mathbf{c}' + \alpha \mathbf{d}' \mod p$ is the coefficient vector of the polynomial $f(X)$ that the extractor outputs, $y = f(x) \mod p$.

4.5 Improving verification to $O(l^\epsilon)$

We can extend the techniques in this section to construct polynomial commitment schemes with better verification complexity $O(l^\epsilon)$ for constant $0 < \epsilon < 1$. This extension uses the Box theorem (Theorem 4.6) in its full generality for dimensions greater than 2. We elaborate on this in Appendix B.1.

The construction of the scheme is very similar to sqPC, and the only change is the way in which the **TEST** query vector is generated – we now use Kronecker products of e many $l^{1/e}$-length vectors ($el^{1/e}$ random elements) to construct the query vector $\mathbf{z}$. When $e = 2$, this is the same process as sqPC. This would then give us verifier complexity of $O(l^{1/e})$ as the structure of the query vector simplifies the computation of σ.
The proof of extraction proceeds almost identically to that of sqPC in Theorem 4.2, with the difference being that we now argue the existence of a e-dimensional box in an appropriate space instead of a rectangle, which gives rise to 2^e equations.
This can be represented as a binary tree of depth e, and to obtain the necessary conditions on the $\mathbf{d}$ vector we recursively subtract the 2^e equations at the leaves. In addition, part 3 of Theorem 4.2 does not generalise to higher dimensions, hence we use a weaker condition which can be easily satisfied at the cost of making $\alpha = p^{O(el)}$. This does not affect verifier efficiency though, as α is an exact power of p.

However, due to the bound in the Box theorem becoming weaker as the dimension increases, this impacts the Prover's cheating probability – as the dimension increases, so does the probability that the Prover breaks soundness (if the dimension is e, the cheating probability is $< \frac{C_e}{p^{2-e+1}}$). Hence, this construction can only be used as long as the cheating probability is negligible, which is satisfied for *constant* dimension e – specifically not when $e = \log l$, and hence we need different techniques to obtain logarithmic verification.

We give the proof of extraction in Appendix B.1. The proof is a direct generalisation of Theorem 4.2 using the Box theorem in its full generality.

5 Dew-PC – Constant-Sized PCS with Logarithmic Verifier

We prove our main result on PCS in this section. Recall that an IPC implies a PCS. Hence, our starting point is IPC from §3. The challenge then is to reduce the linear verification time of that scheme to logarithmic for the derived PCS, while retaining constant size. Our basic idea to achieve log-verification is to use *Kronecker products* of log-many length-2 vectors for $\mathbf{z}$ in **TEST** and $\mathbf{q}$ in **IPP** protocols (cf. Fig. 2 and 3). These modifications do help us reduce the verification time to logarithmic (Theorem 5.8 below). However, with this new choice for $\mathbf{z}$, the extractability proof from §3 completely breaks down. Proving extractability with Kronecker product $\mathbf{z}$ requires substantially new ideas. A crucial tool in our proof is a *new extremal combinatorial bound*: we prove a tight bound on the maximum number of points in the discrete cube $[n]^d$ that do not contain a subset called a d-structure (§ 5.2.2). A d-structure is a generalization of a d-dimensional hyper-rectangle (box). In the special case of rectangles, extremal combinatorial bounds are in fact known in the literature. For $d = 2$, this is the well-known Zarankiewicz problem [Bol04] and for constant $d > 2$, a generalization was proved by Rosenfeld [Ros16].

Indeed, combining these earlier results with the Kronecker products idea, we can already get *sublinear* verification time: $\sqrt{l}$ using [Bol04] and l^ϵ for any constant $\epsilon > 0$ using [Ros16]; see details in §4 and Appendix B. Unfortunately, these bounds behave like $n^{d-2^{-d+1}}$ and hence rapidly approach n^d as d grows. At a very high level, that enables a cheating prover to succeed with non-negligible probability on Kronecker products of a super-constant number of vectors. When we choose a set of points from a universe, we say a *forbidden configuration* is a subset of points forming a structure that is not allowed to be contained in the chosen set. By generalizing forbidden configurations from rectangles to d-structures (which suffices for us), we obtain upper bounds of $O(dn^{d-1})$ – a vanishingly small fraction of n^d. This improved bound enables a verifier to catch a cheating prover with high probability.

5.1 Our Polynomial Commitment Scheme : Dew-PC

To construct Dew-PC, we use ideas based on Kronecker products as test and query vectors to modify the **TEST** and **IPP** protocols from §3 to **logTEST** and **logIPP** as defined below.

For notational simplicity, let the degree d of the polynomial be equal to $l - 1$.

For **logTEST**, the query vector $\mathbf{z}$ in Fig. 2 is defined using $2 \log l$ random elements of $\mathbb{Z}_p$:

1. Sample random $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\log l}$ from $\mathbb{Z}_p^2$ where $\mathbf{x_j} = (x_{j,0}, x_{j,1})$.
2. For $0 \le k \le l-1$, let $(k_0, \ldots, k_{\log l - 1})$ be the base-2 representation of k so that $k = k_0 \cdot 2^0 + \cdots + k_{\log l - 1} \cdot 2^{\log l - 1}$. Then,

$$z_k \equiv z_{k_0,\ldots,k_{\log l-1}} \coloneqq \prod_{j=1}^{\log l} x_{j,k_{j-1}}. \tag{23}$$

Also, for **logIPP**, the query vector $\mathbf{q}$ in Fig. 3 is defined by the single evaluation point $x \in \mathbb{Z}_p$ by setting

$$q_k \coloneqq \prod_{0 \le j \le \log l - 1} (x^{k_j 2^j} \mod p). \tag{24}$$

Note that z_k, q_k are integers that are $< p^{\log l}$ and $q_k \bmod p = x^k \bmod p$.

Our PCS $\mathsf{Dew\text{-}PC} = (\mathsf{setup}, \mathsf{commit}, \mathsf{open}, \mathsf{eval})$ is now constructed as follows:

- $\mathsf{setup}(1^\kappa, D)$: Here, κ is the security parameter and D is an upper bound on the degree of the committed polynomial. Sample a group of unknown order (we use class groups) $\mathbb{G} \leftarrow \mathsf{GGen}(\kappa)$ and a random $g \leftarrow\$ \mathbb{G}$. $PRG : \{0,1\}^\lambda \to \mathbb{Z}_p^{2\log(D+1)}$ is a pseudorandom generator. Define $\alpha \coloneqq p^{5\log(D+1)}$ (p is a large prime such that $\mathsf{len}(p) = \mathsf{poly}(\kappa)$).
  `Return` $\mathsf{pp} = (\kappa, \mathbb{G}, g, PRG, p)$.
- $\mathsf{commit}(\mathsf{pp}, D, f(X) \in \mathbb{Z}_p[X], l-1)$: Define the commitment $C \coloneqq g^{\sum_{i=0}^{l-1} f_i \alpha^{2i}}$, where f_i are the coefficients of the degree $(l-1)$ polynomial $f(X)$ considered as integers from $[0, p-1]$ and the sum is also over the integers. If $l - 1 \le D$, return $(C, \mathbf{f})$, return error otherwise.
- $\mathsf{open}(\mathsf{pp}, D, f(X) \in \mathbb{Z}_p[X], l-1, C, \tilde{\mathbf{f}})$: Check that
  (i) $l - 1 \le D$,
  (ii) $\sum_{i=0}^{l-1} \tilde{f}_i \alpha^{2i} \in \mathbb{Z}$, $C = g^{\sum_{i=0}^{l-1} \tilde{f}_i \alpha^{2i}}$ and $\tilde{\mathbf{f}} \in \mathbb{Q}(1 + \log l, 2l+1)^l$.

  Recall that

$$\mathbb{Q}(\beta_1, \beta_2) \coloneqq \left\{ \frac{a}{b} : \ \gcd(a,b) = 1,\ \gcd(b,p) = 1,\ 0 < b < p^{\beta_1},\ |a/b| \le \beta_2 \alpha \right\},$$

  where $|a|$ denotes the absolute value of the integer a.
  (iii) $\tilde{f}_i = f_i \mod p$ where $f_i \in \mathbb{Z}_p$ are the coefficients of $f(X)$.

  `return` **1** if all checks (i)-(iii) above pass, else `return` **0**.
- $\mathsf{eval}(\mathsf{pp}, D, C, l-1, x, v; f(X))$: The eval protocol consists of two sub-protocols **logTEST** and **logIPP** as described in Figures 5 and 6 and
  - `If` $l - 1 > D$, `return` **0**
  - `Else` Run **logTEST**$(C, l-1; f(X))$ and **logIPP**$(C, l-1, x, v; f(X))$.
    `return` **1** if both these protocols accept `else return` **0**.

The protocols **logTEST** and **logIPP** described in Figures 5 and 6 are simply variants of **TEST** and **IPP** in Figures 2 and 3 by replacing the query vectors with kronecker products of shorter vectors as in (23) and (24).

As written in Figures 5 and 6, the **logTEST** and **logIPP** protocols do not reach the claimed bounds on communication and computation complexity. To obtain these, the Verifier sends a seed for the PRG instead of the random query vector in **logTEST** and replace all expensive group exponentiations for the verifier by invocations of Wesolowski's PoE protocol. The full protocols in Figures 7 and 8 are expanded versions of the protocols in Figures 5 and 6 thus obtained.

$\mathsf{CoeffSplit}(\alpha, a, b, i)$

1. Write $a \cdot b$ in base α, call the resulting representation vector $\mathbf{c}$.
2. Set $v \coloneqq c_i$, $\gamma \coloneqq \sum_{j=0}^{i-1} c_j \alpha^j$, $\lambda \coloneqq \sum_{j=i+1}^{\lceil \log_\alpha ab \rceil} c_j \alpha^j$
3. Output the tuple (v, γ, λ).

Figure 4: CoeffSplit

logTEST

Prover		**Verifier**
$C \coloneqq \mathsf{commit}(\mathsf{pp}, D, f(X) \in \mathbb{Z}_p[X], d)$	$\xrightarrow{C}$	
	$\xleftarrow{\mathbf{z} \text{ defined by } (23)}$	$\mathbf{x_1}, \ldots, \mathbf{x}_{\log l} \leftarrow\$ \mathbb{Z}_p^2$
Computations in **TEST** / (Λ, Γ)	$\xrightarrow{(\Lambda, \Gamma)}$	Checks
		$\sigma \coloneqq \sum_{j=0}^{l-1} \alpha^{2l-2-2j} z_j$
		$E \coloneqq \frac{g^{\alpha^{2l-1}}}{C}, \Delta \coloneqq \frac{g^{\alpha^{2l-1}}}{\Gamma}$
		1 : $\mathsf{PoKPE}(C)$ accepts
		2 : $\mathsf{PoKPE}(E)$ accepts
		3 : $\mathsf{PoKPE}(\Lambda)$ accepts
		4 : $\mathsf{PoKPE}(\Gamma)$ accepts
		5 : $\mathsf{PoKPE}(\Delta)$ accepts
		6 : $C^\sigma \stackrel{?}{=} \Lambda^{\alpha^{2l}} \Gamma$

The blue colored parts are the changes from the **TEST** protocol in Figure 2. "Computations in **TEST**" are steps $(7, 8, 9)$ in Figure 7 (same as in Figure 2).

Figure 5: The **logTEST** Protocol

logIPP

Prover		**Verifier**
	$\xleftarrow{\mathbf{q} \text{ defined by } (24)}$	x is the evaluation point
Computations in **logIPP** / (v, n, Λ, Γ), $N \coloneqq g^n$	$\xrightarrow{(v, N, \Lambda, \Gamma)}$	Checks
		$\sigma \coloneqq \sum_{j=0}^{l-1} \alpha^{2l-1-2j} q_j$
		$\Delta \coloneqq \frac{g^{\alpha^{2l-2}}}{\Gamma}$
		1 : $v \in \mathbb{Z}_p$
		2 : $\mathsf{PoKPE}(\Lambda)$ accepts
		3 : $\mathsf{PoKPE}(\Gamma)$ accepts
		4 : $\mathsf{PoKPE}(\Delta)$ accepts
		5 : $\mathsf{PoKPE}(N)$ accepts
		6 : $C^\sigma \stackrel{?}{=} (g^v N^p)^{\alpha^{2l-1}} \Lambda^{\alpha^{2l}} \Gamma$

The blue colored parts are the changes from the **IPP** protocol in Figure 3. "Computations in **logIPP**" are steps $(3, 4, 5)$ in Figure 8 (same as in Figure 3).

Figure 6: The **logIPP** Protocol

logTEST

1 : $\mathcal{P}$ computes $C \coloneqq \mathsf{commit}(\mathsf{pp}, D, f(X) \in \mathbb{Z}_p[X], l-1),\ A \coloneqq g^{\alpha^{2l-1}}$ and sends C, A to $\mathcal{V}$

2 : $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PoE}(g, A, \alpha^{2l-1})$ // Showing that $A = g^{\alpha^{2l-1}}$

3 : $\mathcal{V}$ samples a random seed s as input to the PRG and sends s to $\mathcal{P}$.

4 : $\mathcal{P}$ and $\mathcal{V}$ compute $PRG(s) \to (z_{1,0}, \ldots, z_{\log l,0}, z_{1,1}, \ldots, z_{\log l,1}) \in \mathbb{Z}_p^{2\log l}$.

5 : $\mathcal{P}$ and $\mathcal{V}$ compute $\mathbf{z}$ as defined in equation 23.

6 : $\mathcal{P}$ computes $(v, \gamma, \lambda) \leftarrow \mathsf{CoeffSplit}\left(\alpha, \sum_{j=0}^{l-1} f_j \alpha^{2j}, \sum_{j=0}^{l-1} \alpha^{2l-2-2j} z_j, 2l-1\right)$ (defined in Figure 4)

7 : $\mathcal{P}$ computes $\Lambda \coloneqq g^{\lambda},\ \Gamma \coloneqq g^{\gamma}$ and sends (Λ, Γ) to $\mathcal{V}$

8 : $\mathcal{P}$ computes $\sigma \coloneqq \sum_{j=0}^{l-1} \alpha^{2l-2-2j} z_j$

9 : $\mathcal{V}$ computes $E \coloneqq AC^{-1},\ \Delta \coloneqq A\Gamma^{-1}$

10 : $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PoKPE}(C), \mathsf{PoKPE}(E), \mathsf{PoKPE}(\Lambda), \mathsf{PoKPE}(\Gamma), \mathsf{PoKPE}(\Delta)$

11 : $\mathcal{P}$ computes $R \coloneqq \Lambda^{\alpha^{2l}}\Gamma$ and sends R to $\mathcal{V}$

12 : $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PoE}(\Lambda, R/\Gamma, \alpha^{2l})$ // Showing that $R = \Lambda^{\alpha^{2l}}\Gamma$

13 : $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PoE}(C, R, \sigma)$ // Showing that $C^{\sigma} = R$

Figure 7: Full **logTEST** protocol - obtained from Figure 5 by replacing expensive verifier exponentiations by PoE protocols and using a PRG for succinctness

logIPP

1 : $\mathcal{P}$ sends $B \coloneqq g^{\alpha^{2l-2}}$ to $\mathcal{V}$

2 : $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PoE}(g, B, \alpha^{2l-2})$ // Showing that $B = g^{\alpha^{2l-2}}$

3 : $\mathcal{V}$ sends the evaluation point $x \in \mathbb{Z}_p$ to $\mathcal{P}$.

4 : $\mathcal{P}$ and $\mathcal{V}$ compute $\mathbf{q}$ as defined in equation (24).

5 : $\mathcal{P}$ computes $(v, \gamma, \lambda) \leftarrow \mathsf{CoeffSplit}\left(\alpha, \sum_{j=0}^{l-1} f_j \alpha^{2j}, \sum_{j=0}^{l-1} \alpha^{2l-1-2j} q_j, 2l-1\right)$ (defined in Figure 4)

6 : $\mathcal{P}$ computes $\Lambda \coloneqq g^{\lambda},\ \Gamma \coloneqq g^{\gamma},\ N \coloneqq g^{n}$ and sends $(v \mod p, N, \Lambda, \Gamma)$ to $\mathcal{V}$

7 : $\mathcal{P}$ computes $\sigma \coloneqq \sum_{j=0}^{l-1} \alpha^{2l-1-2j} q_j$

8 : $\mathcal{V}$ computes $\Delta \coloneqq B\Gamma^{-1}$

9 : $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PoKPE}(N), \mathsf{PoKPE}(\Lambda), \mathsf{PoKPE}(\Gamma), \mathsf{PoKPE}(\Delta)$

10 : $\mathcal{P}$ computes $R \coloneqq \Lambda^{\alpha^{2l}}\Gamma,\ S \coloneqq C^{\sigma}$ and sends R, S to $\mathcal{V}$

11 : $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PoE}(\Lambda, R/\Gamma, \alpha^{2l})$ // Showing that $R = \Lambda^{\alpha^{2l}}\Gamma$

12 : $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PoE}(C, S, \sigma)$ // Showing that $C^{\sigma} = S$

13 : $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PoE}(g^v N^p, S/R, \alpha^{2l-1})$ // Showing that $C^{\sigma} = (g^v N^p)^{\alpha^{2l-1}}\Lambda^{\alpha^{2l}}\Gamma$

Figure 8: Full **logIPP** protocol - obtained from Figure 6 by replacing verifier expensive exponentiations by PoE protocols and using a PRG for succinctness

5.1.1 Non-interactive Dew-PC using Fiat-Shamir

Note that even though **logTEST** and **logIPP** are described as separate protocols for ease of exposition, they are both run as part of eval. Protocol eval is public-coin and we can use the Fiat-Shamir heuristic [FS87, CCH+19] to obtain a non-interactive version in the ROM. First, the prover applies the RO on the commitment C to obtain a seed s for the PRG in **logTEST**. Then, s is expanded by the PRG to obtain $\mathbf{x}_1, \dots \mathbf{x}_{\log l}$, and the random query vector $\mathbf{z}$ is computed as described in Eq (23).
The non-interactive (NI) transcript consists of all the elements communicated in both protocols along with the NI versions of PoE and PoKPE from both protocols. Hence, the transcript communicated is $\pi = ((C, A, \Lambda, \Gamma, R)_{\mathbf{logTEST}}, (B, N, \Lambda, \Gamma, R, S)_{\mathbf{logIPP}}, \pi_{\mathsf{PoE}}, \pi_{\mathsf{PoKPE}})$ where π_{PoE} consists of NI transcripts of steps $(4, 20, 21)$ and $(2, 16, 17, 18)$ in **logTEST** and **logIPP** respectively, and π_{PoKPE} consists of NI transcripts of steps $(13, 14, 15, 16, 17)$ and $(10, 11, 12, 13)$ in **logTEST** and **logIPP** respectively (from Fig. 7 and 8).
It is easy to see that if an interactive protocol is succinct w.r.t. communication and verifier computation, its non-interactive Fiat-Shamir transformed version will be succinct according to Definition 2.8.

We prove completeness, extractability, and succinctness of Dew-PC in §5.2. We also show how to achieve hiding commitment and zero-knowledge evaluation in §7.1.

5.2 Proofs of Extractability and Succinctness of Dew-PC

We now prove the properties of our polynomial commitment scheme Dew-PC constructed in Section 5.1. Proof of completeness is analogous (taking care of the new choices of parameters) to the **IPP** in Theorem 3.1 and is deferred to Appendix C. Since the commitment scheme remains unchanged, the proof of binding remains as in Theorem 3.2. Proof of succinctness is given in Section 5.2.4. Section 5.2.5 contains concrete estimates of proof sizes.

Extractability of Dew-PC is the most nontrivial of the three properties to prove and hence we begin with a sketch of the proof below. A detailed proof appears in Section 5.2.3.

5.2.1 Proof sketch of Extractability for Dew-PC

The high level structure of the extractability proof is similar to that of the linear verifier protocol IPC in Theorem 3.3. The proof is structured into two parts, one concerning **logTEST** and the other for **logIPP** which uses the extractor from the **logTEST** theorem to construct the final extractor.
The **logTEST** theorem consists of three parts: (1) extracting the exponent of the commitment using the PoKPE extractor and parsing it as a vector of coefficients, (2) proving certain constraints on the coefficients of the above vector and finally, (3) a global property of the coefficients that uses ideas from (2) and a new result on extending the Schwartz-Zippel lemma for multilinear polynomials from [BF22].
Part (1) (extraction) is identical across both protocols, we can call the PoKPE extractor to output the exponent of the commitment C and write it in base-α to get a vector of coefficients. Part (2) concerns conditions on (a subset of) the extracted vector. Similar to the proof of part 2 (Lemma 3.11) of the **TEST** theorem, we aim to show that the coefficients d_i in the above vector with odd indices are such that $\frac{d_i}{\alpha}$ is a rational with "small" ($\ll \alpha$) denominators. The statement differs from the one in Theorem 3.11 only in the "small"-ness desired as well as the size of α. However, the proof of this part is more involved: Due to the structured query vector in **logTEST** (as opposed to the truly random query vector in **TEST**), we save on computational complexity of the verifier, but we cannot use the same arguments in this proof. Similar to equations (3), (4) in the proof of Lemma 3.11, we obtain equations relating the coefficients of the vector to the query vector whenever the prover succeeds in convincing the verifier. Since the inner product term in those equations are multilinear in the random variables (by choice of the query vector), there are structured sets of l equations which can be folded by recursively subtracting the equations to cancel out all terms except the one containing the coefficient of interest similar to Eq. (5). This structured set is called a $\log l$-structure; definitions and folding of equations are elaborated in §5.2.2.

Informally, d-structures are generalisations of parallelograms (in 2 dimensions) to d dimensions. For instance, a 2-structure is a parallelogram, while a 3-structure can be seen as two parallel parallelograms with the same base length and height (note that this is more general than a parallelepiped - in a

parallelepiped, the two parallelograms have to be congruent). For higher dimensions, the construction is recursive.

Fig. 10 shows two examples of 3-structures. To show that a $\log l$-structure actually exists in the desired space, we use a counting argument detailed in Theorem 5.2 that gives a lower bound on the number of d-structures in a d-dimensional hypercube. This theorem is similar to the Box theorem (Theorem 4.6), but these structures are more general than boxes and hence leads to significantly better (and tighter) bounds.
The theorem states that in an $[n]^d$ integer lattice, we can choose at most $(n^d - (n-1)^d)$ points in the matrix such that there exists no d-structure. In the extractability proof, we can argue that the prover succeeds with non-negligible probability and hence must populate an appropriately chosen lattice with more points than this lower bound, leading to the existence of a $\log l$-structure.

This process is actually also done in the **TEST** proof, but only requires two equations (corresponding to a 2-structure in the above framework), and the random points can be chosen such that they differ in only one coordinate. Subtracting these two equations then gives us the required term. In the **logTEST** proof, we instead have to fold l equations recursively two at a time.

Part (3) shows a global property of the vector $\mathbf{d}$. From part (2), we know that each coefficient d_i is such that d_i/α is close to a rational with "small" denominators, and this part shows a non-trivial bound B on the LCM of all the denominators. The proof is by contradiction – assuming that the LCM of the vector of denominators is larger than B, we first show that there exists a sub-vector of denominators for which the LCM is between B and B^2. Now, we can partition the randomness space, and use the fact that the prover succeeds with non-negligible probability to argue the existence of τ-structures (for $\tau < \log l$). The integer τ depends on the size of the sub-vector obtained in the first step. Now, similar to the techniques in part (2), we can recursively subtract (fold) the equations given by the points of the τ-structure. However, since $\tau < \log l$, we are left with multiple monomials and not a single coordinate like in part (2). We can now use the bounds on the LCM of the sub-vector and a Schwartz-Zippel analogue for multilinear polynomials from [BF22] to show that the prover succeeds with atmost negligible probability, which is a contradiction.

5.2.2 Counting structures

Our proofs of extractability for constant size PCS with log-time verification rely on certain combinatorial structures. We define those structures and describe some of their properties in this section. We prove extractability and succinctness of our PCS in Section 5.2.3.

A d-structure is an object that can be embedded in an $n \times \cdots \times n$ binary (hyper)matrix of dimension d (call such a binary hypermatrix an n^d matrix). Any $n \times \cdots \times n$ matrix can be canonically (after fixing an order on the dimensions) divided into n matrices of size n^{d-1} each and dimension $(d-1)$ such that they are all parallel (along the "highest" dimension). This also gives a notion of distance between two parallel matrices.

The structures are defined recursively and they are, informally, generalisations of parallelograms to higher dimensions. For example, a 1-structure is simply two points marked 1 on a line, 2-structures is a parallelogram in an $n \times n$ matrix with corners marked 1, and a 3-structure in an n^3 matrix consists of two parallelograms with the same height and base length in any two parallel submatrices of dimension 2 – note that the two parallelograms forming the 3-structure do not have to be congruent; they just satisfy a notion of similarity defined below.

Definition 5.1 (d-structure and d-similarity)**.** *A d-structure is defined by the following two mututally recursive definitions.*

- *A d-structure in an n^d matrix is formed by two $(d-1)$-similar $(d-1)$-structures in two parallel $(d-1)$-dimensional matrices.*
  A 1-structure is defined by 2 entries numbered 1 in any 1-dimensional submatrix (line).
- *Two d-structures in an n^d matrix are d-similar if all the $(d-1)$-structures that form them are pairwise $(d-1)$-similar and the distance between the $(d-1)$-dimensional (parallel) submatrices that they belong are the same for both the d-structures.*

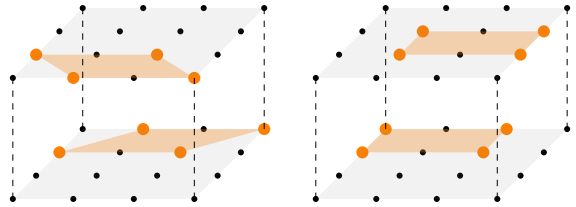
$$\begin{bmatrix} \cdot & \cdot & 1_1 & \cdot & \cdot & 1_2 & \cdot & \cdot & \cdot & \cdot & 1_3 & \cdot & \cdot & 1_4 \end{bmatrix}$$

(a) similar 1-structures are formed by $(1_1, 1_2)$ and $(1_3, 1_4)$ (there are two other dissimilar 1-structures as well)

$$\begin{bmatrix} \cdot & 1_1 & \cdot & 1_2 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1_3 & \cdot & 1_4 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1_5 & \cdot & 1_6 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

(b) similar 2-structures are formed by $(1_1, 1_2, 1_3, 1_4)$ and $(1_3, 1_4, 1_5, 1_6)$ (A dissimilar 2-structure is $(1_1, 1_2, 1_5, 1_6)$)

Figure 9: Examples of 1-structures and 2-structures



(a) (b)

Figure 10: The 8 orange nodes in the two parallel planes form a 3-structure in each cube. Both the 3-structures (in (a) and (b)) are similar.
Note that (a) is an example of a 3-structure but is not a parallelepiped.

> *Two 1-structures are defined to be 1-similar if the distance between the two 1s in both structures are equal.*

Fig. 9 and 10 give examples of some 1-, 2-, and 3-structures. Since (a) and (b) of Fig. 10 are two 3-similar 3-structures, we obtain a 4-structure by placing them in two parallel submatrices of an n^4 matrix.

An alternative, equivalent, view of d-structures is to think of an n^d binary matrix as a set of points with integer coordinates in $[n]^d$, which we use to denote the set of points in $\mathbb{Z}^d$ with coordinates in $\{0, \ldots, n-1\}$. In this view, a d-structure $D \subseteq [n]^d$ may be inductively defined as follows.

1. For $d = 1$, any set of two distinct points in $[n]$ is a 1-structure.
2. For $d > 1$, there exist $y_d \in [n]$, $0 < a_d \in [n]$, and two $(d-1)$-structures $D_0, D_1 \subseteq [n]^{d-1}$ such that
   (a) $D = \{(\mathbf{u}, y_d) : \mathbf{u} \in D_0\} \cup \{(\mathbf{v}, y_d + a_d) : \mathbf{v} \in D_1\}$, and
   (b) D_0 and D_1 are $(d-1)$-similar.

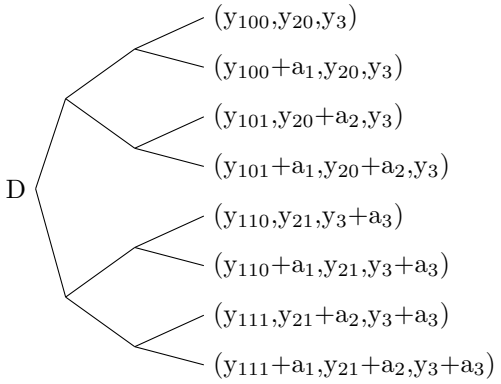
By unwinding the above definition, it is easy to see that a d-structure D defines a *unique* binary tree of recursive decomposition into $(d-1)$-, $(d-2)$-, $\cdots$ 1-structures. At *level* h (root is at level d, leaves at level 0) of this tree, we can assign a fixed "distance" value a_h (the same a_h for each of the 2^{d-h} nodes at that level), which is equal to the distance between the child $(d-h-1)$-structures (as defined above) of a $(d-h)$-structure at level h. Such a structure is naturally labeled by a binary string of length h (e.g. left child of D_s is labeled by D_{s0} and right child by D_{s1}, recursively starting with D at the root). Furthermore, if D_s is such a structure at level h, where $s \in \{0,1\}^h$, then there is an integer $y_{h,s}$ such that its two child $(d-h-1)$-structures D_{s0} and D_{s1} live in parallel planes (in dimension $d-h$) at indices $y_{h,s}$ and $y_{h,s} + a_h$ respectively (as in the second part of the definition above). These observations immediately lead to the following characterization of d-structures.

Observation: The set of points in a d-structure $D \subseteq [n]^d$ may be written as:

$$\left\{ \left(y_{1,i_1,i_2,\ldots,i_{d-1}} + i_d a_1 \ , \ \ldots \ , \ y_{j,i_1,i_2,\ldots,i_{d-j}} + i_{d-j+1} a_j \ , \ \ldots \ , \ y_d + i_1 a_d \right) \right\},$$

where $y_{h,i} \in [n]$, $0 < a_h \in [n]$ for $1 \le h \le d$, $i := (i_1, \ldots, i_{d-h}) \in \{0,1\}^{d-h}$.

The figure below illustrates the above observations for $d = 3$.



Note: d-similarity is an equivalence relation on d-structures.

We now prove our main result on d-structures.

Theorem 5.2. *The maximum number of* 1*s an* n^d *binary matrix can contain such that there exists no* d*-structure is* $N_d := n^d - (n-1)^d$*. This bound is tight.*

Proof. We will prove by induction on d the stronger claim that if an n^d binary matrix contains M 1's, then it must contain at least $M - N_d$ d-dissimilar d-structures.

When $d = 1$, the matrix is just a line of length n and $N_d = 1$. Clearly, if the number of 1s in the line is 2 or more, the number of 1-structures is non-zero. Moreover, given N ones on a line, there are at least $N - 1 = (N - N_d)$ 1-dissimilar 1-structures. Suppose the statement were true for dimension $d-1$. For dimension d, let there be $N > N_d$ ones in the n^d matrix. Divide the n^d matrix into n parallel n^{d-1} matrices and index them by $j \in [n]$. Since $(d-1)$-similarity is an equivalence relation, let the set of equivalence classes of pairwise $(d-1)$-dissimilar $(d-1)$-structures in the j^{th} n^{d-1} matrix be D_j and the number of 1s in the j^{th} submatrix be M_j. All references to structures in the rest of the proof refer to equivalence classes of structures.

Consider the subset of indices J for which $M_j > N_{d-1}$ (the D_j are then guaranteed to contain at least $(M_j - N_{d-1})$ many $(d-1)$-structures by the inductive hypothesis). WLOG, let $J = [m]$ for some $m \leq n$.

First, we can rearrange some terms: define $i(x) := |\{j : x \in D_j\}|$ for all $(d-1)$-structures $x \in \bigcup_{j\in[m]} D_j$. It is easy to see that

$$\sum_{j\in[m]} |D_j| = \sum_{x\in\bigcup_{j\in[m]} D_j} i(x)$$

Notice that there can exist at most $(n-1)^{d-1}$ $(d-1)$-dissimilar $(d-1)$-structures. (Each d-structure is defined by the distance between the two lower dimensional structures that form it - the number of

possible distances is $(n-1)$ in an n^d matrix.) Hence,

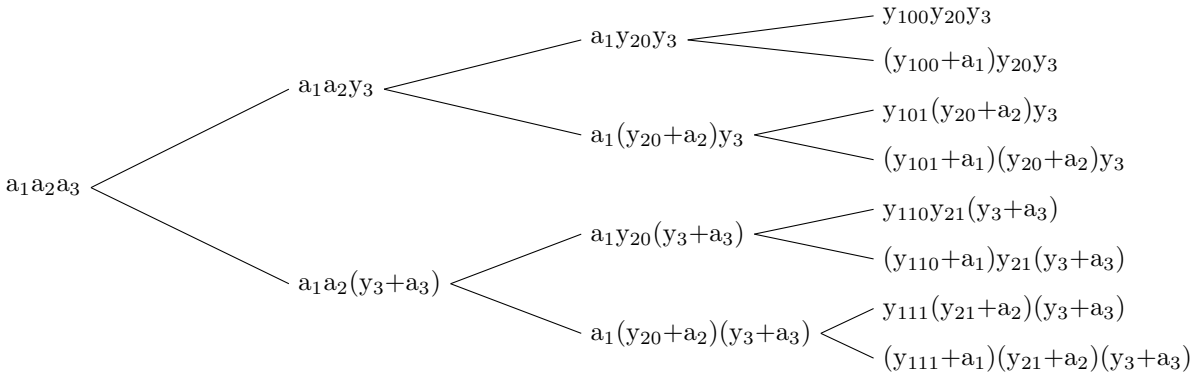
$$\begin{aligned}
(n-1)^{d-1} &\geq \left|\bigcup_{j\in[m]} D_j\right| \\
&= \sum_{j\in[m]} |D_j| - \sum_{x\in\bigcup_{j\in[m]} D_j} [i(x)-1] \\
&\geq \sum_{j\in[m]} (M_j - N_{d-1}) - \sum_{x\in\bigcup_{j\in[m]} D_j} [i(x)-1] \\
&\geq \sum_{j\in[n]} (M_j - N_{d-1}) - \sum_{x\in\bigcup_{j\in[m]} D_j} [i(x)-1] \\
&= N - nN_{d-1} - \sum_{x\in\bigcup_{j\in[m]} D_j} [i(x)-1] \\
\implies \sum_{x\in\bigcup_{j\in[m]} D_j} [i(x)-1] &\geq (N-N_d) + \left(N_d - nN_{d-1} - (n-1)^{d-1}\right) \\
&= N - N_d
\end{aligned}$$

where the last equality is given by the recurrence relation of N_d.

For any x, $[i(x)-1]$ is always non-negative. Also, if $i(x) = k$ for some $k > 1$, we obtain at least $k-1$ d-dissimilar d-structures made of $(d-1)$-similar $(d-1)$-structures (similar to x) which are in parallel n^{d-1} matrices at different distances (this argument is similar to the $d = 1$ case). Hence, the number of d-dissimilar d-structures is lower bounded by the LHS of the sum in the previous inequality, which is larger than $N - N_d$.

For any d, constructing the lower bounding instance with N_d ones is simple by induction: the inductive step divides the n^d matrix into n parallel n^{d-1} matrices. Then, fill one matrix completely with ones and fill the rest of the $(n-1)$ matrices with N_{d-1} ones in a way that there exist no $(d-1)$-structures in any of them (possible by the inductive hypothesis on the previous step). Clearly, there can exist no d-structure in this n^d matrix, as $(d-1)$-structures only exist in a single n^{d-1} submatrix. The number of ones in the n^d matrix is $n^{d-1} + (n-1)N_{d-1} = N_d$ (by another recurrence relation for N_d). □

In our application, each point in a d-structure will map to a monomial on its coordinates. When we add all these monomials from d-structure with appropriate signs, namely recursively subtract the "shifted" monomial from its unshifted sibling, we induce a lot of cancellations and end up with a monomial on the "distances" a_h defining d-structure. See the figure below for an illustration. We critically use this cancellation property in the next section to vastly simplify a collection of equations among inner products between the prover's committed vector and the verifier's test/query vector.



5.2.3 Extractability of Dew-PC

We are now ready to prove the extractability of our PCS Dew-PC.

Define

$$S_{log} \coloneqq \left\{ \frac{m\alpha - n}{k} \ : \ m, n, k \in \mathbb{Z},\ gcd(m,k) = 1,\ 0 < m \leq k < p^{\log l},\ -l < n < k + l \right\}$$

and functions $\chi_m, \chi_n : S_{log}^q \to \mathbb{Q}^q$ which isolates the vector of fractions m/k and n/k from the elements of S^q:

$$\mathbf{v} \in S_{log} \implies \mathbf{v} = \left(\frac{m_i\alpha - n_i}{k_i} \right)_i, \quad \chi_m(\mathbf{v}) := \left(\frac{m_i}{k_i} \right)_i, \quad \text{and} \quad \chi_n(\mathbf{v}) := \left(\frac{n_i}{k_i} \right)_i.$$

These functions can be made well-defined by fixing a representation of elements of S: for any $d \in S$, consider the representation (m, n, k) as the one with the smallest denominator k and if there are multiple such representations, we pick the one with the smallest m.

For a finite (multi-)set of rational numbers in reduced form

$$F \subset \left\{ \frac{a}{b} \ : \ a \in \mathbb{Z},\ b \in \mathbb{Z}^+ \right\},$$

define $\text{DenLCM}(F)$ as the *LCM of the denominators* of elements of F, $\text{lcm}(\{b \ : \ a/b \in F\})$. For a vector $\mathbf{f}$, $\text{DenLCM}(\mathbf{f})$ is defined to be the DenLCM of the set of its coefficients.

Theorem 5.3. *The polynomial commitment scheme Dew-PC satisfies Extractability (Def. 2.7) in the Generic Group Model.*

Proof. The proof of this theorem consists of two theorems about the two protocols **logTEST** and **logIPP**. Both theorems rely on the fact that the adversary is *generic*.

Theorem 5.4. *If the Verifier in* **logTEST** *outputs accept with non-negligible probability over the choice of the random* $\mathbf{z_1}, \ldots, \mathbf{z}_{\log \mathbf{l}} \in \mathbb{Z}_p^2$*, there exists an extractor that outputs vectors* $\mathbf{c}, \mathbf{d} \in \mathbb{Z}_p^l$ *satisfying the following properties:*

1. $C = g^{\sum_{i=0}^{l-1}(c_i + \alpha d_i)\alpha^{2i}}$
2. $\mathbf{d} \in S_{log}^l$
3. $\text{DenLCM}(\chi_m(\mathbf{d})) < p^{2\log l + 1}$

Proof. We define the extractor Ext_T to first invoke the PoKPE extractor for C, which outputs an exponent $c > 0$ such that $C = g^c$. Since E also passes the PoKPE protocol and $C \cdot E = g^{\alpha^{2l-1}}$, we infer that $c < \alpha^{2l-1}$. Consider the base-α representation of c, which is a $2l$-length vector. Ext_T then outputs the even indexed coordinates as $\mathbf{c}$ and the odd indexed coordinates as $\mathbf{d}$.

Note that by definition, the first condition in the theorem is satisfied: $C = g^{\sum_{i=0}^{l-1}(c_i + \alpha d_i)\alpha^{2i}}$. An honest prover would clearly choose $d_i = 0$ and $c_i = x_i$, $0 \leq c_i \leq p - 1$ for $0 \leq i \leq l - 1$ to commit to a vector $\mathbf{x} \in \mathbb{Z}_p^l$. However, with a cheating prover, we are only guaranteed (at this point) that $0 \leq c_i, d_i \leq \alpha - 1$.

Now we use the checks done by the **logTEST** verifier to derive conditions on the above extracted vector and show the second part of the theorem – $\mathbf{d} \in S_{\log}^l$.
Suppose the prover succeeds with a non-negligible probability over the random choice of $\mathbf{z_1}, \ldots, \mathbf{z}_{\log \mathbf{l}}$ from $\mathbb{Z}_p^2$.
Fix an arbitrary index $0 \leq k \leq l - 1$, equivalently its binary representation $(k_1, \ldots k_{\log l})$. Consider the partition of the space $\mathbb{Z}_p^{2\log l}$ by sets of the form

$$T_{\mathbf{q}} \coloneqq \{(x_{1,k_1}, \ldots, x_{\log l, k_{\log l}}, \mathbf{q}) : x_{j,k_j} \in \mathbb{Z}_p,\ 1 \leq j \leq \log l\} \text{ for } \mathbf{q} \in \mathbb{Z}_p^{\log l}.$$

Since the success probability of the prover is non-negligible, it is at least $\frac{\log l}{p}$. Hence, at least one of these sets (which are $\log l$-dimensional spaces) must have more than $\log l p^{\log l - 1} \geq p^{\log l} - (p-1)^{\log l}$ accepting points, which implies by Theorem 5.2 that there exists a $\log l$-structure in this space consisting of l accepting points. A d-structure for any dimension d is a generalisation of a hyper-rectangle / box

(see Section 5.2.2 for definition), which is structured in a way to allow for subtraction between pairs of vertices.

For some fixed $1 < a_j < p$ and for all $g_1 g_2 \dots g_\tau \in \{0,1\}^\tau$, let this $\log l$-structure be represented by

$$B \coloneqq \{ \big(y_{1,g_1,g_2,\dots,g_{\log l-1}} + g_{\log l} a_1, \dots, y_{j,g_1,g_2,\dots,g_{\log l-j}} + g_{\log l-j+1} a_j, \dots, y_{\log l} + g_1 a_{\log l}\big) : \\ 1 \le j \le \log l,\ y_{\dots} \in \mathbb{Z}_p \}$$

All the points in B can be considered as the leaves of a binary tree, with leaves indexed as $g_1 g_2 \dots g_{\log l}$. Starting from the root, at each node, the left child is labeled 1 and the right child is labeled 0. Thus the leftmost leaf would have index $11\dots1$, and the rightmost leaf will have index $00\dots0$.

Now, Lemma 3.6 and 3.7 give us equations corresponding to each accepting point on the $\log l$-structure relating $\mathbf{c}, \mathbf{d}$ (given by Ext_T from Lemma 3.10) and the random variables x_{j,i_j}. We recall the definition of the query vector $\mathbf{z}$ from Equation (23), where for each coordinate z_i if the binary representation of $i = i_1 \dots i_{\log l}$, then

$$z_i \equiv z_{i_1,i_2,\dots,i_{\log l}} \coloneqq \prod_{j=1}^{\log l} x_{j,i_j}$$

$$\langle \mathbf{d}, \mathbf{z} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z} \rangle}{\alpha} \right\rfloor + u_1 \mod \alpha = 0 \mod \alpha$$

The term $\langle \mathbf{d}, \mathbf{z} \rangle \mod \alpha$ can be expanded as follows:

$$\begin{aligned}
\langle \mathbf{d}, \mathbf{z} \rangle \mod \alpha &= \sum_{i_1,i_2,\dots,i_{\log l} \in \{0,1\}} d_{i_1,i_2,\dots,i_{\log l}} \cdot \prod_{j=1,}^{\log l} x_{j,i_j} \mod \alpha \\
&= \sum_{i_1,i_2,\dots,i_{\log l} \in \{0,1\}} d_{i_1,i_2,\dots,i_{\log l}} \cdot \prod_{j=1}^{\log l} x_{j,k_j} \cdot \prod_{\substack{j=1,\\ i_j \neq k_j}}^{\log l} x_{j,i_j} \mod \alpha \\
&= \sum_{i_1,i_2,\dots,i_{\log l} \in \{0,1\}} d_{i_1,i_2,\dots,i_{\log l}} \cdot \prod_{j=1}^{\log l} \big(y_{j,g_1,g_2,\dots,g_{\log l-j}} + g_{\log l-j+1} a_j\big) \cdot \prod_{\substack{j=1,\\ i_j \neq k_j}}^{\log l} x_{j,i_j} \mod \alpha
\end{aligned}$$

This expansion holds at height $\log l$ (for all leaves $g_1 g_2 \dots g_{\log l}$). To obtain the required conditions on $\mathbf{d}$, we subtract the l equations in a specific order to cancel out all but one term. This is possible due to the fact that the coefficients of $d_{i_1 \dots i_{\log l}}$ are multilinear in each of the randomly sampled variables.

More precisely, at any intermediate height in the binary tree, we obtain the equation at that node by subtracting the equation at the right child from the equation at the left child. For instance, at height $(\log l - 1)$, the first term takes the form :

$$a_1 \sum_{i_1,i_2,\dots,i_{\log l-1} \in \{0,1\}} d_{k_1,i_2,\dots,i_{\log l-1},i_{\log l}} \cdot \prod_{j=2}^{\log l} \big(y_{j,g_1,g_2,\dots,g_{\log l-j}} + g_{\log l-j+1} a_j\big) \cdot \prod_{\substack{j=2\\ i_j \neq k_j}}^{\log l} x_{j,i_j} \mod \alpha$$

In general, we get at height $0 \le t < \log l$,

$$\prod_{j=1}^{\log l-t} a_i \sum_{i_1,i_2,\dots,i_t \in \{0,1\}} d_{k_1,k_2,\dots,k_{\log l-t},i_{\log l-t+1},\dots,i_{\log l}} \cdot \prod_{j=\log l-t+1}^{\log l} \big(y_{j,g_1,g_2,\dots,g_{\log l-j}} + g_{\log l-j+1} a_j\big) \\ \cdot \prod_{\substack{j=\log l-t+1\\ i_j \neq k_j}}^{\log l} x_{j,i_j} \mod \alpha$$

Notice that at the root, i.e., at height 0, we are left with the single term

$$a_1 \dots a_{\log l} \cdot d_{k_1,k_2,\dots,k_{\log l}}$$

For the rest of the equation, we only use bounds on the other terms and not the exact expression. The actual expression is a symbolic subtraction of the floor terms and the 'u' terms. This is similar to what is done in Lemma 4.7 generalised to higher dimensions.
At level $0 \le t \le \log l$, indexing the l points/leaves by $\mathbf{z}_{g_1 g_2 \dots g_{\log l}}$ from left to right, we get the expression for the remaining two terms (call this expression n) as

$$\left(\sum_{g_1,g_2,\dots,g_{\log l} \in \{0,1\}} (-1)^{\sum_{j=1}^{\log l} i_j} \cdot \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_{g_1,g_2,\dots,g_{\log 1}}} \rangle}{\alpha} \right\rfloor + \sum_{g_1,g_2,\dots,g_{\log l} \in \{0,1\}} (-1)^{\sum_{j=1}^{\log l} i_j} \cdot u_{g_1,g_2,\dots,g_{\log l}} \right) \mod \alpha$$

For example, in the 3-dimensional case, labeling the 2^3 points/leaves by $\mathbf{z_i}$, we get the following expression

$$\begin{aligned} n \coloneqq & \left[\left(\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_1} \rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_2} \rangle}{\alpha} \right\rfloor \right) - \left(\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_3} \rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_4} \rangle}{\alpha} \right\rfloor \right) \right] \\ & - \left[\left(\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_5} \rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_6} \rangle}{\alpha} \right\rfloor \right) - \left(\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_7} \rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_8} \rangle}{\alpha} \right\rfloor \right) \right] \\ & + [(u_1 - u_2) - (u_3 - u_4)] - [(u_5 - u_6) - (u_7 - u_8)] \end{aligned}$$

Since for all x, $x - 1 \le \lfloor x \rfloor < x$ and $u \in \{0, 1\}$, we can see that for the above expression,

$$\begin{aligned} \frac{c_i \cdot a_1 a_2 a_3}{\alpha} - 8 < n < \frac{c_i \cdot a_1 a_2 a_3}{\alpha} + 8 \\ \implies -8 < n < a_1 a_2 a_3 + 8 \end{aligned}$$

In our general case, using the same bounds on the floor terms, we can show that the resulting equation at the root node is of the form

$$a_1 \dots a_{\log l} \cdot d_{k_1,\dots,k_{\log l}} = -(n) \mod \alpha$$

where $-l < n < \prod_{i=1}^{\log l} a_i + l$. Hence, there exists m such that

$$d_{k_1,\dots,k_{\log l}} = \frac{m\alpha - n}{a_1 \cdots a_{\log l}}$$

where $m \le \prod_{i=1}^{\log l} a_i < p^{\log l}$ (as $d_{k_1,\dots,k_{\log l}} < \alpha$) and $-l < n < \prod_{i=1}^{\log l} a_i + l$ as shown above.

Hence, $d_{k_1,\dots,k_{\log l}} \in S_{log}$. Since $(k_1, \dots, k_{\log l})$ was arbitrary, $\mathbf{d} \in S_{log}^l$.

The third part of the theorem is proved below in Lemma 5.6.

□

Before proving part (3) of the extraction theorem, we recall the statement of the extended Schwartz-Zippel Lemma as given in [BF22]:

Theorem 5.5 (Corollary 2 from [BF22])**.** *For all N such that $\log N \ge 8\mu^2 + \log_2(2\mu)\lambda$, we have for any μ-linear polynomial f that is coprime with N,*

$$Pr_{x \leftarrow [0,m)^\mu}[f(x) = 0 \mod N] \le 2^{-\lambda} + \frac{\mu}{m}$$

Lemma 5.6 (Part (3) of **logTEST** extraction)**.** *If the verifier accepts with non-negligible probability, we must have* $\mathrm{DenLCM}(\chi_m(\mathbf{d})) \le p^{2\log l + 1}$.

Proof. Suppose the prover succeeds with non-negligible probability over the random choice of $\mathbf{z_1}, \ldots, \mathbf{z}_{\log l}$ from $\mathbb{Z}_p^2$ and that $\text{DenLCM}(\chi_m(\mathbf{d})) > p^{2\log l+1}$. For simplicity, denote $P := p^{4\log l+2}$. We provide an algorithm to find a specific sub-vector $\mathbf{d}^*$ of $\mathbf{d} = (d_i)_{i\in[l]} \equiv (d_{i_1\ldots i_{\log l}})_{i_1\ldots i_{\log l}\in\{0,1\}^{\log l}}$ such that $p^{2\log l+1} < \text{DenLCM}(\mathbf{d}^*) < P$. Consider this algorithm:

0. If $\text{DenLCM}(\chi_m(\mathbf{d})) < P$, HALT with output 0.
1. Divide the l-length vector $\mathbf{d}$ into two sub-vectors, where each sub-vector contains elements with indices i such that $i_1 = 0$ or $i_1 = 1$ respectively. (Each sub-vector consists of $l/2$ elements)

1.5. Consider the sub-vector $\mathbf{d}^*$ that satisfies $\text{DenLCM}(\chi_m(\mathbf{d}^*)) > p^{2\log l+1}$.
If $\text{DenLCM}(\chi_m(\mathbf{d}^*)) < P$, HALT with output 1.

For every step $\tau \geq 2$, (if not already halted)

τ. Divide the sub-vector chosen in step $\tau-1$ into two sub-vectors, corresponding to the elements with indices such that $i_\tau = 0$ and $i_\tau = 1$ respectively. (Each sub-vector now consists of $l/2^\tau$ elements)

τ.5. Consider the sub-vector $\mathbf{d}^*$ that satisfies $\text{DenLCM}(\mathbf{d}^*) > p^{2\log l+1}$.
If $\text{DenLCM}(\chi_m(\mathbf{d}^*)) < P$, HALT with output τ.

We need to show that the algorithm is well-defined and that it terminates. Note that whenever the algorithm moves to step τ for any τ, the previous halting condition was not satisfied, i.e., the sub-vector $\mathbf{d}^*$ in step τ will have $\text{DenLCM}(\chi_m(\mathbf{d}^*)) > P$. A simple application of the Pigeonhole Principle then shows the existence of a further sub-vector satisfying the condition $\text{DenLCM} > p^{2\log l+1}$, as required in step τ.5.

We can also show that this algorithm will terminate in at most $\log l - 1$ steps. Notice that at step $\log l - 1$, there exists a sub-vector with $\text{DenLCM} > p^{2\log l+1}$ (by the above argument) with two elements in it. However, from part (2) of Theorem 5.3, we know that each element d_i in the sub-vector satisfies $\chi_m(d_i) < p^{\log l}$, hence the sub-vector satisfies $\text{DenLCM}(\chi_m(\mathbf{d}^*)) < P$, and the algorithm will terminate, if it had not already terminated by then. Hence, the termination occurs in at most $\log l - 1$ steps.

Case 1: Suppose the algorithm halted with output $\tau = 0$, which implies that the entire vector $\mathbf{d}$ satisfies $p^{2\log l+1} < \text{DenLCM}(\chi_m(\mathbf{d})) < P$. Now consider any accepting point $\mathbf{z}$ in the space $\mathbb{Z}_p^{2\log l}$. Then, Lemma 3.6 and 3.7 give us an equation relating $\mathbf{c}, \mathbf{d}$ and $\mathbf{z}$:

$$\langle \mathbf{d}, \mathbf{z}\rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}\rangle}{\alpha} \right\rfloor + u_1 \mod \alpha = 0 \mod \alpha \tag{25}$$

$$\implies \langle \mathbf{d}, \mathbf{z}\rangle \mod \alpha = -\left(\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}\rangle}{\alpha} \right\rfloor + u_1 \right) \mod \alpha$$

$$\implies \alpha \cdot \left(\sum_{i_1,i_2,\ldots,i_{\log l}\in\{0,1\}} \frac{m_{i_1\ldots i_{\log l}}}{k_{i_1\ldots i_{\log l}}} \prod_{j=1}^{\log l} x_{j,i_j} \right) - \left(\sum_{i_1,i_2,\ldots,i_{\log l}\in\{0,1\}} \frac{n_{i_1\ldots i_{\log l}}}{k_{i_1\ldots i_{\log l}}} \prod_{j=1}^{\log l} x_{j,i_j} \right) = M'\alpha - N'$$

using the fact that $d_i = (m_i\alpha - n_i)/k_i$ for each $i \in [l]$ (from Theorem 5.4). Call the two terms on the LHS M and N respectively. Then, since $\text{DenLCM}(\chi_m(\mathbf{d})) < P$, we can use Lemma 3.8 to show that $M = M', N = N'$, and specifically $M \in \mathbb{Z}$. This can be expanded as follows:

$$\sum_{i_1,i_2,\ldots,i_{\log l}\in\{0,1\}} \frac{m_{i_1\ldots i_{\log l}}}{k_{i_1\ldots i_{\log l}}} \prod_{j=1}^{\log l} x_{j,i_j} \in \mathbb{Z}$$

$$\implies \sum_{i_1,i_2,\ldots,i_{\log l}\in\{0,1\}} m_{i_1\ldots i_{\log l}} \cdot \frac{L}{k_{i_1\ldots i_{\log l}}} \prod_{j=1}^{\log l} x_{j,i_j} = 0 \mod L$$

Here, we have that $L > p^{2\log l+1}$ and that the coefficients of the multilinear polynomial are coprime to L (as L is the LCM of the $k_{i_1\ldots i_{\log l}}$). Hence, using Theorem 2 (Inverse LCSZ) from [BF22] (restated

here in Theorem 5.5), we can show that this equation is satisfied by at most negligible probability $\epsilon \leq (2^{-\kappa} + \log l/p)$ over the choice of $x_{j,i_j} \in \mathbb{Z}_p$ as long as $p^{2\log l+1} > 2^{\kappa}(\log l)^{\kappa} l^{8\log l}$ – the condition given in the theorem.
Hence, the prover can succeed for as many points as the number of solutions to the above equation; which is a negligible fraction of points in the space $\mathbb{Z}_p^{2\log l}$, which is a contradiction.

Case 2: Now suppose the algorithm halted with output $1 \leq \tau \leq \log l - 1$, with the sub-vector corresponding to those elements of $\mathbf{d}$ with fixed indices $k_1, \ldots, k_\tau$. We also know from the above discussion that this sub-vector $\mathbf{d}^*$ of $\mathbf{d}$ satisfies $p^{2\log l+1} < \text{DenLCM}(\chi_m(\mathbf{d}^*)) < P$. Define $L := \text{DenLCM}(\mathbf{d}^*)$.
Consider the partition of $\mathbb{Z}_p^{2\log l}$ by sets (τ-dimensional spaces) of the form

$$T_{\mathbf{q}} := \{(x_{1,k_1}, \ldots, x_{\tau,k_\tau}, \mathbf{q}) : x_{j,k_j} \in \mathbb{Z}_p,\ 1 \leq j \leq \tau\} \text{ for } \mathbf{q} \in \mathbb{Z}_p^{2\log l - \tau}.$$

Denote by $||T_{\mathbf{q}}||$ the number of accepting points in $T_{\mathbf{q}}$ (accepting points are those values of $\mathbf{z} \in \mathbb{Z}_p^{2\log l}$ for which the prover can succeed in **logTEST**). Now suppose for some $\mathbf{q}$ it holds that $||T_{\mathbf{q}}|| > N_\tau$ ($N_\tau \leq \tau p^{\tau-1}$ is as defined in § 5.2.2). Then, by Theorem 5.2, there exists a τ-structure B of accepting points in the τ-dimensional space $T_{\mathbf{q}}$, where B can be represented by the set:

For some fixed $1 < a_j < p$ and for all $g_1 g_2 \ldots g_\tau \in \{0,1\}^\tau$, let this τ-structure be represented by

$$B := \{\left(y_{1,g_1,g_2,\ldots,g_{\tau-1}} + g_\tau a_1, \ldots, y_{j,g_1,g_2,\ldots,g_{\tau-j}} + g_{\tau-j+1} a_j, \ldots, y_\tau + g_1 a_\tau\right) : 1 \leq j \leq \tau,\ y_{\ldots} \in \mathbb{Z}_p\}$$

All the points in B can be considered as the leaves of a binary tree, with leaves indexed as $g_1 g_2 \ldots g_\tau$. These points are just specific choices of the x_{j,i_j} in the following equations, which allow us to recursively subtract (fold) them.

Now, Lemma 3.6 and 3.7 give us equations corresponding to each accepting point on the $\log l$-structure relating $\mathbf{c}, \mathbf{d}$ (given by Ext_T from Lemma 3.10) and the random variables x_{j,i_j}. We recall the definition of the query vector $\mathbf{z}$ from Equation (23), where for each coordinate z_i is the binary representation of $i = i_1 \ldots i_{\log l}$, then

$$z_i \equiv z_{i_1,i_2,\ldots,i_{\log l}} := \prod_{j=1}^{\log l} x_{j,i_j}$$

$$\langle \mathbf{d}, \mathbf{z} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z} \rangle}{\alpha} \right\rfloor + u_1 \mod \alpha = 0 \mod \alpha \tag{26}$$

The term $\langle \mathbf{d}, \mathbf{z} \rangle \mod \alpha$ can be expanded as follows (for $\mathbf{z} \in B$):

$$\begin{aligned}
\langle \mathbf{d}, \mathbf{z} \rangle \mod \alpha &= \sum_{i_1,i_2,\ldots,i_{\log l} \in \{0,1\}} d_{i_1,i_2,\ldots,i_{\log l}} \cdot \prod_{j=1,}^{\log l} x_{j,i_j} \mod \alpha \\
&= \sum_{i_1,i_2,\ldots,i_{\log l} \in \{0,1\}} d_{i_1,i_2,\ldots,i_{\log l}} \cdot \prod_{j=1}^{\tau} x_{j,k_j} \cdot \prod_{\substack{j=1, \\ i_j \neq k_j}}^{\tau} x_{j,i_j} \prod_{j=\tau+1}^{\log l} x_{j,i_j} \mod \alpha \\
&= \sum_{i_1,i_2,\ldots,i_{\log l} \in \{0,1\}} d_{i_1,i_2,\ldots,i_{\log l}} \cdot \prod_{j=1}^{\tau} \left(y_{j,g_1,g_2,\ldots,g_{\tau-j}} + g_{\tau-j+1} a_j\right) \cdot \prod_{\substack{j=1, \\ i_j \neq k_j}}^{\tau} x_{j,i_j} \prod_{j=\tau+1}^{\log l} x_{j,i_j} \mod \alpha
\end{aligned}$$

Similar to the process in part (2) of Theorem 5.4, we can define a binary tree of equations of height τ and fold the equations using the subtraction operation. The inner product term at the root node can then be written as:

$$\prod_{j=1}^{\tau} a_j \sum_{i_{\tau+1},i_{\tau+2},\ldots,i_{\log l} \in \{0,1\}} d_{k_1,k_2,\ldots,k_\tau,i_{\tau+1},\ldots,i_{\log l}} \cdot \prod_{j=\tau+1}^{\log l} x_{j,i_j} \mod \alpha$$

We can then write the folded equation (26) as

$$\prod_{j=1}^{\tau} a_j \sum_{i_{\tau+1},i_{\tau+2},\ldots,i_{\log l}\in\{0,1\}} d_{k_1,k_2,\ldots,k_\tau,i_{\tau+1},\ldots,i_{\log l}} \cdot \prod_{j=j+1}^{\log l} x_{j,i_j} \mod \alpha = -N' \mod \alpha$$

$$\implies \alpha \cdot \left(\prod_{j=1}^{\tau} a_j \sum_{i_{\tau+1},\ldots,i_{\log l}\in\{0,1\}} \frac{m_{\ldots}}{k_{\ldots}} \prod_{j=\tau+1}^{\log l} x_{j,i_j}\right) - \left(\prod_{j=1}^{\log l} a_j \sum_{i_{\tau+1},\ldots,i_{\log l}\in\{0,1\}} \frac{n_{\ldots}}{k_{\ldots}} \prod_{j=\tau+1}^{\log l} x_{j,i_j}\right) = M'\alpha - N'$$

$$\equiv M\alpha - N = M'\alpha - N'$$

using the fact that $d_i = (m_i\alpha - n_i)/k_i$ for each $i \in [l]$ (from Theorem 5.4) and where M, N are defined as the terms in the brackets in the above equation. We can now use Lemma 3.8 and that $L < P$ to show that $M = M', N = N'$ (this only requires $\alpha > 2l^2 p^{5\log l+2}$). Importantly, since $M', N' \in \mathbb{Z}$, so are M and N. Then write $M \in \mathbb{Z}$ as follows:

$$\prod_{j=1}^{\tau} a_j \sum_{i_{\tau+1},\ldots i_{\log l}\in\{0,1\}} \frac{m_{i_{\tau+1},\ldots i_{\log l}}}{k_{i_{\tau+1},\ldots i_{\log l}}} x_{\tau+1,i_{\tau+1}} \ldots x_{\log l,i_{\log l}} \in \mathbb{Z}$$

$$\implies \prod_{j=1}^{\tau} a_j \sum_{i_{\tau+1},\ldots i_{\log l}\in\{0,1\}} m_{i_{\tau+1},\ldots i_{\log l}} \cdot \frac{L}{k_{i_{\tau+1},\ldots i_{\log l}}} x_{\tau+1,i_{\tau+1}} \ldots x_{\log l,i_{\log l}} = 0 \mod L$$

$$\implies \sum_{i_{\tau+1},\ldots i_{\log l}\in\{0,1\}} \frac{m_{i_{\tau+1},\ldots i_{\log l}} L}{k_{i_{\tau+1},\ldots i_{\log l}}} x_{\tau+1,i_{\tau+1}} \ldots x_{\log l,i_{\log l}} = 0 \mod L'$$

Here L' is such that $L'|L$ and $L/\prod_j a_j > p^{\log l}$, as $L > p^{2\log l+1}$ and each $a_j < p$. In addition, since L is the LCM of all the $k_{\ldots}$, we have the condition that the coefficients of the above multilinear polynomial are coprime to L'.

Note that

1. the elements x_{j,i_j} are fixed by the vector $\mathbf{q}$ that defines each $T_{\mathbf{q}}$, and
2. all τ-similar τ-structures have the same values of a_j for $1 \le a_j \le \tau$.

Hence, any bound on the number of solutions to this equation also bounds the total number of any arbitrary equivalence class of τ-similar τ-structures over all choices of $\mathbf{q}$ (and hence $T_{\mathbf{q}}$).

This equation is multilinear over the variables x_{j,i_j} and is of total degree at most $\log l$. We can use Theorem 2 (Inverse LCSZ) from [BF22] to show that this equation is satisfied by at most negligible probability ϵ over the choice of $x_{j,i_j} \in \mathbb{Z}_p$ as long as $p^{\log l} > 2^\kappa (\log l)^\kappa l^{8\log l}$.
This argument holds for any arbitrary τ-structure. We can bound the number of τ-dissimilar τ-structures over all $T_{\mathbf{q}}$ as $\epsilon \cdot p^\tau \cdot p^{2\log l-\tau}$ (as there are $< p^\tau$ unique τ-structures). Then, we use the stronger statement proved in Theorem 5.2 to obtain a bound on the number of accepting points in all the $T_{\mathbf{q}}$s, which is

$$p^{2\log l}\epsilon + p^{2\log l-\tau} \cdot N_\tau \le \left(\epsilon + \frac{\tau}{p}\right) p^{2\log l}$$

The probability of success for the prover is then $\epsilon + \tau/p$ which is negligible for all $0 \le \tau \le \log l - 1$. This contradicts the fact that the prover succeeds with non-negligible probability, hence DenLCM $< p^{2\log l+1}$. □

Theorem 5.7. *If the Verifier in* **logIPP** *outputs accept with non-negligible probability and the Verifier of* **logTEST** *also did so, there exists an extractor that outputs an opening $\tilde{f} \in \mathbb{Z}_p[x]$ and an opening hint $\tilde{\mathbf{c}}$ in $\mathbb{Q}(\log l + 1, l + 1)$ for C such that $v = \tilde{f}(x)$.*

Proof. We can define the extractor Ext for eval using the extractor Ext_T of **TEST**. Since the **TEST** verifier outputs accept with non-negligible probability, there exists Ext_T that outputs vectors $\mathbf{c}, \mathbf{d}$ such that they satisfy the conditions in Theorem 5.4. Ext invokes Ext_T and performs the following computations on $\mathbf{c}, \mathbf{d}$:

1. Compute m_i, n_i, k_i for every i such that $d_i = \frac{m_i\alpha - n_i}{k_i}$.
2. Define vectors $\mathbf{c}', \mathbf{d}' \in \mathbb{Z}_p^l$ as
$$c'_i \coloneqq c_i + \frac{m_{i-1}}{k_{i-1}} \text{ and } d'_i \coloneqq -\frac{n_i}{k_i}.$$
$(m_{i-1} \coloneqq 0, k_{i-1} \coloneqq 1)$
3. Output $\mathbf{c}' + \alpha\mathbf{d}'$ as the opening hint, and $(\mathbf{c}' + \alpha\mathbf{d}') \mod p$ as the opening to the commitment.

Note that this extractor is indeed efficient, since Ext_T is efficient and the only non-trivial computations done are in Step 1 above. This can be done efficiently, details are given in Lemma 4.9.

By construction, this is a valid opening hint, as $\mathbf{d} \in S^l_{log} \implies \mathbf{c}' + \alpha\mathbf{d}' \in \mathbb{Q}(\log l + 1, 2l + 1)$ and since it is just a rearrangement of the coordinates of $\mathbf{c}$ and $\mathbf{d}$, we have $C = g^{\sum_{i=0}^{l-1}(c'_i + \alpha d'_i)\alpha^{2i}}$, which is one of the required conditions.

Now, since the verifier accepts in **IPP**, Lemmas 3.6 and 3.7 along with the final check of the verifier gives us equations corresponding to the coefficients of α^{2l-1} and α^{2l-2}:

$$\langle \mathbf{d}, \mathbf{q}^+ \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u' = 0 \mod \alpha \tag{27}$$

$$\langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u = v + np \mod \alpha \tag{28}$$

Due to the bounds on coefficients of $\mathbf{z}$, we know that Equation 27's LHS must be either 0 or α. Also define

$$M' \coloneqq \sum_{i=0}^{l-1} \frac{m_i}{k_i} q_i^+, \text{ and } N' \coloneqq \sum_{i=0}^{l-1} \frac{n_i}{k_i} q_i^+$$

1. If the LHS of Equation (27) is 0, then so are each of the terms in the LHS, as they are all non-negative. Hence, $\langle \mathbf{d}, \mathbf{q}^+ \rangle = 0 \mod \alpha$, and by definition, $u = 0$.
Hence, we can simplify Equation 28
$$\begin{aligned} v + np &= \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u \mod \alpha \\ &= \langle \mathbf{c}, \mathbf{q} \rangle + \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \mod \alpha = \langle \mathbf{c}, \mathbf{q} \rangle + M' - \frac{N'}{\alpha} \mod \alpha \end{aligned}$$
$M' - N'/\alpha$ is an integer and since $N' < \alpha$ by choice of α and the query vector $\mathbf{q} \implies N' = 0$.
$$\begin{aligned} &v + np = \langle \mathbf{c}, \mathbf{q} \rangle + M' \mod \alpha = \langle \mathbf{c}', \mathbf{q} \rangle \mod \alpha \\ &\implies v = \langle \mathbf{c}', \mathbf{q} \rangle \mod p = \langle \mathbf{c}', \mathbf{q} \rangle + \alpha \langle \mathbf{d}', \mathbf{q} \rangle \mod p, \end{aligned}$$
as $\alpha = 0 \mod p$ and $\langle \mathbf{d}', \mathbf{q} \rangle$ is invertible modulo p (or simply $0 \mod p$). In either case, $v = \langle \mathbf{c}' + \alpha\mathbf{d}', \mathbf{q} \rangle \mod p$.
2. If the LHS is α, we get that $u = 1$ by definition. From the properties of $\mathbf{c}$ and $\mathbf{d}$, we know that $M' \in \mathbb{Q}$ has a denominator of at most $p^{\log l}$ (and coprime to α in the reduced form). Write Equation 27 in the form $\langle \mathbf{d}, \mathbf{q}^+ \rangle = M\alpha - N$, by moving all the terms but the inner product to the RHS and calling it N. Now, Lemma 3.8 implies that $M', N' \in \mathbb{Z}$ (since $\alpha = p^{5 \log l}$).
Now,
$$\begin{aligned} v + np &= \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u \mod \alpha \\ &= \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor M' - \frac{N'}{\alpha} \right\rfloor + 1 \mod \alpha \\ &= \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + M' - 1 + 1 \mod \alpha \end{aligned}$$
as $N' < \alpha$. Hence, as in the previous step, we get that $v = \langle \mathbf{c}' + \alpha\mathbf{d}', \mathbf{q} \rangle \mod p$.

Thus, the extracted opening equals the claimed inner product v in both cases. □

□

5.2.4 Succinctness of Dew-PC

By the earlier remark about Fiat-Shamir, Theorem 5.8 below implies that the noninteractive version of Dew-PC is succinct according to Def. 2.8.

Theorem 5.8 (Proof and Verifier Succinctness)**.** *In the PCS Dew-PC, the commitment and evaluation proof sizes are* $\mathsf{poly}(\kappa)$ *and the Verifier runs in time* $\mathsf{poly}(\kappa) \cdot \mathsf{log}(l)$.

Proof. Proof succinctness is easy to see; the commitment is a single group element and the evaluation protocol only communicates a constant number of group elements (the PoE protocols are also constant-sized) and the query vector elements $\mathbf{x}_1, \ldots, \mathbf{x}_{\log l}$. However, the $2 \log l$ *random* field elements are sent succinctly by using a Pseudorandom Generator (PRG) with a short seed to generate the vector locally (steps 3, 4 in Fig.7) – this brings the communication back to constant.

To analyse the verifier computation, notice that the the only potentially expensive computations are the computation of σ and raising group elements to large powers (the PoKPE protocols consist of a constant number of PoKE protocols, which are efficient). The group exponentiations are made more efficient for the verifier by engaging in a constant number of PoE protocols (Def 2.5) with the prover. The only remaining bottleneck is the computation of $\sigma \mod q$ for some prime q in the invocation of Weslowski's PoE (this is true for σ in both **logTEST** and **logIPP**)

In **logTEST**, using the definition of the test vector in (23) and direct manipulation implies that $\sigma \mod q$ can be computed in $O(\log l)$ time in the following way.

$$\sigma \mod q = \sum_{k=0}^{l-1} \alpha^{2l-2-2k} z_k \mod q = \alpha^{2l-2} \prod_{i=1}^{\log l} \left(x_{i,0} + x_{i,1} \alpha^{-2^{i+1}} \right) \mod q$$

In **logIPP**, by a similar manipulation as above using the definition of the the query vector in (24), we obtain

$$\sigma \mod q = \sum_{k=0}^{l-1} \alpha^{2l-1-2k} q_k \mod q = \alpha^{2l-1} \prod_{i=0}^{\log l - 1} \left(1 + (x^{2^i} \mod p)\, \alpha^{-2^{i+1}} \right) \mod q$$

Also note that for efficient computation, we need to compute $\alpha \mod q$ and $\alpha^{-1} \mod q$ (If $\alpha^{-1} \mod q$ does not exist, then $\alpha = 0 \mod q$ and computing σ becomes trivial). In this case, since $\alpha = p^L$ for some $L > 5 \log l = O(\log l)$, computing $\alpha \mod q = p^L \mod q$ can be efficiently done in $O(\log L) = O(\log \log l)$ using repeated squaring. Once this is found, $\alpha^{-1} \mod q$ can also be efficiently found using the Extended Euclidean algorithm. □

5.2.5 Concrete proof sizes

Using batching techniques from [BBF19] (as mentioned in §3.5), we can batch PoE and PoKPE proofs using the PoKCR protocol to obtain a smaller evaluation proof for the Dew-PC protocol in Figures 7 and 8. Since the size of class group elements significantly dwarfs the sizes of field elements, the number of group elements is a good measure of the concrete proof size. Recent work [DGS21] suggests that for 128-bit security (equivalent to 3072-bit RSA), the class group discriminant would have to be around 6784 bits in length, leading to group elements of size 5088 bits (after compression using techniques from [DGS21]).

Each evaluation proof consists of 66 group elements after batching. We believe further optimisations may be possible, and leave it to future work.

6 Fixing the DARK PCS

In this section, we recover the DARK result [BFS20] by showing how to fill the gap in their proof of security. As noted by [BHR+21], the proofs of Lemmas 8 (for instantiation with RSA groups) and 9 (for instantiation with class groups) in the DARK paper are flawed, which means we do not know of an extractor for the DARK commitment scheme. The work of [BHR+21] does not resolve the gap; instead, they give a modified construction.

Here, we define a commitment scheme that is very similar to the DARK scheme and achieves logarithmic proof size and verification time from their paper under the same assumptions, but at the expense of a quadratic prover time. We start from our PCS Dew-PC in Section 5 which is syntactically close to the DARK commitment scheme. In both schemes, the commitment is an encoding of the evaluation of the polynomial at some integer. The major difference between the PCS we present below and the DARK scheme is the point of evaluation q (we use α^2 instead), and its size. In DARK, the lower bound on q is $p^{O(\log l)}$, for a degree l polynomial, while we increase the bound, by setting α to be $p^{O(l)}$, which allows us to prove extraction. However, the increased α also leads to an increased prover time. In addition, we allow for rational opening hints unlike in DARK which only allows for integer opening hints, which is crucial for the fix to Lemmas 8 and 9 from DARK.

As shown in Theorem 5.8, the verifier can be made efficient by a careful choice of α; specifically, an exact power of p, say, $\alpha = p^{2l+\log l+2}$. We also emphasize that simply increasing the bound on q in the DARK construction does not resolve the problem; we additionally need the relaxation provided by our commitment scheme to allow fractions as opening hints to show extraction.

We now define the PCS $\mathsf{PC} = (\mathsf{setup}, \mathsf{commit}, \mathsf{open}, \mathsf{eval})$ as below:

- $\mathsf{setup}(1^\kappa, D)$: $\mathbb{G} \leftarrow\!\$\ \mathsf{GGen}(\kappa), g \leftarrow\!\$\ \mathbb{G}$. Return $\mathsf{pp} = (\lambda, \mathbb{G}, g, p)$, set $\alpha := p^{2D+\log D+2}$.
- $\mathsf{commit}(\mathsf{pp}, D, \mathbf{f}, l)$: If $l \le D$, $C := g^{\sum_{i=0}^{l-1} f_i \alpha^{2i}}$, considering $f_i \in \mathbb{Z}_p$ as integers in the range $[0, p-1]$. Return $(C, \mathbf{f})$.
- $\mathsf{open}(\mathsf{pp}, D, \mathbf{f}, l, C, \tilde{\mathbf{c}})$: Return 1 if all conditions hold, else return 0.
  - $l \le D$
  - $C = g^{\sum_{i=0}^{l-1} \tilde{c}_i \alpha^{2i}}$, $\sum_{i=0}^{l-1} \tilde{c}_i \alpha^{2i} \in \mathbb{Z}$, $\tilde{\mathbf{c}} \in \mathbb{Q}(l, lp^{\log l})^l$,

    where the set $\mathbb{Q}(\beta_1, \beta_2)$ is defined as before:

    $$\mathbb{Q}(\beta_1, \beta_2) := \left\{ \frac{a}{b} : \gcd(a, b) = 1,\ \gcd(b, p) = 1,\ 0 < b < p^{\beta_1},\ |a/b| \le \beta_2 \alpha \right\}$$

  - $\tilde{\mathbf{c}} = \mathbf{f} \mod p$
- eval: The eval protocol is described in Fig. 11.

Note that this is exactly the same PCS as defined in Section B.1, except for eval. We construct eval using the recursive divide and combine strategy along the lines of the DARK construction. Now, by relying on the binding of our commitment scheme, we are able to combine extractor outputs at each level in a valid way which is the heart of the DARK extraction procedure. Furthermore, without the need for PoKPE in eval, we do not need GGM anymore, and can show extraction for PC by rewinding the prover under the same assumptions as DARK.

We will now show a version of Lemma 9 from the DARK paper for the PCS PC and recover the proof of extraction.

Lemma 6.1. *Given commitments C, C', C_L, C_R such that $C = C_L^\gamma C_R$, $C' = C_L^{\gamma'} C_r$ for different $\gamma, \gamma' \in \mathbb{Z}_p$, opening hints $\mathbf{f}, \mathbf{f}' \in \mathbb{Q}(\beta_1, \beta_2)^l$ for C and C', and $v, v' \in \mathbb{Z}_p$ such that $v = f(z) \mod p, v' = f'(z) \mod p$, there exists an extractor that outputs opening hints $\mathbf{f_L}, \mathbf{f_R} \in \mathbb{Q}(2\beta_1 + 1, 2p\beta_2)^l$ for C_L and C_R, and v_L, v_R satisfying $v_L = f_L(z) \mod p, v_R = f_R(z) \mod p$ or a fractional root of g or an element in $\mathbb{G}$ of known order.*

Proof. Using the two equalities, we can write C_L and C_R as

$$C_L = \left(C\,(C')^{-1}\right)^{1/(\gamma-\gamma')}, \quad C_R = \left(C^{\gamma'}\,(C')^{-\gamma}\right)^{1/(\gamma-\gamma')}$$

Now, since we have opening hints for both C and C', we can write the exponents of C_L and C_R as a functions of $\mathbf{f}, \mathbf{f}', \gamma, \gamma'$ (or obtain an element of low order):

$$C_L = g^{\sum_{k=0}^{l-1} \frac{f_k - f'_k}{\gamma-\gamma'}\alpha^{2k}}, \quad C_R = g^{\sum_{k=0}^{l-1} \frac{f'_k\gamma - f_k\gamma'}{\gamma-\gamma'}\alpha^{2k}}$$

Call these coefficients $f_{L,k}$ and $f_{R,k}$ respectively. Then, we can check that $f_{L,k}, f_{R,k} \in \mathbb{Q}(2\beta_1+1, 2p\beta_2)$ for all k. If the exponents of g were not dyadic rationals, we would have a valid fractional root of g, which breaks the Fractional Root Assumption (Assumption 3). □

Protocol eval

Set parameter $\alpha > p^{2(D+\log D)+1}$

$\mathsf{eval}(\mathsf{pp}, D, C, l, z, v, \beta_1, \beta_2; f(X) \in \mathbb{F}_p[X])$:

1. If $l < D$, invoke $\mathsf{evalBounded}(\mathsf{pp}, C, l, z, v, \beta_1, \beta_2; f(X) \in \mathbb{Q}(\beta_1, \beta_2)[X])$, else abort.

$\mathsf{evalBounded}(\mathsf{pp}, C, l, z, v, \beta_1, \beta_2; f(X) \in \mathbb{Q}(\beta_1, \beta_2)[X])$:

if $(l = 0)$ **then**

1. Prover sends $f(X)$ to the Verifier
2. Verifier checks that $f \in \mathbb{Q}(0,1)[X]$
3. Verifier checks that $f \equiv v \bmod p$
4. Verifier checks that $g^f = C$
5. Verifier outputs 1 if all checks pass, and 0 otherwise

end if

if $(l+1)$ is odd **then**

1. $l' = l+1, C' = C^{\alpha^2}, v' = v \cdot x \bmod p, f'(X) = X \cdot f(X)$
2. Invoke $\mathsf{evalBounded}(\mathsf{pp}, C', l', z, v', \beta_1, \beta_2; f'(X))$

else

1. Prover and Verifier compute $l' = \frac{l+1}{2} - 1$
2. Prover computes $f_L(X) = \sum_{i=0}^{l'} f_i X^i, f_R(X) = \sum_{i=0}^{l'} f_{l'+i+1} X^i$
3. Prover computes $v_L = f_L(z) \bmod p, v_R = f_R(z) \bmod p$
4. Prover computes $C_L = g^{f_L(\alpha^2)}, C_R = g^{f_R(\alpha^2)}$
5. Prover sends v_L, v_R, C_L, C_R to the verifier
6. Verifier checks that $v = v_L + v_R \cdot z^{l'+1} \bmod p$, and outputs 0 if it fails
7. Prover and Verifier run $\mathsf{PoE}(C_R, C/C_L, \alpha^{2l'+2})$ to prove that $C = C_L C_R^{(\alpha^{2l'+2})}$
8. Verifier chooses random $\gamma \leftarrow\$ \mathbb{Z}_p$ and sends to the Prover
9. Prover and Verifier compute $v' = \gamma v_L + v_R \bmod p, C' = C_L^{\gamma} C_R, \beta'_1 = 2\beta_1 + 1, \beta'_2 = 2p\beta_2$
10. Prover computes $f'(X) = \gamma f_L(X) + f_R(X)$
11. Invoke $\mathsf{evalBounded}(\mathsf{pp}, C', l', z, v', \beta'_1, \beta'_2; f'(X))$

end if

Figure 11: eval

We can now show extraction by proving a theorem about witness-extended emulation as done in DARK. This gives a correct proof of Theorem 2 from the DARK paper. The proof is along the lines of the proof of Theorem 2 in DARK, except that we invoke Lemma 6.1 (Corrected Lemma 9) to combine the polynomials extracted from child nodes into a polynomial for the next level.

Theorem 6.2. *Let* GGen *generate groups* $\mathbb{G}$ *of unknown order such that the order of* $\mathbb{G}$ *is odd, and such that there exists a PPT algorithm for taking square roots in* $\mathbb{G}$*. The PCS (from Construction 6) for polynomials in* $\mathbb{Z}_p[X]$ *of degree at most* $l = poly(\kappa)$*, instantiated using* $\alpha > p^{2(l+\log l)+1}$ *and* GGen*, has witness-extended emulation if the Adaptive root assumption and the 2-Strong RSA Assumption hold for* GGen*.*

Proof. We will show security by extracting a polynomial $f(X) \in \mathbb{Q}(l, lp^{\log l})[X]$, which is inside the binding space of the commitment scheme. We will also show that $g^{f(\alpha^2)} = C$ and $f(z) \mod p = v$. The

proof will use the general forking lemma to show witness-extended emulation.
In particular, given a transcript with 2 challenges per round (i.e., $< 2(l+1)$ transcripts), we can construct an extractor that outputs either an opening to the commitment C, or an element of known order, or a fractional root of g.

First, we can replace all PoEs by true statements in each transcript. From Lemma 7 in DARK, the resulting eval protocol where the verifier makes the check in step (7) directly instead of invoking a PoE results in an identical eval protocol.
Given a tree of accepting transcripts with branching factor 2 as in the general forking lemma, we will extract a witness at each node of the tree, when given witnesses for each of the node's children. Each level corresponds to an invocation of evalBounded. Denote the inputs to evalBounded with subscripts (superscripts for polynomials) indicating the round: $(C_i, z, v_i, l_i, f^{(i)}(X))$.
In each round, we will extract a polynomial $f^{(i)}(X) \in \mathbb{Q}(\beta_1, \beta_2)[X]$ that is an opening hint for C_i of degree at most l_i and such that $f^i(z) = v_i \mod p$.

Starting from the leaves of the tree, we can extract directly from the transcript the $l_{\log l} = 0$ degree (constant) polynomial such that $f(X) \in \mathbb{Q}(0, 1)$, and $v = f(z) \mod p$ and $g^f = C$.
Now, we show how to compute the witness at every level $i-1$ given witness for level i.

- If $l_i + 1$ is odd: We have that $C_{i-1}^{\alpha^2} = C_i = g^{f^i(\alpha^2)}$, we must have $f^i(\alpha^2) = \frac{f^{i-1}(\alpha^2)}{\alpha^2}$. If this is not an integer, we obtain a fractional root of g.
  If the fraction above is an integer, we have our required polynomial of degree $l_{i-1} = l_i - 1$, such that $C_{i-1} = g^{f^{i-1}(\alpha^2)}$ and $f^{i-1}(z) = z^{-1} \cdot f^i(z) \mod p = z^{-1} \cdot v_i \mod p = v_{i-1} \mod p$.
- If $d_i + 1$ is even: We can call Lemma 6.1 on the children of the node to obtain $f_L^i, f_R^i, y_L^i, y_R^i$. Defining $f^i(X) = f_L(X) + X^{\frac{l_i+1}{2}} f_R(X)$ and $v^i = v_L^i + z^{\frac{l_i+1}{2}} v_R^i \mod p$, we obtain the required properties on f^i.

Note that Lemma 6.1 is called at most $\log l$ times, as there are $\log l$ rounds (when $l_i + 1$ is odd, the bounds on the coefficients do not change, they're just shifted.). Since β_1 and β_2 are defined by the check in the last round (done directly by the verifier), we have that $\beta_1 = 0$ and $\beta_2 = 1$.
Hence, the final opening hints are in $\mathbb{Q}(l, lp^{\log l})[X]$. Theorem 3.2 gives a lower bound on α for this opening hint to be within the binding range of the PCS; $\alpha > p^{2(l+\log l)+1}$ suffices.

□

7 Dew – Transparent zkSNARKs from Dew-PC

As a corollary of our PCS, we get concrete instantiations of new *transparent succinct arguments* by applying the now standard compilation process to compile an information theoretic proof in an idealized model using into a succinct argument using a PCS.

The modular approach advocated for designing efficient arguments consists of two steps; constructing an information theoretic protocol in an abstract model (PCP, linear PCP, IOP etc.), and then compiling the information-theoretic protocol via a cryptographic compiler to obtain an argument system. Many recent constructions of zkSNARKs [BFS20, CHM+20, GWC19] follow this approach where the information theoretic object is an algebraic variant of IOP, and the cryptographic primitive in the compiler is a polynomial commitment scheme. Marlin [CHM+20] uses an IOP abstraction called algebraic holographic proofs (AHP), and [BFS20] uses an abstraction called polynomial IOPs (PIOPs). In both these abstractions, the prover and the verifier interact where the prover provides oracle access to a set of polynomials, and the verifier sends random challenges. Then, the verifier asks for evaluations of these polynomials at these challenge points and decides to accept or reject based on the answers. PLONK [GWC19] uses an abstraction called idealized low degree protocols (ILDPs) that proceeds in a similar way except that at the end of the protocol, the verifier checks a set of polynomial identities over the oracles sent by the prover. Polynomial Holographic IOP (PHP) [CFF+21] specializes the IOP notion in two ways (i) it is holographic – that is, the verifier has access to a set of oracle polynomials created during the setup phase that encode the relation, (ii) the verifier can directly check polynomial identities. The high level idea to build a zkSNARK with universal SRS starting from PIOPs/AHPs/ILDPs/PHPs is the following: the

argument prover commits to the polynomials obtained from the information-theoretic prover, and then uses the evaluation opening property of the polynomial commitment scheme to respond to the evaluation queries of the verifier in a verifiable way.

We present concrete instantiations of zkSNARKs obtained by using our transparent PCS to cryptographically compile the AHP underlying the constructions of Sonic, Marlin and PLONK. The AHPs underlying PLONK and Sonic were given in in [ABC+21]. The choice of abstraction of AHP is not important, and one could use our polynomial commitment scheme to compile similar notions like PIOPs and PHPs as well.

7.1 Hiding Commitments and Zero-Knowledge Evaluation

Our constructions in Sections 3, 4.1, and 5 are not hiding and do not have zero-knowledge evaluation, but there are known techniques from [BFS20] which would allow us to convert these protocols into hiding and zero-knowledge evaluation variants.

We define hiding and HVZK below and show how to make our PCS hiding as well as the evaluation protocol zero knowledge, using the techniques from [BFS20].

Definition 7.1 (Hiding)**.** *A polynomial commitment scheme* PC *is hiding if for all PPT* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*, the probability of distinguishing between commitments of different messages is negligible in* κ*:*

$$\left| 1 - 2 \cdot \Pr\left[\hat{b} = b \ : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{setup}(1^\kappa, D) \\ (f_0(X), f_1(X), d, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pp}) \\ b \xleftarrow{\$} \{0,1\} \\ (C, \tilde{\mathbf{c}}) \leftarrow \mathsf{commit}(\mathsf{pp}, f_b(X), d) \\ \hat{b} \leftarrow \mathcal{A}_1(\mathsf{st}, C) \end{array} \right] \right| \leq \mathsf{negl}(\kappa).$$

Definition 7.2 (HVZK for interactive arguments)**.** *Let* $\mathsf{View}_{\langle P(x,w), V(x)\rangle}$ *denote the view of the verifier in an interactive protocol for some relation* $\mathcal{R}$ *on common input* x *and prover witness input* w*. The interactive protocol has* δ*-statistical honest verifier zero-knowledge if there exists a probabilistic polynomial time algorithm* $\mathcal{S}$ *such that for every* $(x, w) \in \mathcal{R}$*, the distribution* $\mathcal{S}(x)$ *is* δ*-close to* $\mathsf{View}_{\langle P(x,w), V(x)\rangle}$ *(as distributions over the randomness of* P *and* V*).*

Hiding Commitments: The PCS can be converted into variants with hiding commitments by adding a degree $d+1$ term with a large random coefficient. This technique is proven to give a statistically hiding polynomial commitment scheme if $\langle g \rangle = \langle g^{\alpha^{2d+2}} \rangle$ but [BFS20] requires a subgroup indistinguishability assumption on the group of unknown order to prove this claim.

Formally, the commitment algorithm has to be modified as follows:

- $\mathsf{commitH}(\mathsf{pp}, f(X), d) \to (C, \tilde{\mathbf{c}})$: Where $f(X)$ is lifted to $\hat{f}(X) \in [0, p-1]$ and then a random integer r is sampled as $r \xleftarrow{\$} [0, B \cdot 2^\kappa)$. Now compute $\tilde{f} = \hat{f}(X) + r \cdot X^{d+1}$ and return $(C \leftarrow g^{\tilde{f}(\alpha^2)}, \tilde{f})$.

Theorem 7.3 (Proof of Statistical Hiding)**.** *The commitment scheme* $\Gamma = (\mathsf{setup}, \mathsf{commitH}, \mathsf{open})$ *is statistically hiding if* $B \gg |\mathbb{G}|$ *and if* $\langle g \rangle = \langle g^{\alpha^{2d+2}} \rangle$*. It is binding if the commitment described in Sections 3, 4.1, B.1 and 5 are binding.*

Proof. The hiding commitment is a commitment to a degree $d+1$ polynomial. So it directly inherits the binding property from the non-hiding scheme.

To show hiding, we will use the fact that the uniform distributions $[0, b]$ and $[a, a+b]$ have statistical distance $\frac{a}{b}$, i.e., the probability that any algorithm can distinguish the distributions from a single sample is less than $\frac{a}{b}$. Similarly $C \leftarrow g^{\hat{f}(\alpha^2) + r \cdot \alpha^{2d+2}}$ for $r \xleftarrow{\$} [0, B \cdot 2^\kappa)$ has statistical distance at most $2^{-\kappa}$ from a uniform element generated by g if $B \geq |\langle g \rangle|$. This means that two polynomial commitments can be distinguished by any algorithm with probability at most $2^{-\kappa+1}$.

□

$\mathsf{evalZK}(\mathsf{pp}, C, d, x, v; f(X), r)$

1 : $\mathcal{P}$ samples a random degree d $k(X) \xleftarrow{\$} [0, (p-1)^2 \cdot 2^\kappa][X]$ and $r_k \xleftarrow{\$} [0, B \cdot 2^\kappa)$ and computes

$R \leftarrow g^{k(\alpha^2)+r_k \cdot \alpha^{2d+2}}$ and $y_k \leftarrow k(z) \mod p$

2 : $\mathcal{P}$ sends R and y_k to $\mathcal{V}$

3 : $\mathcal{V}$ samples random $c \xleftarrow{\$} [0, p-1]$ and sends it to $\mathcal{P}$

4 : $\mathcal{P}$ computes $s(X) \leftarrow k(X) + c \cdot f(X)$, as well as $r_s \leftarrow r_k + c \cdot r$.

5 : $\mathcal{P}$ sends r_s to $\mathcal{V}$

6 : $\mathcal{P}$ and $\mathcal{V}$ compute $C_s \leftarrow R \cdot C^c \cdot g^{-\alpha^{2d+2} \cdot r_s}$

and $y_s \leftarrow y_k + c \cdot y \mod p$ // $C_s = g^{s(\alpha^2)}$

7 : $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{eval}(\mathsf{pp}, C_s, d, z, y; s(X))$ // $s(z) \mod p = y_s$

Figure 12: Protocol for evalZK

Unfortunately, in a setting without a trusted setup, $g^{\alpha^{2d+2}}$ might only generate a subgroup of $\langle g \rangle$. The commitment then becomes computationally hiding under a Subgroup Indistinguishability Assumption [BG10]: the precise assumption is that no efficient adversary can distinguish a random element of $\langle h \rangle$ from $\langle g \rangle$ for any non-trivial $h \in \langle g \rangle$.

Theorem 7.4 (Proof of Computational Hiding)**.** *The commitment scheme* $\Gamma = (\mathsf{setup}, \mathsf{commitH}, \mathsf{open})$ *is computationally hiding if* $B \gg |\mathbb{G}|$ *and if the Subgroup Indistinguishability Assumption [BG10] holds. It is binding if the commitment described in Sections 3, 4.1, B.1 and 5 are binding.*

Proof. The proof of binding remains as above.

The proof of computational hiding is as follows. Suppose there exists a non-uniform distinguisher $D_{f,g}$ that can distinguish freshly generated commitments to f and g with non-negligible probability, then a non-uniform adversary $\mathcal{A}_{f,g}$ may be constructed as follows: upon receipt of g' sampled either from $\langle g \rangle$ or from $\langle g^{\alpha^{2d+2}} \rangle$, it sends $g^{f(\alpha^2)}g'$ and $g^{g(\alpha^2)}g'$ to the distinguisher $D_{f,g}$ and returns its answer. In the case that g' was sampled from $\langle g^{\alpha^{2d+2}} \rangle$ this is a statistical simulation of a pair of commitments to f and g respectively, hence the distinguisher will succeed with non-negligible probability.

In the case that g' was sampled from $\langle g \rangle$, the pair is actually statistically indistinguishable, and thus the distinguisher must fail. Thus, $\mathcal{A}_{f,g}$ is able to distinguish from which group g' was sampled with non-negligible probability, contradicting the Subgroup Indistinguishability Assumption.

□

Zero-Knowledge Evaluation: Let $f(X)$ be the committed polynomial, using the hiding commitment scheme. The prover wants to convince the verifier that $f(z) \mod p = y$. To do this the prover commits to a degree d polynomial $k(X)$ with random coefficients. The prover also reveals $y' \leftarrow k(z) \mod p$. The verifier then sends a random challenge c and the prover and verifier can compute a commitment to $s(X) \leftarrow k(X) + c \cdot f(X)$. The random polynomial $k(X)$ ensures that $s(X)$ is distributed statistically close to a random polynomial. The verifier must now check that $s(z) \mod p = y' + c \cdot y \mod p$. Instead of sending $s(X)$ in the clear, the prover just sends the commitment randomness to provide the verifier with a non-hiding commitment to $s(X)$. The prover and verifier can then use the standard Eval protocol to efficiently evaluate s at z.

Theorem 7.5 (Proof of Zero-Knowledge Evaluation)**.** *Let* eval *have perfect completeness and extractability. Assuming that* commitH *is statistically hiding and both the order assumption and the strong RSA assumption hold for* GGen *the protocol EvalZK has perfect completeness, extractability and* δ*-statistical honest-verifier zero-knowledge for* $\delta \leq (d+1) \cdot 2^{-\kappa}$.

Proof. The proof of completeness is straightforward.

We can see that the protocol maintains witness extended emulation, as the extractor can use two transcripts with different c, c' along with two calls to the extractor of the original eval protocol to extract $f(X)$ in evalZK or break an assumption from these two transcripts, in the same way as in Theorem 3.13. This might lead to a slight growth in the bound on α for binding to hold. However the final bound on α is that required for extractability, which is usually higher than the bound required for binding, and so the final bound will usually remain unaffected.

We can construct the simulator $\mathcal{S}$ to show zero-knowledge as follows. Start with a polynomial $s(X) \xleftarrow{\$} [0, (p-1)^2 \cdot 2^\kappa][X]$ with uniform random coefficients, and a blinding factor $r_s \xleftarrow{\$} [0, B \cdot p \cdot 2^\kappa)$. The simulator $\mathcal{S}$ then chooses a random challenge $c \xleftarrow{\$} [0, p-1]$ and computes $R = g^{s(\alpha^2)+r_s\alpha^{2d+2}} \cdot C^{-c}$. The simulator then performs the rest of the Eval protocol honestly using $s(X)$ as the witness.

The simulator's r_s is distributed identically to the honest r_s. So by the hiding property of the commitment scheme, R is statistically indistinguishable from any other commitment. The simulated and the honest $s(X)$ have statistical distance at most $2^{-\kappa}$ from a random polynomial. The coefficients of $c \cdot f(x)$ are in $[0, (p-1)^2]$, and the coefficients of the blinding polynomial $k(X)$ are sampled from a range that is larger by a factor 2^κ. So the distribution of coefficients of $s(X) = k(X) + c \cdot f(X)$ is at a statistical distance at most $2^{-\kappa}$ away from the uniform distribution over $[0, (p-1)^2 \cdot 2^\kappa]$. Since the distributions of simulated and real coefficients have a statistical distance of at most $2^{-\kappa}$, the statistical distance between the simulated $s(X)$ and the real $s(X)$ is at most $(d+1) \cdot 2^{-\kappa}$. The evaluation with Eval cannot leak more than $s(X)$ itself. Therefore the views of the simulated and real transcripts are δ-close with $\delta \leq (d+1) \cdot 2^{-\kappa}$. Thus, the protocol is δ-statistically honest verifier zero-knowledge.

□

7.2 Algebraic Holographic Proof

Definition 7.6 (AHP [CHM+20])**.** *An* Algebraic Holographic Proof (AHP) *over a field family $\mathcal{F}$ for an indexed relation $\mathcal{R}$ is specified by the following tuple:*

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

where $\mathsf{k}, \mathsf{s}, \mathsf{d} : \{0,1\}^* \to \mathbb{N}$ are polynomial-time computable functions; $\mathcal{I}, \mathcal{P}, \mathcal{V}$ are the indexer*,* prover*, and* verifier *algorithms; k specifies the number of rounds, s specifies the number of polynomials in each round, and d specifies degree bounds on these polynomials. The protocol proceeds as follows:*

- ***Offline phase*** *The indexer $\mathcal{I}$ receives as input a field $\mathbb{F} \in \mathcal{F}$, index i for $\mathcal{R}$, and outputs $\mathsf{s}(0)$ polynomials $p_{0,1}, \ldots, p_{0,\mathsf{s}(0)} \in \mathbb{F}[X]$ of degrees at most $\mathsf{d}(|\mathsf{i}|, 0, 1), \ldots, \mathsf{d}(|\mathsf{i}|, 0, \mathsf{s}(0))$ respectively. The offline phase does not depend on the instance or witness, and simply consists of encoding the given index i.*
- ***Online phase*** *The prover $\mathcal{P}$ receives $(\mathbb{F}, \mathsf{i}, \mathsf{x}, \mathsf{w})$, for an instance x and witness w such that $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$. The verifier $\mathcal{V}$ receives $(\mathbb{F}, \mathsf{x})$ and oracle access to the polynomials output by $\mathcal{I}(\mathbb{F}, \mathsf{i})$. The prover $\mathcal{P}$ and the verifier $\mathcal{V}$ interact over $\mathsf{k} = \mathsf{k}(|\mathsf{i}|)$ rounds. In the i-th round, $i \in [\mathsf{k}]$, the verifier $\mathcal{V}$ sends a message $\rho_i \in \mathbb{F}^*$ to the prover $\mathcal{P}$; the prover $\mathcal{P}$ responds with $\mathsf{s}(i)$ oracle polynomials $p_{i,1}, \ldots, p_{i,\mathsf{s}(i)} \in \mathbb{F}[X]$. After k rounds, the verifer outputs additional randomness $\rho_{\mathsf{k}+1} \in \mathbb{F}^*$ which is an auxiliary input to $\mathcal{V}$ in subsequent phases.*
- ***Query phase*** *Let $\mathbf{p} = (p_{i,j})_{i \in [\mathsf{k}], j \in [\mathsf{s}(i)]}$ be a vector consisting of all the polynomials sent by the prover $\mathcal{P}$. The verifier can query any of the polynomials it has received any number of times. Concretely, $\mathcal{V}$ executes a subroutine $\mathcal{Q}_\mathcal{V}$ that receives $(\mathbb{F}, \mathsf{x}; \rho_1, \ldots, \rho_{\mathsf{k}+1})$ and outputs a query set Q consisting of tuples $((i,j), z)$ to be interpreted as "query $p_{i,j}$ at $z \in \mathbb{F}$". We denote a vector consisting of query answers by $\mathbf{p}(Q)$.*
- ***Decision phase*** *The verifier outputs "accept" or "reject" based on the answers to the queries and the verifier's randomness. Concretely, $\mathcal{V}$ executes a subroutine $\mathcal{D}_\mathcal{V}$ that receives $(\mathbb{F}, \mathsf{x}, \mathbf{p}(Q); \rho_1, \ldots, \rho_{\mathsf{k}+1})$ as input, and outputs a decision bit.*

The function d *determines which provers to consider for the completeness and soundness properties of the proof system. A (potentially malicious) prover* $\tilde{\mathcal{P}}$ *is considered* admissible *for* AHP *if, in interaction with the verifier* $\mathcal{V}$*, it holds that for every round* $i \in [\mathsf{k}]$ *and oracle index* $j \in [s(i)]$ *we have* $\deg(p_{i,j}) \leq \mathsf{d}(|\mathsf{i}|, i, j)$*. The honest prover* $\mathcal{P}$ *is required to be admissible under this definition. An AHP satisfies completeness, knowledge soundness and zero-knowledge as defined below.*

- *Completeness: An* AHP *is complete if for all* $\mathbb{F} \in \mathcal{F}$ *and any* $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$*, the decision bit returned by* $\mathcal{V}^{\mathcal{I}(\mathbb{F},\mathsf{i})}(\mathbb{F}, \mathsf{x})$ *after interacting with an honest* $\mathcal{P}(\mathbb{F}, \mathsf{i}, \mathsf{x}, \mathsf{w})$ *is* 1*.*
- *Knowledge soundness: An* AHP *is* ϵ*-knowledge-sound if there exists a polynomial-time knowledge extractor* Ext *such that for any admissible prover* $\mathcal{P}^*$*, field* $\mathbb{F} \in \mathcal{F}$*, relation* i*, instance* x *and auxiliary input* z*:*

$$\Pr\left[(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R} : \mathsf{w} \leftarrow \mathsf{Ext}^{\mathcal{P}^*}(\mathbb{F}, \mathsf{i}, \mathsf{x}, z)\right] \geq \Pr[\langle \mathcal{P}^*(\mathbb{F}, \mathsf{i}, \mathsf{x}, z), \mathcal{V}^{\mathcal{I}(\mathbb{F},\mathsf{i})}(\mathbb{F}, \mathsf{x})\rangle = 1] - \epsilon$$

  where Ext *has oracle access to* $\mathcal{P}^*$*, i.e., it can query the next message function of* $\mathcal{P}^*$*, rewind it to obtain all the messages and polynomials returned by it.*
- *Zero-knowledge: Let* b *be a bound on the number of evaluation queries made by* $\mathcal{V}$*. Let* C *be a circuit that tests that the evaluation queries are admissible. An AHP is zero-knowledge if there exists an efficient simulator that can interact with a verifier and can effectively simulate when the verifier makes* b *queries, and these queries satisfy an admissible test that is efficiently checked by the circuit* C*. An algorithm is called* (b, C)*-query if it makes at most* b *queries to the oracles it has access to, and each query individually leads the checker* C *to output* 1*. Formally,* AHP *is* (b, C)*-zero-knowledge if there exists a probabilistic polynomial time simulator* Sim *such that for every field* $\mathbb{F} \in \mathcal{F}$*, every* $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$*, and* (b, C)*-query algorithm* $\mathcal{V}^*$*, the following two random variables are identical:*

  1. $\mathsf{View}_{\langle \mathcal{P}(\mathbb{F},\mathsf{i},\mathsf{x},\mathsf{w}), \mathcal{V}^*(x)\rangle}$*, which is the view of the verifier, given by* $(r, a_1, \ldots, a_q)$ *where* r *is the random tape of* $\mathcal{V}$ *and* $a_1, \ldots, a_q$ *are the responses to* $\mathcal{V}$*'s queries from the oracles sent by* $\mathcal{P}$*.*
  2. $\mathsf{Sim}^{\mathcal{V}^*}(\mathbb{F}, \mathsf{i}, \mathsf{x})$ *which is the output of the simulator, appended to the random tape* r *of the verifier.* Sim *can interact with* $\mathcal{V}$ *and answer* $\mathcal{V}$*'s oracle queries, without rewinding* $\mathcal{V}$*.*

7.3 Transparent preprocessing SNARK with universal SRS

We obtain concrete instantiations of new transparent succinct arguments by applying our polynomial commitment scheme from Sec. 5 to compile different AHPs like Sonic, PLONK, Marlin. Then, applying the Fiat–Shamir transformation to the resulting public-coin preprocessing argument yields a preprocessing zkSNARK with universal SRS in the ROM.

Using the compilation process in [BFS20, CHM+20], $\mathsf{AHP} = (\mathcal{I}, \mathcal{P}, \mathcal{V})$ for $\mathcal{R}$ can be turned into a *preprocessing argument system with universal SRS*, denoted by $\mathsf{AoK} = (\mathsf{S}, \mathsf{I}, \mathsf{P}, \mathsf{V})$ for $\mathcal{R}$. We note that our PCS indeed has the property that the extracted polynomial satisfies a prescribed degree bound D as shown in the proof of extractability. In addition, our protocol can also enforce different degree bounds d_i for different polynomials as long as $d_i \leq D$. This guarantee suffices for compilation of AHP into a succinct argument. In the reduction, a cheating argument prover can be used to construct an admissible AHP prover – the extractor that outputs polynomials within a prescribed degree bound guarantees that the AHP prover is admissible. We outline the compilation process of [CHM+20] in Fig. 13.

Theorem 7.7 ([CHM+20])**.** *Let* $\mathcal{F}$ *be a field family and* $\mathcal{R}$ *be an indexed relation. Consider the following components:*

- $\mathsf{AHP} = (\mathsf{k}, \mathsf{s}, \mathsf{d}, \mathcal{I}, \mathcal{P}, \mathcal{V})$ *is a knowledge sound AHP for* $\mathcal{R}$
- $\mathsf{PC} = (\mathsf{setup}, \mathsf{commit}, \mathsf{open}, \mathsf{eval})$ *is a succinct polynomial commitment over* $\mathcal{F}$ *with binding and extractability*

Then, the construction $\mathsf{AoK} = (\mathsf{S}, \mathsf{I}, \mathsf{P}, \mathsf{V})$ *is a preprocessing argument system for the relation* $\mathcal{R}$*, and satisfies the following properties:*

Compiler from AHP to AoK

Let $\mathsf{AHP} = (\mathcal{I}, \mathcal{P}, \mathcal{V})$ be an AHP for $\mathcal{R}$. Let $\mathsf{PC} = (\mathsf{setup}, \mathsf{commit}, \mathsf{open}, \mathsf{eval})$ be a polynomial commitment scheme. Construct an argument system $\mathsf{AoK} = (\mathsf{S}, \mathsf{I}, \mathsf{P}, \mathsf{V})$ as follows:

- Setup $\mathsf{S}(1^\kappa)$: Run $\mathsf{pp} \leftarrow \mathsf{setup}(1^\kappa)$ and output $\mathsf{srs} := \mathsf{pp}$.
- Preprocessing I: On input i, I runs the AHP indexer $\mathcal{I}$ to obtain polynomials $(p_{0,j})_{j=1}^{\mathsf{s}(0)} \in \mathbb{F}[X]$, and computes commitments to these polynomials; $(c_{0,j})_{j=1}^{\mathsf{s}(0)} = \mathsf{commit}((p_{0,j})_{j=1}^{\mathsf{s}(0)})$. The indexer I outputs $\mathsf{ipk} := ((p_{0,j})_{j=1}^{\mathsf{s}(0)}, (c_{0,j})_{j=1}^{\mathsf{s}(0)})$ and $\mathsf{ivk} := ((c_{0,j})_{j=1}^{\mathsf{s}(0)})$.
- Prover P and Verifier V: For every round $i \in [\mathsf{k}]$, P and V simulate the interaction between the AHP prover $\mathcal{P}(\mathbb{F}, \mathsf{i}, \mathsf{x}, \mathsf{w})$ and verifier $\mathcal{V}(\mathbb{F}, \mathsf{i}, \mathsf{x})$. In the i-th round:
  1. V talks to P and internally runs $\mathcal{V}$. V receives $\rho_i \in \mathbb{F}$ from $\mathcal{V}$, and forwards it to P.
  2. P interacts with V and internally runs $\mathcal{P}$. P forwards the received ρ_i to $\mathcal{P}$, and receives $\mathsf{s}(i)$ polynomials $p_{i,1}, \ldots, p_{i,\mathsf{s}(i)} \in \mathbb{F}[X]$. P invokes the polynomial commitment scheme to commit to each of these polynomials.
  $$C_{i,j} = \mathsf{commit}(p_{i,j}), \text{ for } j = 1, \ldots, \mathsf{s}$$
  P sends the vector of commitments $\mathbf{C} = \{C_{i,j}\}$ to V.
- P and V simulate the query phase of the AHP.
  1. Let $\mathbf{p} = (p_{i,j})_{i \in [\mathsf{k}], j \in [\mathsf{s}(i)]}$ denote the vector consisting of all the polynomials sent by $\mathcal{P}$, and $\mathbf{C}$ the vector of commitments to $\mathbf{p}$. $\mathcal{V}$ executes $\mathcal{Q}_\mathcal{V}(\mathbb{F}, \mathsf{x}; \rho_1, \ldots, \rho_{\mathsf{k}+1})$ and outputs a query set Q consisting of tuples $((i,j), z)$. V forwards the set of query points $((i,j), z)$ to P.
  2. P sends $\mathbf{v}$, a vector of all the claimed evaluations for query points in Q. P and V run $\mathsf{eval}(\mathsf{srs}, \mathbf{C}, d, \mathbf{z}, \mathbf{v}; \mathbf{p})$ where $\mathbf{z}$ is the vector of all query points in Q. In the above, eval is the *batched* evaluation protocol of the polynomial commitment scheme that proves the evaluation of multiple polynomial commitments.
- Decision phase:
  1. V runs $\mathcal{V}$'s decision algorithm $\mathcal{D}_\mathcal{V}(\mathbb{F}, \mathsf{x}, \mathbf{v}; \rho_1, \ldots, \rho_{\mathsf{k}+1})$ to receive a decision bit b_1.
  2. Let b_2 be the decision bit of V from the execution of eval.
  3. V accepts if both b_1 and b_2 are 1.

Figure 13: Compiler from [CHM+20]

- *Preprocessing complexity: Indexer time of* AHP *plus time to commit to* $\mathsf{s}(0)$ *polynomials in* PC.
- *Communication complexity:* $O_\kappa(q)$ *bits where* q *is the query complexity of* AHP.
- *Verification complexity: Verifier time of* AHP *plus time to batch verify* q *evaluations in* PC.
- *Prover complexity: Prover time of* AHP *plus time to commit to* $\sum_{i=1}^{\mathsf{k}} \mathsf{s}(i)$ *polynomials in* PC, *plus the time to produce batch evaluation proofs for* q *queries in* PC.

In the above theorem, if PC is hiding with zero-knowledge evaluation protocol, and if AHP is (b, C)-zero-knowledge where b is the query complexity of AHP and C is a polynomial-time query checker, then AoK is zero knowledge.

7.4 Concrete Instantiations

Theorem 7.8 (PLONK [GWC19])**.** *There exists a* 3*-round AHP for NP relation* $\mathcal{R}$*, where the verifier makes* 12 *queries to* 11 *polynomials at* 2 *distinct points. The indexer complexity is* $O(n)$ *to preprocess* 6 *univariate polynomials of degree* n*, where* n *is the arithmetic circuit complexity of* $\mathcal{R}$*.*

Corollary 1 (Succinct Transparent AoK)**.** *For an* NP *relation with arithmetic complexity* n*, construction 13 instantiated with our PCS with constant proof size and logarithmic verification* Dew-PC *(Section 5) and PLONK's AHP is a* $O(1)$ *round interactive argument of knowledge as per Definition 2.10 in the GGM and satisfies the following properties:*

- *Preprocessing complexity:* $\tilde{O}(n)$
- *Communication complexity:* $O_\kappa(1)$

- *Verification complexity:* $O(\log n)$
- *Prover complexity:* $O(n^2)$

Theorem 7.9 (Marlin [CHM+20]). *There exists a* 4*-round AHP for NP relation* $\mathcal{R}$ *that makes* 21 *queries at* 3 *distinct query points to* 21 *polynomials. The indexer complexity is* $O(n)$ *to preprocess* 9 *univariate polynomials of degree* n*, where* n *is the R1CS complexity of* $\mathcal{R}$*.*

Corollary 2 (Succinct Transparent AoK). *For an* NP *relation with R1CS complexity* n*, construction 13 instantiated with our PCS with constant proof size and logarithmic verification* Dew-PC *(Section 5), and Marlin's AHP is a* $O(1)$ *round interactive argument of knowledge as per Definition 2.10 in the GGM, and satisfies the following properties:*

- *Preprocessing complexity:* $\tilde{O}(n)$
- *Communication complexity:* $O_\kappa(1)$
- *Verification complexity:* $O(\log n)$
- *Prover complexity:* $O(n^2)$

We compare the SNARKs resulting from our PCS with different succinct arguments in Figure 14, and give an estimate of SNARK proof sizes resulting from our PCS. As estimated in § 5.2.5, our PCS evaluation proofs use 66 group elements and given our batching techniques in § 3.5, we can combine all queries to all polynomials into a single evaluation proof. This adds an overhead of only the number of polynomials given by the AHPs. Therefore, the SNARK from Corollary 1 has a proof size of 77 group elements (66 group elements for a batched evaluation proof plus 11 groups elements for committing to 11 polynomials). Similarly, the SNARK from Corollary 2 has a proof size of 87 group elements. The proof also includes a constant number of field elements. However, since each field element is orders of magnitude smaller than a group element, we ignore them in our size estimates.

Protocol	**Setup**	$\lvert\mathsf{pp}\rvert$	**Verifier**	**Proof size**	**Assumption**
Groth16 [Gro16]	private	$O(n)$	$O(1)$	$O(1)$	GGM
Basilisk [RZ21b]	updatable	$O(n)$	$O(\log n)$	$O(1)$	AGM, q-DL
STARK [BBHR19]	public	$O(1)$	$O(\log^2 n)$	$O(\log^2 n)$	CRHF
BulletProofs [BBB+18]	public	$O(n)$	$O(n)$	$O(\log n)$	DL
Block *et al.* [BHR+21]	public	$O(1)$	$O(\log^k n)$	$O(\log^k n)$	HO
CS Proof (VC compiler) [BBF19]	public	$O(1)$	$O(\log n)$	$O(1)$	GGM
Protocols in this work:					
DARK-Fix (§6)	public	$O(1)$	$O(\log n)$	$O(\log n)$	AR, S-RSA
Dew (from Dew-PC in §5)	public	$O(1)$	$O(\log n)$	$O(1)$	GGM

Figure 14: Comparison between the SNARKs resulting from this work and other zero-knowledge succinct argument protocols. Public setup indicates that the SNARK is transparent. We omit the prover complexity in the table since our focus is on verifier and communication complexity. Dew has $\tilde{O}(n)$ prover complexity after preprocessing, while the construction from §6 incurs $\tilde{O}(n^2)$ prover complexity. In the table, n denotes the size of the NP statement. k is some constant. The security parameter λ is omitted for clarity. We compare with Basilisk as it is one of the state-of-the-art SNARKs with private but updatable setup. The asymptotics of Supersonic, as presented in [BFS20] using the broken DARK polynomial commitment scheme, are recovered using our scheme (as shown in Section 6). AR = Adaptive Root, S-RSA = Strong RSA, HO = Hidden Order, CR = Collision Resistant Hash Function.

References

[ABC+21] Diego F. Aranha, Emil Madsen Bennedsen, Matteo Campanelli, Chaya Ganesh, Claudio Orlandi, and Akira Takahashi. Eclipse: Enhanced compiling method for pedersen-committed zksnark engines. Cryptology ePrint Archive, Report 2021/934, 2021. https://ia.cr/2021/934.

[ALM+92] S Arora, C Lund, R Motwani, M Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334, 2018.

[BBF19] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 561–586. Springer, Heidelberg, August 2019.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 701–732. Springer, Heidelberg, August 2019.

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

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

[BF22] Benedikt Bünz and Ben Fisch. Schwartz-zippel for multilinear polynomials mod n. Cryptology ePrint Archive, Report 2022/458, 2022. https://ia.cr/2022/458.

[BFLS91] László Babai, Lance Fortnow, Leonid A Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 21–32, 1991.

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

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 1–20. Springer, Heidelberg, August 2010.

[BH01] Johannes Buchmann and Safuat Hamdy. A survey on iq cryptography. In *In Proceedings of Public Key Cryptography and Computational Number Theory*, pages 1–15, 2001.

[BHR+21] Alexander R. Block, Justin Holmgren, Alon Rosen, Ron D. Rothblum, and Pratik Soni. Time- and space-efficient arguments from groups of unknown order. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 123–152, Virtual Event, August 2021. Springer, Heidelberg.

[Bol04] Béla Bollobás. *Extremal Graph Theory*. Reprint of the 1978 original. Dover Publications, Inc., Mineola, NY, ISBN: 0-486-43596-2, 2004.

[BS96] Wieb Bosma and Peter Stevenhagen. On the computation of quadratic 2-class groups. *Journal de Théorie des Nombres de Bordeaux*, 8(2):283–313, 1996.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.

[CFF+21] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zksnarks and commit-and-prove extensions. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 3–33, Cham, 2021. Springer International Publishing.

[CFKS22] Hien Chu, Dario Fiore, Dimitris Kolonelos, and Dominique Schröder. Inner product functional commitments with constant-size public parameters and openings. Cryptology ePrint Archive, Report 2022/524, 2022. https://ia.cr/2022/524.

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

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001.

[DGS21] Samuel Dobson, Steven Galbraith, and Benjamin Smith. Trustless unknown-order groups. *Mathematical Cryptology*, 2021.

[DK02] Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 256–271. Springer, Heidelberg, April / May 2002.

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

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

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4):205–214, August 1998.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 113–122. ACM Press, May 2008.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th FOCS*, pages 174–187. IEEE Computer Society Press, October 1986.

[Gro05] Jens Groth. Non-interactive zero-knowledge arguments for voting. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 467–482. Springer, Heidelberg, June 2005.

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

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

[GVW02] Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Comput. Complex.*, 11(1/2):1–53, June 2002.

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

[Kil92a] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732, 1992.

[Kil92b] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

[KM03] Stephen Kwek and Kurt Mehlhorn. Optimal search for rationals. *Information Processing Letters*, 86(1):23–26, 2003.

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

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

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

[LM19] Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 530–560. Springer, Heidelberg, August 2019.

[LRY16] Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016*, volume 55 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl, July 2016.

[Mic94a] Silvio Micali. Cs proofs. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 436–453. IEEE, 1994.

[Mic94b] Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

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

[Pie18] Krzysztof Pietrzak. Simple Verifiable Delay Functions. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 60:1–60:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Ros16] Lisa Rosenfeld. The box problem in two and higher dimensions. Bachelor's thesis, University of Rochester, 2016. https://www.sas.rochester.edu/mth/undergraduate/honorspaperspdfs/rosenfeldhonorsthesis16.pdf.

[RZ21a] Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 774–804, Virtual Event, August 2021. Springer, Heidelberg.

[RZ21b] Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable snarks. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 774–804, Cham, 2021. Springer International Publishing.

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

[Wee05] Hoeteck Wee. On round-efficient argument systems. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 140–152. Springer, Heidelberg, July 2005.

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 379–407. Springer, Heidelberg, May 2019.

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

[Zca] Zcash. https://z.cash.

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

Appendix

A Proofs of Auxiliary Lemmas

Lemma A.1 (Lemma 3.6 restated)**.** . *Given a group G and $g \in G$, if there exist group elements $W, X_1, \ldots, X_m$ such that $W = \prod_{i=1}^{m} X_i$ and a prover succeeds in convincing a* PoKPE *verifier for all of these elements w.r.t. the base g, either*

1. *the prover knows exponents $w, x_1, \ldots, x_m$ such that $w, x_i > 0$ for all i and $w = \sum_{i=1}^{m} X_i$.* OR
2. *the prover knows (a multiple of) the order of g, and can break the Order Assumption (Assumption 1).*

Proof. Since the verifier accepts all the PoKPE proofs, there exists an extractor (that uses the PoKPE extractor) that outputs positive $w, x_1, \ldots, x_m$ as exponents to $W, X_1, \ldots, X_m$ with base g respectively. Construct an adversary $\mathcal{A}$ for the Order assumption that uses the extractor to obtain these exponents and computes $k := w - \sum_{i=1}^{m} x_i$. If this is non-zero, then notice that $g^k = \frac{W}{\prod_{i=1}^{m} X_i} = 1$, which the adversary can use to break the Order assumption. □

Lemma A.2 (Lemma 3.7 restated)**.** *Suppose $K = \sum_{i=0}^{k} M_i \alpha^i$ where M_i's are not necessarily $< \alpha$, but we have a bound $M_i < \alpha(\alpha - 1)\ \forall\, i$. Then, we can write $K = \sum_{i=0}^{k+1} U_i \alpha^i$ where each $U_i < \alpha$, and*

$$U_i := \begin{cases} (M_0 \mod \alpha + u_0) \mod \alpha & \text{if } i = 0 \\ (M_i \mod \alpha + \left\lfloor \frac{M_{i-1}}{\alpha} \right\rfloor + u_i) \mod \alpha & \text{if } 1 \le i \le k \\ (\left\lfloor \frac{M_k}{\alpha} \right\rfloor + u_{k+1}) & \text{if } i = k+1 \end{cases}$$

$$u_i := \begin{cases} 0 & \text{if } i = 0, 1 \\ \left\lfloor \frac{M_{i-1} \mod \alpha + \left\lfloor \frac{M_{i-2}}{\alpha} \right\rfloor + u_{i-1}}{\alpha} \right\rfloor & \text{if } 1 \le i \le k+1 \end{cases}$$

Also, $\forall\, i \;\; u_i \in \{0, 1\}$.

Proof. We first prove the last part: $u_i \in \{0, 1\}$ by induction. We already have $u_0 = u_1 = 0$. Let $u_{i-1} \in \{0, 1\}$. Then

$$\begin{aligned} u_i &:= \left\lfloor \frac{M_{i-1} \mod \alpha + \left\lfloor \frac{M_{i-2}}{\alpha} \right\rfloor + u_{i-1}}{\alpha} \right\rfloor \\ &\le \left\lfloor \frac{\alpha + \alpha - 2 + 1}{\alpha} \right\rfloor \\ &< 2 \end{aligned}$$

Since $u_i \in \mathbb{Z}^+$, the conclusion holds.
Also, notice that

$$\begin{aligned} U_{k+1} &= \left\lfloor \frac{M_k}{\alpha} \right\rfloor + u_{k+1} \\ &< (\alpha - 1) + 1 \\ &< \alpha \end{aligned}$$

Hence, $U_{k+1} < \alpha$, and the other U_i are already $< \alpha$ due to them being computed $\mod \alpha$.
The equality between the sums follows by telescoping and the repeated use of the identity

$$a \mod b = a - b \left\lfloor \frac{a}{b} \right\rfloor$$

□

Lemma A.3 (Lemma 3.8 restated)**.** *Suppose for some α, $M'\alpha - N' = M\alpha - N$, where $M', N' \in \mathbb{Q}$ and $M, N \in \mathbb{Z}$.*
If $|N|, |N'| < B$, and $M' = \frac{x}{y}$ for the smallest possible y and $y < \frac{\alpha}{2B}$, then $M' = M$ and $N' = N$.

Proof. Consider the equivalent equation $\alpha(M' - M) = N - N'$. Clearly, $|N - N'| < 2B$. WLOG let $M' > M$. Then, least positive value $(M' - M)$ can take is $\frac{1}{y}$, as M is an integer
But if $M' \neq M$, then

$$\alpha(M' - M) > \frac{\alpha}{y} > 2B > |N - N'|$$

for any N, N', which is a contradiction. Hence, $M = M', N = N'$. □

Lemma A.4 (Lemma 3.9 restated)**.** *Given $k, p \in \mathbb{Z}$ and $n < \frac{p}{2}$, then any polynomial $f(x_1, \ldots, x_n) \coloneqq \sum_{i=1}^{n} m_i x_i$ is such that for all $c \in \mathbb{Z}_p$ and $b = GCD(m_1, \ldots, m_n, k)$,*

$$\Pr_{x_i \leftarrow\$ \mathbb{Z}_p}\left(\sum_{i=1}^{n} m_i x_i = c \mod k\right) \leq \frac{b}{k} + \frac{n}{p}.$$

Proof. Proof proceeds by induction on n. Fix an arbitrary $c \in \mathbb{Z}_k$. WLOG, let $b = 1$, else we can divide all coefficients and c, and consider the probability $\mod k/b$. (If c is not divisible by b, the probability is 0, which is smaller than the required upper bound anyway.)
When $n = 1$, this is trivial, when $k \geq p$,

$$\Pr_{x_1 \in \mathbb{Z}_p}(x_1 = c \mod k) = \begin{cases} \frac{1}{p} & c < p \\ 0 & p \leq c < k \end{cases}$$

and when $k < p$,

$$\Pr_{x_1 \in \mathbb{Z}_p}(x_1 = c \mod k) = \begin{cases} \frac{1}{p} + \frac{1}{k} & c < p \mod k \\ \frac{1}{k} & p \mod k \leq c < p \end{cases}$$

Suppose the statement is true for $n - 1$. For n, we can write

$$\Pr_{x_i \in \mathbb{Z}_p}(\sum_{i=1}^{n} m_i x_i = c \mod k) = \frac{1}{p}\sum_{\theta \in \mathbb{Z}_p} \Pr_{x_i \in \mathbb{Z}_p}(\sum_{i=1}^{n-1} m_i x_i = c - m_n\theta \mod k)$$

Let b' be the GCD of all m_i with k except m_n. Since $b = 1$, b' must be coprime to m_n.
Now we claim that the maximum number of $\theta \in \mathbb{Z}_p$ for which $b' | (c - m_n\theta)$ (which is an arithmetic progression on integers) is $\lceil \frac{p}{b'} \rceil$. This is because any two θ_1, θ_2 are such that $b' | m_n(\theta_1 - \theta_2)$. Since m_n is coprime to b', θ_1 and θ_2 must differ by at least b', hence lower bounding the common difference of the arithmetic progression.

Then, using the inductive hypothesis and the fact that there are at most $\lceil \frac{p}{b'} \rceil$ $\theta \in \mathbb{Z}_p$ such that the probability on the RHS is non-zero,

$$\begin{aligned}
\frac{1}{p}\sum_{\theta \in \mathbb{Z}_p} \Pr_{x_i \in \mathbb{Z}_p}(\sum_{i=1}^{n-1} m_i x_i = c - m_n\theta \mod k) &\leq \frac{1}{p} \cdot \left\lceil \frac{p}{b'} \right\rceil \cdot \left(\frac{n-1}{p} + \frac{b'}{k}\right) \\
&< \frac{1}{p} \cdot \left(\frac{p}{b'} + 1\right) \cdot \left(\frac{n-1}{p} + \frac{b'}{k}\right) \\
&= \frac{n-1}{b'p} + \frac{1}{k} + \frac{n-1}{p^2} + \frac{b'}{kp} \\
&< \frac{n-1}{p} + \frac{1}{k} + \frac{1}{2p} + \frac{1}{2p} \\
&= \frac{n}{p} + \frac{1}{k}
\end{aligned}$$

□

B Constant-size PCS with l^ϵ-time verifier

In this section, we generalize the ideas from Section 4 to improve the verifier complexity from $O(\sqrt{l})$ to $O(l^\epsilon)$ for any constant $0 < \epsilon < 1$.

B.1 Construction of Polynomial Commitment Scheme sPC

We will define a family of protocols $\mathbf{TEST}_e$ parametrized by an integer $e > 1$ to replace the **TEST** protocol, with the only difference being the query vector $\mathbf{z} \in \mathbb{Z}_p^l$ that is being sent. The modified query vector $\mathbf{z}$ is generated as follows:

1. Sample random $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_e}$ from $\mathbb{Z}_p^{l^{1/e}}$. Coordinates of $\mathbf{x_j}$ are denoted by $x_{j,n}$ for $0 \le n < l^{1/e}$.
2. Any index k from 0 to $l-1$ can be written in base – $l^{1/e}$, consider the e-bit representation as $i_1 i_2 \ldots i_e$. Then,

$$z_k \equiv z_{i_1,i_2,\ldots,i_e} \coloneqq \prod_{j=1}^{e} x_{j,i_j}. \tag{29}$$

Specifically, we replace blue part of the **TEST** protocol in Figure 2 with the following:

$$\xleftarrow{\mathbf{z} \text{ defined by (29)}} \qquad \mathbf{x_1}, \ldots, \mathbf{x_e} \leftarrow\!\$\, \mathbb{Z}_p^{l^{1/e}}$$

We will also use the $\mathbf{IPP}_{sub}$ protocol as defined in Section 4.1 from a single random $x \leftarrow\!\$\, \mathbb{Z}_p$ and modifying the blue part of the **IPP** protocol (Figure 3) as explained in Section 4.1.

These changes are to ensure that σ in both $\mathbf{TEST}_e$ and $\mathbf{IPP}_{sub}$ are efficiently computable, details are given in Theorem B.5.

Polynomial Commitment Scheme sPC. The polynomial commitment sPC we use in this section is essentially the same as sqPC defined in Section 4.1 except that we replace $\mathbf{TEST}_2$ protocol in sqPC with $\mathbf{TEST}_e$ protocol to get sPC.

For this section, we assume $\alpha = p^L$ for an $L = O(el)$.

Before we present the proof of extractability, for notational simplicity we parametrise the definition of the set S to $S(\beta_1, \beta_2)$ and extend the domain of definition of χ_m, χ_n to $S(\beta_1, \beta_2)$ defined before Theorem 3.3.

$$S(\beta_1, \beta_2) \coloneqq \left\{ \frac{m\alpha - n}{k} : \; m, n, k \in \mathbb{Z},\, gcd(m,k) = 1,\, 0 < m \le k < p^{\beta_1}, -\beta_2 < n < k + \beta_2 \right\}$$

Theorem B.1. *The polynomial commitment scheme sPC (parametrised by e) satisfies Extractability (Def. 2.7) for $(\beta_1, \beta_2) = (e+1, 2^e+1)$ and $\alpha > 2^{e+1} p^{el+1+\log l}$, $\alpha = 0 \mod p$ in the Generic Group Model.*

Proof. This proof proceeds similar to the proof of extractability of the inner product scheme 3.1, we again have two theorems concerning $TEST_e$ and $\mathbf{IPP}_{sub}$.

Theorem B.2. *If the Verifier in $TEST_e$ instantiated with $\alpha > 2^{e+1} p^{el+1+\log l}$ outputs accept with non-negligible probability over the choice of the random $\mathbf{z} \in \mathbb{Z}_p^l$, there exists an efficient extractor in the GGM that outputs vectors $\mathbf{c}, \mathbf{d} \in \mathbb{Z}_p^l$ satisfying the following properties:*

1. $C = g^{\sum_{i=0}^{l-1} (c_i + \alpha d_i)\alpha^{2i}}$ *(proved in Lemma 3.10)*
2. $\mathbf{d} \in S(e, 2^e)^l$ *(proved in Lemma B.4)*
3. $\mathrm{DenLCM}(\chi_m(\mathbf{d})) < p^{el}$

Note that part 3 of the above Theorem is trivial from part 2 and the fact that the vector $\mathbf{d}$ is of length l. Since part 1 is also proved earlier, we only provide the proof of part 2 below, in Lemma B.4. The proof is a direct generalisation of the ideas in Lemma 4.7 using the higher dimensional box theorem.

Theorem B.3. *If the Verifier in* $\mathbf{IPP}_{sub}$ *instantiated with* α *such that* $\alpha > 2^{e+1}p^{el+1+\log l}, \alpha = 0 \mod p$ *outputs accept with non-negligible probability for some query vector* $\mathbf{q}$*, and given that the Verifier of* $TEST_e$ *also did so, there exists an efficient extractor in the GGM that outputs an opening* $\tilde{\mathbf{f}} \in \mathbb{Z}_p[x]$ *and an opening hint* $\tilde{\mathbf{c}}$ *in* $\mathbb{Q}(e+1, 2^e+1)$ *for* C *such that* $v = \langle \mathbf{f}, \mathbf{q} \rangle \mod p$.

We do not repeat the proof of Theorem B.3 as it is almost identical to that of Theorem 3.5. The only changes are in some of the parameters, specifically α being much larger and the query vector elements being as large as $p^{\log l}$. □

Lemma B.4. *If the success probability of the prover is at least* $C_e \cdot p^{-2^{-e+1}}$*, for some constant* $C_e > 1$*, then for all* $0 \leq i \leq l-1$*,* $d_i \in S(e, 2^e)$.

Proof. Suppose the prover succeeds with a non-negligible probability ($> C_e \cdot p^{-2^{-e+1}}$) over the random choice of $\mathbf{z}$ from elements of $\mathbb{Z}_p^{el^{1/e}}$.
Fix an arbitrary $0 \leq i \leq l-1$, equivalently the e-digit representation in base $l^{1/e}$: $(i_1, \ldots i_e)$. Consider the partition of the space $\mathbb{Z}_p^{el^{1/e}}$ by sets of the form

$$T_{\mathbf{q}} \coloneqq \{(x_{1,i_1}, \ldots, x_{e,i_e}, \mathbf{q}) : x_{j,i_j} \in \mathbb{Z}_p,\ 1 \leq j \leq e\} \text{ for } \mathbf{q} \in \mathbb{Z}_p^{el^{1/e}-e}.$$

Since the success probability of the prover is at least $\frac{C_e}{p^{2^{-e+1}}}$, at least one of these sets (which are e-dimensional spaces) must have more than $C_e \cdot p^{e-2^{-e+1}}$ points. Let such a set be denoted $T_{\mathbf{q}^*}$. Now, Lemma 4.6 gives an e-dimensional cube of accepting points in such a $T_{\mathbf{q}^*}$. Let that e-dimensional cube in $T_{\mathbf{q}^*}$ be given by

$$B := B(a_1, \ldots, a_e) := \{(x^*_{1,i_1} + a_1 b_1, \ldots, x^*_{e,i_e} + a_e b_e) \ :\ b_j \in \{0,1\}\},$$

for some fixed $0 < a_i < p$ and $x^*_{ji_j} \in \mathbb{Z}_p$.

Finally, let $\mathbf{z}_b \in \mathbf{Z}_p^l$ be the 2^e vectors constructed as in (29) from the vertices of the cube B above c^*.

Now, Lemma 3.6 and 3.7 give us equations corresponding to each accepting point, relating $\mathbf{c}, \mathbf{d}$ and $\mathbf{z}_b$.

Consider the complete binary tree of depth e, with the left child labeled 1 and the right child labeled 0 for each node. This gives a e-bit representation for all leaf nodes. Since we have 2^e equations corresponding to the points $(x^*_{1,i_1} + a_1 b_1, \ldots, x^*_{e,i_e} + a_e b_e)$ for some $0 < a_j < p$ and every $b_j \in \{0,1\}$, we have a natural assignment of points $(x^*_{1,i_1} + a_1 b_1, \ldots, x^*_{e,i_e} + a_e b_e)$ to leaves $b_1 b_2 \ldots b_e$. Define the equation at the leaf node by considering the difference of the left child equation and the right child equation. We can then combine equations at all non-leaf nodes in a higher dimensional analog of the process in the proof of Lemma 4.7.

Using such combinations as we move up the tree, the equation at the root node can be computed to be of the form

$$a_1 \ldots a_e \cdot d_{i_1,\ldots,i_e} = -(n) \mod \alpha$$

where n is the expression for the root term consisting of the floor parts and the u terms from the equations for each individual part.

This is the same process done in Lemma 4.7 generalised to higher dimensions. For example, when $e = 3$, n is of the form (and labeling the 2^3 points/leaves by z_i for $0 \leq i < 8$)

$$\left[\left(\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_1} \rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_2} \rangle}{\alpha} \right\rfloor\right) - \left(\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_3} \rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_4} \rangle}{\alpha} \right\rfloor\right)\right]$$
$$- \left[\left(\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_5} \rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_6} \rangle}{\alpha} \right\rfloor\right) - \left(\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_7} \rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_8} \rangle}{\alpha} \right\rfloor\right)\right]$$
$$+ [(u_1 - u_2) - (u_3 - u_4)] - [(u_5 - u_6) - (u_7 - u_8)]$$

Since for all x, $x - 1 \leq \lfloor x \rfloor < x$ and $u \in \{0, 1\}$, we can see that for the above expression,

$$\frac{c_i \cdot a_1 a_2 a_3}{\alpha} - 8 < n < \frac{c_i \cdot a_1 a_2 a_3}{\alpha} + 8$$
$$\implies -8 < n < a_1 a_2 a_3 + 8$$

In general, we get

$$-2^e < n < \prod_{i=1}^{e} a_i + 2^e$$

Since we already know that $d_{i_1,\ldots,i_e} < \alpha$, we can write the above as

$$d_{i_1,\ldots,i_e} = \frac{m\alpha - n}{a_1 \cdots a_e}$$

where $m \leq a_1 \cdots a_e$ and $-2^e < n < a_1 \cdots a_e + 2^e$. Hence, $d_{i_1,\ldots,i_e} \in S(e, 2^e)$. Since $(i_1, \ldots, i_e)$ was arbitrary, $\mathbf{d} \in S(e, 2^e)^l$.

□

Theorem B.5. *The polynomial commitment scheme* sPC *(parametrised by e) satisfies* proof succinctness *as defined in Definition 2.8 and the Verifier in* eval *performs $O(1)$ group operations and $O(l^{1/e})$ field operations (where $l - 1$ is the degree of the polynomial, and $e > 1$ is any constant).*

Proof. Proof succinctness follows from the same arguments as in Theorem 5.8. Similarly, we can see that the verifier still only does a constant number of group operations and we now have two bottlenecks w.r.t field operations; the computation of $\sigma \mod q$ for a random prime q and $\alpha \mod q$ for the same q.
Again, we need to analyse $TEST_e$ and $\mathbf{IPP}_{sub}$ separately. In $TEST_e$, the definition of the query vector implies that we can rewrite the computation of $\sigma \mod q$ in the following way:

$$\sigma \mod q = \sum_{k=0}^{l-1} \alpha^{2l-2-2j} z_j \mod q = \alpha^{2l-2} \prod_{i=1}^{e} \sum_{j=0}^{l^{1/e}-1} \frac{x_{i,j}}{\alpha^{2l^{1-i/e}j}} \mod q$$

This can be computed in $O(l^{1/e})$ time given $\alpha \mod q$. In $\mathbf{IPP}_{sub}$, the query vector is formatted in such a way that the computation can be done in $O(\log l)$ time (given $\alpha \mod q$), the analysis is the same as in Theorem 4.5,

$$\sigma \mod q = \sum_{k=0}^{l-1} \alpha^{2l-1-2j} q_k \mod q = \alpha^{2l-1} \prod_{i=0}^{\log l - 1} \left(1 + \frac{x^{2^i} \mod p}{\alpha^{2^{i+1}}}\right) \mod q$$

Now, for efficient computation, we need to compute $\alpha \mod q$. Since we pick $\alpha = p^{O(l)}$, this cannot be done in sublinear time for arbitrary α. Hence, we fix $\alpha = p^L$ for some $L > el + 1 + \log l = O(l)$. Then, computing $\alpha \mod q = p^L \mod q$, which can be efficiently done in $O(\log L) = O(\log l)$ using repeated squaring. Once this is found, $\alpha^{-1} \mod q$ can also be efficiently found using the Extended Euclidean algorithm. □

C Completeness of Dew-PC

Theorem C.1 (Completeness)**.** *The polynomial commitment scheme Dew-PC satisfies Completeness (Def 2.5).*

Proof. Note that by definition of CoeffSplit and completeness of PoKPE, all the PoKPE checks will accept.

To show that the last checks in **logTEST** and **logIPP** hold, it suffices to show that $v = 0$ in **TEST** and $v = f(x) \mod p$ in **IPP**. We will show this by expanding the computations done in CoeffSplit (defined

in Figure 1).
In **logTEST**, direct manipulation shows

$$\sum_{j=0}^{l-1} f_j \alpha^{2j} \times \sum_{j=0}^{l-1} \alpha^{2l-2-2j} z_j$$
$$= \alpha^{2l} \underbrace{\left(\sum_{j'>j} \alpha^{2(j'-j)-2} f_{j'} z_j \right)}_{\lambda} + \underbrace{\left(\sum_{j'<j} \alpha^{2l-2-2(j-j')} f_{j'} z_j + \sum_{j'=j} \alpha^{2l-2} f_j z_j \right)}_{\gamma}$$

and notice that since $\alpha > p^{2l \log l}$, these are indeed the γ, λ returned by CoeffSplit, and $v = 0$.

And, in **logIPP**, for the query vector $\mathbf{q}$ defined in (24)

$$\sum_{j=0}^{l-1} f_j \alpha^{2j} \times \sum_{j=0}^{l-1} \alpha^{2l-1-2j} q_j = \alpha^{2l-1} \left(\underbrace{\sum_{i=0}^{l-1} f_j q_j \mod p}_{v} + p \underbrace{\left\lfloor \frac{\sum_{i=0}^{l-1} f_j q_j}{p} \right\rfloor}_{n} \right)$$
$$+ \alpha^{2l-2}(0) + \alpha^{2l} \left(\underbrace{\sum_{j'>j} \alpha^{2(j'-j)-1} f_{j'} q_j}_{\lambda} \right) + \left(\underbrace{\sum_{j'<j} \alpha^{2l-1-2(j-j')} f_{j'} q_j}_{\gamma} \right)$$

Here, again since $\alpha > p^{2l \log l}$, $(v + np, \lambda, \gamma)$ above coincide with the output of CoeffSplit, and $v = \langle \mathbf{f}, \mathbf{q} \rangle \mod p$.
Note: The coefficients we ask for in CoeffSplit is $2l - 1$ for both **logTEST** and **logIPP** as the query vectors are appropriately shifted in the definition of σ – the honest answer we expect in **logTEST** is 0, while in **logIPP** it is the evaluation of the polynomial. □