Dew: A Transparent Constant-sized Polynomial Commitment Scheme

Arasu Arun[1]*, Chaya Ganesh[2], Satya Lokam[3], Tushar Mopuri[2]*, and Sriram Sridhar[4]*

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

Abstract. We construct a polynomial commitment scheme with constant (i.e., independent of the degree) sized evaluation proofs and logarithmic (in the degree) verification time in the transparent setting. To the best of our knowledge, this is the first result achieving this combination of properties.

We build our scheme from an inner product commitment scheme with constant-sized proofs but with linear verification time. To improve the verification time to logarithmic for polynomial commitments, we prove a new extremal combinatorial bound. Our constructions rely on groups of unknown order instantiated by class groups. We prove security of our constructions in the Generic Group Model.

Compiling known information-theoretic proof systems using our polynomial commitment scheme yields transparent and constant-sized zk-SNARKs (Zero-knowledge Succinct Non-interactive ARguments of Knowledge) with logarithmic verification.

1 Introduction

A Polynomial Commitment Scheme (PCS) [18] allows a prover to commit to a polynomial P of degree d so that, later. a verifier can query for $P(x)$ at an argument x of its choice and the prover can, together with its response, furnish an evaluation proof that its response is indeed consistent with its commitment. The commitment and the evaluation proof are required to be *succinct*, that is, of size independent of, or logarithmic, in d.

Polynomial commitments have applications in verifiable secret sharing [15], anonymous credentials [10], and zero-knowledge sets [22], among others. But by

* Work partially done while at Microsoft Research India.

far, their most dominant application is to constructions of zkSNARKs (zero-knowledge Succinct Non-interactive ARguments of Knowledge) for all of NP. Indeed, improvements to PCS imply improvements to SNARKs when combined with established modular approaches to SNARKs. On the other hand, a SNARK for all of NP in particular implies a succinct PCS by instantiating the SNARK for the NP-relation "$y = P(x)$ and the commitment C opens to P", where C is a succinct commitment to P. While this incidental corollary of a SNARK implies succinct PCS, we desire a "direct" construction of PCS without writing polynomial evaluation as a generic NP relation, since most SNARK constructions themselves use PCS as a crucial ingredient. PCS is therefore a core cryptographic construct and there is a strong motivation to construct one with the best possible parameters.

Transparent setup. Non-interactive proof systems are typically in the Common Reference String (CRS) model where a CRS is generated during a setup phase which needs to be *trusted* if the CRS uses secret randomness. Constructions that do not involve a trusted setup phase and the verifier randomness consists of only *public coins* are called *transparent*. A recent line of work [12, 23, 11, 17] to construct SNARKs follows a modular approach: first, an information-theoretic component is constructed; then this is compiled into an argument using cryptographic tools, typically a PCS. 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. 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 trusted-setup assumptions [18], or are when they are transparent, only achieve logarithmic proof size [9, 3, 20][5]. We address this challenge in this work.

1.1 Our Contributions

We present the first PCS with constant size commitment, constant size evaluation proof [6] and logarithmic verification 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 a verifier's query vectors. Our IPC is succinct – that is, the size of the commitment and the proof of a correct opening are independent of the length of the vector and linear (in the length of the vector) time verification. Building on this IPC, we construct a succinct PCS,

[5] A flaw in the proof of security of the DARK scheme [9] was discovered by Block et al [3], who propose a different PCS with logarithmic proof size. We also note that a revised version of DARK [8] also proposes a fix by showing that the DARK PCS satisfies a property called almost-special-soundness which suffices for extraction.

[6] Constant is $O_\kappa(1)$. That is, independent of the size of the input, and polynomial only in the security parameter.

resulting in a *transparent constant-sized PCS* but, importantly, with *logarithmic time verification*. Our PCS is the *first* construction to achieve the above combination of properties 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 §4, 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 [9, 12], we obtain a *transparent constant-sized zkSNARK with constant-sized public parameters* (§5). The resulting zkSNARKs achieve $O_\kappa(1)$ communication and $O(\log n)$ verification[7], where n is the complexity of the NP relation (e.g., 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. The VCs of [5, 19] are such candidates.

A New Combinatorial Lemma. As noted above, we improve the verification time from linear (in length of vector) in our IPC to logarithmic (in degree of polynomial) in our PCS. We achieve this efficiency improvement using Kronecker products (details in § 1.3.2 and §4.1); but their naive application breaks soundness. We recover soundness by solving a problem in extremal combinatorics. A special case of our problem asks: how many points can we choose in the discrete cube $[n]^d$ such that that set of points does not contain the corners of a d-dimensional hyper-rectangle (box)? When $d = 2$, this is the Zarankiewicz problem [4] in extremal graph theory for which an asymptotically tight bound of $\sim n^{3/2}$ is known. For higher d, the "Box Theorem" due to Rosenfeld [24] proves the bound $\sim n^{d-2^{-d+1}}$. We can use these bounds in the soundness proof of our PCS to obtain n^ϵ verification time for any constant $0 < \epsilon < 1$. While this improves on linear, our goal is to obtain *logarithmic* verification.

We achieve logarithmic verification by generalizing boxes in the extremal problem to "d-cancellation structures." With these d-cancellation structures, we can continue to exploit certain cancellation properties required for soundness (details in §4.2) similar to those for boxes but also, more importantly, succeed in proving much better bounds on the number of points in $[n]^d$ that do not contain a d-cancellation structure. Our bounds are $\sim dn^{d-1}$ and it is crucial for our soundness that this is a negligible fraction of the whole space (of size n^d); in contrast, as $d \to \infty$, the box theorem above gives a bound that approaches n^d – essentially filling the whole space.

[7] In the preprocessing setting.

To the best of our knowledge, our result is the first application of an extremal combinatorics theorem in the construction of PCS and SNARKs and we believe this to be of independent interest. We note that extremal combinatorics results like this have found applications in complexity theory and theoretical computer science in general.

Recovering the DARK [9] 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. The DARK recovery does not require GGM; we achieve this result under the same assumptions made in DARK, i.e., the Adaptive Root and Strong RSA Assumptions. We note that [3] gives a construction that achieves similar results as DARK by modifying DARK's evaluation protocol, and a subsequent revision of DARK [8] shows that the DARK PCS satisfies a property called almost-special-soundness. In contrast, our construction is a commitment scheme that is syntactically close to DARK, has a similar evaluation protocol and recovers the flawed lemmas. We present this in the full version [1].

1.2 Related Work

Functional commitments were introduced by [21] as a generalization of vector commitments, where a prover can commit to a vector, and later open the commitment at functions of the committed vector with a succinct proof that the answer is consistent with the committed vector. The work of [21] also showed a construction for functional commitments for linear functions. Lai and Malavolta [19] put forth the notion of Linear Map Commitments (LMC) that allow a prover to open a commitment to the output of a linear map. The constructions from [21, 19] achieve succinctness — constant commitment and proof size, but require trusted setup.

In a recent concurrent work, [13] 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. Though a PCS was not their goal, a PCS resulting from the inner product commitment scheme of [13] in the natural way results in a linear time verifier. In contrast, we achieve logarithmic verification time for our PCS.

Polynomial commitment schemes were introduced in [18], and have since led to several variants being used in recent SNARKs. The KZG scheme [18] gives constant-sized commitments and proofs, but require a trusted setup. In the transparent setting, Wahby et al. [25] 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. [27] construct a polynomial commitment from FRI (Fast Reed Solomon IOPP) that is transparent, has constant size commitments, but evaluation proofs have size $O(\log^2 d)$.

As mentioned earlier, Bünz et al [9] used a Diophantine Argument of Knowledge (DARK), and constructed a polynomial commitment scheme with proof size $O(\log d)$ and $O(\log d)$ verification time for polynomials of degree d. Block et al [3] identified a gap in the proof of security of the DARK scheme and propose a modification that sidesteps the gap in extraction, resulting in a PCS of polylogarithmic proof size and verification time.

1.3 Technical Overview

The intuitive starting point of our commitment schemes is a natural mapping from vectors to group elements *via* integers. Specifically, for a vector[8] $\mathbf{c}$, define $\mathsf{int}_\alpha(\mathbf{c}) := \langle \mathbf{c}, \boldsymbol{\alpha} \rangle := \sum_0^{l-1} c_i \alpha^i$, where $\boldsymbol{\alpha} := (1, \alpha, \alpha^2, \ldots, \alpha^{l-1})$ and α is sufficiently large. Let us also define $C := g^{\mathsf{int}_\alpha(\mathbf{c})}$ for a given group element $g \in \mathbb{G}$. When the group $\mathbb{G}$ is a *group of unknown order* and $g \in \mathbb{G}$ is random (but chosen during set up), we can show that a prover can prove knowledge of a unique positive exponent of C with base g and that the α-base representation $\mathbf{c}$ of that exponent must be a valid opening of C. This follows from a Proof of Knowledge of a *Positive* Exponent (PoKPE) protocol that builds on Wesolowski's Proof of Exponent (PoE) protocol [26] (details in Section 2.4).

We now wish to use the commitment C made by a Prover to vector c in a protocol for inner product $\langle \mathbf{c}, \mathbf{q} \rangle$, where $\mathbf{q}$ is the *query vector* from the Verifier. To that end, let us consider the integer product

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

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

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

Note that the verifier can compute the l.h.s. of (2). Hence if the prover claims that the inner product $\langle \mathbf{c}, \mathbf{q} \rangle$ evaluates to v and *also* sends g^L and g^H to the verifier (and convinces the verifier of values L and H using a PoKPE protocol), 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 contribution, outlined below, 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. This intuition is essentially the basis for our inner product evaluation protocol **IPP** in Fig 3 and the "additional machinery" appears as the **TEST** protocol in Fig 2.

[8] Our vectors are over $\mathbb{Z}_p$ and we map elements of $\mathbb{Z}_p$ to integers $\{0, \ldots, p-1\}$.

We remark that the above intuition is reminiscent of approaches in [19, 9]. However, our approach below differs significantly from theirs to achieve *constant sized* proofs (unlike in [9] that uses recursion to obtain logarithmic size) and *transparent* setting where α is not secret (unlike the trusted setup in [19]).

1.3.1 A TEST Protocol and Extracting Structure from Overflows

A cheating prover could use a committed vector (derived by computing the α-base representation of exponent of g in C using the PoKPE extractor) with coordinate values that could cause "overflow" in the coefficients on the r.h.s. of (1). In that case, we can no longer guarantee the correctness of the inner product as the middle coefficient. We now describe ideas that help us overcome this challenge.

To control the issues caused by overflow, we intersperse 0's between coordinates of $\mathbf{c}$: we 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. More generally, let $\mathbf{d}$ denote the subvector in the odd positions and let $\mathbf{c}\|\mathbf{d}$ denote the combined vector of length $2l$. Note first that completeness continues to hold with this change since an honest prover commits to $\mathbf{c}\|\mathbf{0}$, with $\mathbf{c} \in \mathbb{Z}_p^l$ and satisfy analogs of (1) and (2) for length-$2l$ vectors with 0's in odd positions, or equivalently, with α^2 replacing α. Second, and this is our next crucial step, note that the verifier can run a **TEST** protocol (cf. Fig 2) that queries for the inner product $\langle(\mathbf{c}\|\mathbf{d})\ ,\ (\mathbf{0}\|\mathbf{z})\rangle$, where $\mathbf{z} \in \mathbb{Z}_p^l$ is a uniformly chosen *random* vector and the verifier can check if the middle coefficient of (generalization to $2l$-length vectors of) (2) is zero. *Assuming no overflows*, the middle coefficient would be $\langle \mathbf{d}, \mathbf{z}\rangle$. In this case, by the Schwartz-Zippel lemma, a cheating prover choosing nonzero $\mathbf{d}$ would be caught with high probability. However, a cheating prover could choose nonzero $\mathbf{d}$ and still pass the test for $\mathbf{z}$ by causing overflows from L in (1) (due to large products between $\mathbf{c}$ and $\mathbf{z}$ coordinates) to "cancel out" the non-zero value of $\langle \mathbf{d}, \mathbf{z}\rangle$. More precisely, it can be shown using (a generalization of) (1) and the PoKPE protocol relating (2) to (1) that if a prover can succeed in the **TEST** protocol with non-negligible probability, then

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

for some $u \in \{0, 1\}$ must be satisfied (3) with non-negligible probability over a uniformly random $\mathbf{z} \in \mathbb{Z}_p^l$.

Call a test vector $\mathbf{z} \in \mathbb{Z}_p^l$ a *success point* for the prover if (3) is satisfied when verifier chooses $\mathbf{z}$ and prover's commitment C extracts – via a PoKPE protocol – to $\mathbf{c}\|\mathbf{d}$. Now, if a prover has two success points $\mathbf{z}$ and $\mathbf{z}'$ on a "line", i.e., $\mathbf{z}$ and $\mathbf{z}'$ agree on all coordinates except j-th, then $\langle \mathbf{d}, \mathbf{z}\rangle - \langle \mathbf{d}, \mathbf{z}'\rangle = d_j(z_j - z'_j)$ because of cancellations in coordinates $\neq j$. Thus subtracting (3) for $\mathbf{z}'$ from that for $\mathbf{z}$, we obtain

$$d_j \cdot (z_j - z'_j) = \left(\left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}'\rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}\rangle}{\alpha} \right\rfloor + u' - u \right) \mod \alpha. \tag{4}$$

Using bounds on coordinates of $\mathbf{c}$ and $\mathbf{z}$, we conclude that d_j is $\theta_{1j}\alpha + \theta_{2j}$, where θ_1 and θ_2 are rationals with small denominators (a more detailed statement of this structure appears in Theorem 34). An easy combinatorial argument shows that if the prover succeeds with non-negligible probability, e.g., at least $1/p$ (p is $\exp(\kappa)$), then in *every* direction j, there must be a line in the j-th dimension with two success points $\mathbf{z}$ and $\mathbf{z}'$ on it. Hence, if the prover is accepted in the **TEST** protocol (Fig 2) with non-negligible probability, every coordinate of $\mathbf{d}$ can be expressed as $d_j = \theta_{1j}\alpha + \theta_{2j}$ with θ's as above – *this is the structure we extract on* $\mathbf{d}$ *that we use to prove soundness*. This Structure Theorem 34 is a crucial technical ingredient of our results.

Armed with the structure theorem, we prove (Theorem 35) that if the inner product evaluation protocol **IPP** (Fig 3) for $\langle(\mathbf{c}\|\mathbf{d}), (\mathbf{q}\|\mathbf{0})\rangle$ succeeds in satisfying (generalizations) of (1) and (2), with query vector $\mathbf{q}\|\mathbf{0}$, 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 $\mathbf{c}$ that is consistent with the claimed inner product.

1.3.2 Logarithmic Verification for Polynomial Commitments

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. Linear verification seems inherent for inner products (since the query vector $\mathbf{q}$ can be arbitrary and the verifier needs to at least read the statement, verifier's computation of $\mathsf{int}_\alpha(\mathsf{reverse}(\mathbf{q}))$ itself will take linear time). But, in a PCS, we can hope to achieve logarithmic verification time since the query vector $\mathbf{x}$ is parameterized by single variable x. In particular, we can compute $\mathsf{int}_\alpha(\mathsf{reverse}(\mathbf{x}))$ in only logarithmic time (cf. (5) below and (11)). This makes the verifier in **IPP** protocol (specialized to a PCS) logarithmic. However, we still have the bottleneck for verifier computation in the **TEST** protocol. Note that while the query vector $\mathbf{x}$ is parameterized by a single variable x, the *test vector* $\mathbf{z}$ is not and hence computing $\mathsf{int}_\alpha(\mathsf{reverse}(\mathbf{z}))$ in checking (2) in **TEST** protocol still seems to require linear verifier time.

To reduce verifier's computation in **TEST** protocol, we use the idea of **Kronecker products**[9]: instead of choosing $\mathbf{z}$ uniformly at random in $\mathbb{Z}_p^l$, we choose $\log l$ vectors $\mathbf{z_0}, \ldots, \mathbf{z_{\log l-1}}$ uniformly at random from $\mathbb{Z}_p^2$ and define $\mathbf{z} = \mathbf{z_0} \otimes \cdots \otimes \mathbf{z_{\log l-1}}$ 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 $i \in [l]$.

$$\mathsf{int}_\alpha(\mathsf{reverse}(\mathbf{z})) = \sum_{i=0}^{l-1} \alpha^{l-i} \prod_{j=0}^{\log l-1} z_{j,i_j} = \alpha^l \cdot \prod_{j=0}^{\log l-1} (z_{j,0} + z_{j,1}\alpha^{-2^j}), \qquad (5)$$

[9] We note that [2] and [20] also use Kronecker products in proof systems albeit with different motivations.

and note that the last product can be computed in logarithmic time. The new test protocol with Kronecker product test vectors is called **logTEST** (identical to **TEST** except with the query vector replaced as above).

While this helps improve verifier efficiency of **TEST**, it *breaks soundness*! The extractability proof of **TEST** in IPC relies on uniform randomness of the test vector $\mathbf{z} \in \mathbb{Z}_p^l$. So, we must now *improve the extractability proof to work with exponentially smaller randomness* in the $\log l$ vectors $\mathbf{z}_j$ of length 2. Specifically, it is crucial to recover an analog of the structure for the $\mathbf{d}$ vector as outlined in Section 1.3.1 but now from this vastly reduced space of verifier's randomness in **logTEST**. We outline how to do this next.

1.3.3 An Extremal Combinatorial Bound

We recover soundness with Kronecker product **TEST** vectors by proving a *new result in extremal combinatorics*. Informally, this theorem (Theorem 45) 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 that we call a d-cancellation structure. A d-cancellation structure generalizes the set of corners of a d-dimensional *box* or a hyper-rectangle. For instance, a 2-cancellation structure is a parallelogram generalizing a rectangle. In the case of a rectangle, this is the well-known Zarankiewicz problem [4] from extremal graph theory and has an asymptotically tight bound of $n^{3/2}$ points (out of n^2) that contain no four points as corners of a rectangle in $[n]^2$. Thus, our problem generalizes this in two ways: first, we consider high dimensions with growing d (but no more than $\log n$) and second, we generalize a rectangle/box to d-cancellation structure. A recursive definition is given in Def. 44. For $d > 2$ and in case of boxes, Rosenfeld [24] proved an upper bound of $\sim n^{d-2^{-d+1}}$ on the maximum 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. Our main contribution is to obtain a significantly smaller upper bound by generalizing boxes to d-cancellation structure: a tight upper bound of $(n^d - (n-1)^d) \leq dn^{d-1}$, which is a vanishingly small fraction of n^d. For example, when the forbidden configurations are generalized from rectangles to parallelograms for $d = 2$, the upper bound improves to $\sim n$ from the $\sim n^{3/2}$ stated above for the Zarankiewicz problem.

We now tie back this combinatorial argument to the goal of extractability. As the name implies, a d-cancellation structure induces cancellations. Recall from Section 1.3.1 that cancellations between two success points (test vectors $\mathbf{z}$ and $\mathbf{z}'$ where the prover succeeds by satisfying (3)) allow us to deduce structural conditions on coordinates of $\mathbf{d}$ using (4). We now generalize that argument to Kronecker products as test vectors where a d-cancellation structure generalizes the role of a line, and cancellations between two points on a line generalize to recursive cancellations among 2^d points in a d-cancellation structure. Finally, the simple combinatorial argument outlined in Section 1.3.1 for **TEST** on the existence of at least one line in each dimension with at least two success points is replaced by the existence of a d-$\mathsf{cancellation\ structure}$ for every index $i = (i_0, \ldots, i_{\log l - 1})$ (corresponding to a $\mathbf{d}$-coordinate d_i, cf. (5)) in the Kronecker product space for **logTEST**.

Specifically, each accepting run of **logTEST** corresponds to a chosen/success point in $[n]^d$ (this is our space of randomness, with $n = p$ and $d = \log l$). By suitable calibration of parameters, we can show that a prover that succeeds with a non-negligible probability gives rise to more than $n^d - (n-1)^d$ chosen points and then our combinatorial bound above implies the existence of a d-$\mathsf{cancellation\ structure}$ B, each of whose "corners" (for simplicity, think of a d-$\mathsf{cancellation\ structure}$ as a box) is a success point. Thus, we obtain 2^d equations like (3) at the corners of B; $\langle \mathbf{d}, \mathbf{z} \rangle$ is a multilinear polynomial with coefficients d_i (i-th coordinate of $\mathbf{d}$, with bit representation of $i = (i_0, \ldots, i_{\log l - 1})$) and variables z_{j,i_j} (cf. (5)) from the Kronecker product **TEST** vector $\mathbf{z} = \otimes_{j=0}^{\log l - 1} \mathbf{z_j}$. The recursive structure of B allows recursively combining these equations by folding, i.e., subtracting equations like (3) along "edges" of B in the same direction. Each successive folding reduces the number of equations by half and eliminates one of the free variables z_{j,i_j} to obtain a multilinear version of (4). After $\log l$ such folding steps, we obtain an equation generalizing (4) with one d_i on l.h.s, that yields the structure on coordinates of $\mathbf{d}$ that we seek. This helps us recover an analog of the structure theorem (Theorem 34) for **logTEST** (Theorem 42).

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 [16] 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 [21]. 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:

- **Setup**$(1^\kappa, D) \to \mathsf{pp}$. On input security parameter κ, and an upper bound D on accepted vector lengths, **Setup** generates public parameters pp.
- **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 $\tilde{\mathbf{c}} \equiv (f_0 \cdot \ldots, f_{l-1})$.
- **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.
- **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 21 (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 22 (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 23 (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 24 (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 [18] 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 25 (Completeness) *For all polynomials $f(X) \in \mathbb{F}[X]$ of degree $d \leq D$, for all $x \in \mathbb{F}$,*

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

Definition 26 (Binding) *A polynomial commitment scheme PC is binding if for all PPT $\mathcal{A}$, the following probability is negligible in κ:*

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

Definition 27 (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 28 (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 [6, 7] 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 [14], and used in [5]. 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.

We informally describe the rest of the assumptions – note that these problems are indeed intractable in the GGM. The formal definitions are deferred to the full version.

Adaptive root assumption. Computing random roots of arbitrary group elements g is hard for any PPT adversary.

Low order assumption. Computing the order of any non-trivial element in a group $\mathbb{G} \leftarrow \mathsf{GGen}$ is hard for any PPT adversary.

2.4 Proofs about Exponents

PoE *(Proof of Exponentiation):* We use Wesolowski's proof of exponentiation (PoE) protocol [26] in it's slightly more generalized form as presented in [5] for the relation $\mathcal{R}_{\mathsf{PoE}} = \{(u, w \in \mathbb{G}, x \in \mathbb{Z}; \perp) : w = u^x \in \mathbb{G}\}$.

PoKE *(Proof of Knowledge of Exponent):* We also use the PoKE protocol from [5] in our protocols. This protocol is an argument of knowledge in the GGM for the relation $\mathcal{R}_{\mathsf{PoKE}} = \{(u, w \in \mathbb{G}; x \in \mathbb{Z}) : w = u^x \in \mathbb{G}\}$.

PoKPE *(Proof of Knowledge of* Positive *exponent):* Define the relation $\mathcal{R}_{\mathsf{PoKPE}} = \{(w \in \mathbb{G}; x \in \mathbb{Z}) : (w = g^x) \wedge (x > 0)\}$. We construct an argument of knowledge for this relation called PoKPE using PoKE and Lagrange's four-square theorem. We also use the notation $\mathsf{PoKPE}\{A, B, \dots\}$ to denote the combined protocol for the set, where the verifier outputs 1 iff PoKPE checks pass for all elements. More details about these protocols appear in the full version.

3 Inner Product Commitment Scheme with Constant-sized Proof

In this section, we construct an inner product commitment (IPC) scheme that achieves constant-sized proof and linear time verification.

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 $g \leftarrow\$ \mathbb{G}$. Define $\alpha = p^{2D}$ (p is a large prime such that $\mathsf{len}(p) = \mathsf{poly}(\kappa)$). Return $\mathsf{pp} = (\kappa, \mathbb{G}, g, p)$ (α does not have to be explicitly returned; it is defined completely by p, D).

– $\mathbf{Com}(\mathsf{pp}, D, f_0, \dots, f_{l-1}, l)$: Define the commitment to $\mathbf{f} = (f_0, \dots, f_{l-1}) \in \mathbb{Z}_p^l$ as $C := g^{\sum_{i=0}^{l-1} f_i \alpha^{2i}}$, considering $f_i \in \mathbb{Z}_p$ as integers in $[0, p-1]$ and the sum in $\mathbb{Z}$. If $l \leq D$, return $(C, \mathbf{f})$, else return error.

– $\mathbf{Open}(\mathsf{pp}, \mathbf{f}, l, C, \tilde{\mathbf{c}})$: ($\mathbf{f}$ is the claimed opening and $\tilde{\mathbf{c}}$ is an opening hint) Return 1 if all the below conditions hold, else return 0.

  - $l \leq D, \tilde{\mathbf{c}} = \mathbf{f} \mod p$ [10]

[10] Treating any coordinate $\frac{a}{b}$ of $\tilde{\mathbf{c}} \mod p$ as $a' \cdot b'^{-1}$, where $a' = (a \mod p) \in \mathbb{Z}_p$ and $b' = (b \mod p) \in \mathbb{Z}_p$.

- $C = g^{\sum_{i=0}^{l-1} \tilde{c}_i \alpha^{2i}}$, exponent $\sum_{i=0}^{d} \tilde{c}_i \alpha^{2i} \in \mathbb{Z}$ and $\tilde{\mathbf{c}} \in \mathbb{Q}(2,3)^l$, where

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

where $|a|$ denote s 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* $\beta_1 \le \beta_1', \beta_2 \le \beta_2'$)

– **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.

- $b_1 \leftarrow$ **TEST**$(C, l; \mathbf{f})$, $b_2 \leftarrow$ **IPP**$(C, l, \mathbf{q}, v; \mathbf{f})$. Return $b = (l \le D) \wedge b_1 \wedge b_2$

3.2 Proofs of Security

We prove that our construction IPC satisfies the requirements of an inner product scheme as defined in §2.1.

Theorem 31 (Completeness) *The inner product commitment scheme* IPC *satisfies Completeness (Definition 21).*

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

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

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

And, in **IPP**,

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

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

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).

Fig. 1: CoeffSplit

TEST

Prover		**Verifier**
$C := \mathsf{commit}(\mathsf{pp}, \mathbf{f} \in \mathbb{Z}_p^l)$	$\xrightarrow{C}$	
	$\xleftarrow{\mathbf{z}}$	$\mathbf{z} \leftarrow\!\!\$\ \mathbb{Z}_p^l$ uniformly at random
Computations in **TEST**	$\xrightarrow{(\Lambda, \Gamma)}$	Checks

Computations in **TEST**

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

Define $\Lambda := g^\lambda$, $\Gamma := g^\gamma$

Checks

$$\sigma := \sum_{j=0}^{l-1} \alpha^{2l-2-2j} z_j$$

$$E := \frac{g^{\alpha^{2l-1}}}{C}, \Delta := \frac{g^{\alpha^{2l-1}}}{\Gamma}$$

1 : $\mathsf{PoKPE}\{C, E, \Lambda, \Gamma, \Delta\}$ accepts

2 : $C^\sigma \overset{?}{=} \Lambda^{\alpha^{2l}} \Gamma$

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

Fig. 2: The **TEST** Protocol

IPP

Prover		**Verifier**
	$\xleftarrow{\mathbf{q}}$	$\mathbf{q} \in \mathbb{Z}_p^l$ is the inner product vector
Computations in **IPP**	$\xrightarrow{(v, N, \Lambda, \Gamma)}$	Checks

Computations in **IPP**

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

Define $\Lambda := g^\lambda$, $\Gamma := g^\gamma$

$N := g^{\lfloor \frac{v}{p} \rfloor}$, $v := v' \mod p$

Checks

$$\sigma := \sum_{j=0}^{l-1} \alpha^{2l-1-2j} q_j$$

$$\Delta := \frac{g^{\alpha^{2l-2}}}{\Gamma}$$

1 : $v \in \mathbb{Z}_p$

2 : $\mathsf{PoKPE}\{\Lambda, \Gamma, \Delta, N\}$ accepts

3 : $C^\sigma \overset{?}{=} (g^v N^p)^{\alpha^{2l-1}} \Lambda^{\alpha^{2l}} \Gamma$

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

Fig. 3: The **IPP** Protocol

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 32 (Binding) *The inner product commitment scheme* IPC *Construction in §3.1 is binding (Definition 22) for opening hint vectors in* $\mathbb{Q}(\beta_1, \beta_2)$ *as long as* $\alpha > 4\beta_2 p^{2\beta_1}$ *and if the Order assumption holds for* GGen.

Proof. Suppose there exists an adversary $\mathcal{A}$ which breaks binding as defined in Definition 22, 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 Low order assumption. 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 gives a contradiction, since

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

Before proving extractability, we need a few definitions. Define

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

as a subset of $\mathbb{Z}$ 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.

Theorem 33 (Extractability) *The inner product commitment scheme* IPC *satisfies Extractability for* $(\beta_1, \beta_2) = (2, 3)$ *(Def. 24) 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 and the extractor of the first theorem and finishes the proof.
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 27) with overwhelming probability.

Theorem 34 (TEST Extractor) *If the Verifier in* **TEST** *outputs accept with non-negligible probability, there exists an efficient extractor* Ext_T *in the GGM that outputs* $\mathbf{c}, \mathbf{d} \in [\alpha]^l$ *such that* $C = g^{\sum_{i=0}^{l-1}(c_i+\alpha d_i)\alpha^{2i}}$ *and* $\mathbf{d} \in S^l$*.*

Theorem 35 (IPP Extractor) *If the Verifier in* **IPP** *outputs accept with non-negligible probability (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 opening hint* $\tilde{\mathbf{c}} \in \mathbb{Q}(2,3)$ *for* C *such that* $v = \langle \tilde{\mathbf{f}}, \mathbf{q} \rangle \mod p$ *and satisfies Extractability (Def. 24).*

3.3 Auxiliary lemmas

Lemma 1. *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* $U_i < \alpha$*,* $u_i \in \{0,1\}$*, and*

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

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

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

3.4 Proof of Theorem 34 - TEST extraction

First, we prove a lemma giving a partial extractor for the **TEST** protocol.

Lemma 3. *If the Verifier in* **TEST** *outputs accept with non-negligible probability, there exists an efficient extractor* Ext_T *in the GGM that outputs with high probability* $\mathbf{c}, \mathbf{d} \in \mathbb{Z}_p^l$ *(that only depends on the commitment* C*) such that* $C = g^{\sum_{i=0}^{l-1}(c_i+\alpha d_i)\alpha^{2i}}$*.*
Moreover, if the random vector used in **TEST** *was* $\mathbf{z} \in \mathbb{Z}_p^l$*, then we also have the relation (for some* $u \in \{0,1\}$*):*

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

Proof. We define an extractor Ext_T that 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 $\mathbf{d}$. By construction, $C = g^{\sum_{i=0}^{l-1}(c_i+\alpha d_i)\alpha^{2i}}$.

Notice that each PoKPE is essentially a range check as well as a proof of knowledge of the exponent. With overwhelming probability, we can assume that each of these statements are true (due to knowledge soundness of PoKPE). Hence, we get that the prover "knows" (formally, an extractor outputs) integers c, γ, λ such that $C = g^c$, $0 < c < \alpha^{2l-1}$, $\Gamma = g^\gamma$, $0 < \gamma < \alpha^{2l-1}$, and $\Lambda = g^\lambda$, $0 < \lambda$.

Now, notice that for any equality of group elements in a group of unknown order, we can (with overwhelming probability) equate their exponents when written with base g *over integers*. This follows from the Low order assumption as long as the prover knows all the exponents w.r.t. some fixed base g (else the prover could compute a multiple of the order of the group).

Hence, given an equation of the form $C^\sigma = \Gamma g^{v\alpha^{2l-1}} \Lambda^{\alpha^{2l}}$ (this is essentially Check 2 for the **TEST** verifier with $y = 0$), we can write

$$\begin{aligned} g^{c\sigma} &= g^{\gamma + v\alpha^{2l-1} + \lambda\alpha^{2l}} \\ \implies c\sigma &= \gamma + v\alpha^{2l-1} + \lambda\alpha^{2l} \end{aligned}$$

Writing this in base α and using Lemma 1, we can compare the coefficients of α^{2l-1} on both sides:

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

In **TEST**, we have $v = 0$, hence we prove the lemma.

Now, using the above lemma, we show that if the prover succeeds with non-negligible probability, we must have $d_i \in S$ for all i.
If the prover succeeds with non-negligible probability, it must hold that it also succeeds for a non-negligible probability over the choice of the random query $\mathbf{z} \in \mathbb{Z}_p^l$. Fix some index $0 \le i \le l-1$.
Partition the randomness space $\mathbb{Z}_p^l$ into 1-dimensional "lines" of length p along the i^{th} dimension:

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

If the prover was only able to succeed for at most one value of $\mathbf{z}$ in all $T_{\mathbf{q}}$, the overall success probability of the prover is bounded by $\frac{1}{p}$, which is negligible and hence a contradiction. Hence, there must exist some $\mathbf{q}$ such that there exists two points $\mathbf{z}_1, \mathbf{z}_2 \in T_{\mathbf{q}}$ such that the prover succeeds in convincing the verifier (for simplicity, we will use z_1, z_2 to denote the i^{th} coordinate that differs in these two vectors. WLOG let $z_2 > z_1$).

Now, using Lemma 3, we get two equations:

$$\langle \mathbf{d}, \mathbf{z_1} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_1} \rangle}{\alpha} \right\rfloor + u_1 \mod \alpha = 0 \mod \alpha \tag{6}$$

$$\langle \mathbf{d}, \mathbf{z_2} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z_2} \rangle}{\alpha} \right\rfloor + u_2 \mod \alpha = 0 \mod \alpha \tag{7}$$

where $u_1, u_2 \in \{0, 1\}$. Our aim is to isolate and prove conditions on a single coordinate d_i, and notice that the inner products $\langle \mathbf{d}, \mathbf{z_1} \rangle$ and $\langle \mathbf{d}, \mathbf{z_2} \rangle$ differ only in the i^{th} term. Hence, subtracting the two equations, we get:

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

Call the term in the brackets on the RHS 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 $-2 < n < (z_2 - z_1) + 2$. We also know that $0 < z_2 - z_1 < p$. Letting $k := z_2 - z_1$,

$$kd_i = -n \mod \alpha \implies d_i = \frac{m\alpha - n}{k}$$

where $-2 < n < k + 2, 0 < k < p$ and $0 < m \leq k$ since $d_i < \alpha$.

Hence $d_i \in S$. Since i was an arbitrary index, $\mathbf{d} \in S^l$.

3.5 Proof of Theorem 35 – IPP extraction

We define the final extractor Ext for eval using the extractor Ext_T from **TEST** that outputs $\mathbf{c}, \mathbf{d} \in \mathbb{Z}_p^l$. Specifically, Ext invokes Ext_T and performs the following additional 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 $m_{-1} := 0$, $k_{-1} := 1$ and define vectors $\mathbf{c}', \mathbf{d}' \in \mathbb{Z}_p^l$ as $c'_i := c_i + \frac{m_{i-1}}{k_{i-1}}$ and $d'_i := -\frac{n_i}{k_i}$.
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 as Ext_T is efficient and the only non-trivial computations done are in Step 1 above, which can be done efficiently (details in full version).

By construction, this is a valid opening hint, as $\mathbf{d} \in S^l \implies \mathbf{c}' + \alpha\mathbf{d}' \in \mathbb{Q}(2, 3)$. This is just a rearrangement of the coordinates of $\mathbf{c}$ and $\mathbf{d}$ and keeps 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}). Hence, $C = g^{\sum_{i=0}^{l-1}(c'_i + \alpha d'_i)\alpha^{2i}}$ and the exponent $\in \mathbb{Z}$.

Now, since the verifier accepts in **IPP**, we can use similar arguments as made in Lemma 3 for the check equation in **IPP** to get

$$c\sigma = \gamma + 0\alpha^{2l-2} + (v+np)\alpha^{2l-1} + \lambda\alpha^{2l}$$

Focusing on the coefficients of α^{2l-2} and α^{2l-1}, we get two equations:

$$\langle \mathbf{d}, \mathbf{q}^+ \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u' = 0 \mod \alpha \tag{8}$$

$$\langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u = v + np \mod \alpha \tag{9}$$

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 $\mathbf{q}$ (chosen by the verifier $\in \mathbb{Z}_p$), we know that Equation 8's LHS over integers 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, $\langle \mathbf{d}, \mathbf{q}^+ \rangle = 0 \mod \alpha$ which implies $u = 0$ (due to Lemma 1). Hence, we can simplify Equation 9

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

   Since $M' - N'/\alpha$ must be an integer and $N' < \alpha \implies N' = 0$.

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

   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 $u = 1$. Now, write Equation 8 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, we get $M'\alpha - N' = M\alpha - N$ where $|N|, |N'| < 3pl$ and M' has denominator at most the LCM of all the k_i, which is at most p^l.

   We can apply Lemma 2 which implies that $M', N' \in \mathbb{Z}$ (since $\alpha > p^{2l}$). Then,

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

$$= \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + \left\lfloor M' - \frac{N'}{\alpha} \right\rfloor + 1 \mod \alpha$$
$$= \langle \mathbf{c}, \mathbf{q} \rangle \mod \alpha + M' - 1 + 1 \mod \alpha$$

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 and we satisfy extractability.

Non-interactivity using Fiat-Shamir. Protocol eval is public-coin and we can use the Fiat-Shamir heuristic [16] to obtain a non-interactive version in the ROM that has a constant-sized proof. The prover applies the RO on the commitment C to obtain the random query vector $\mathbf{z}$. Note that, the query vector $\mathbf{q}$ itself needs to be communicated, but the size of the proof is constant.

4 Dew – Constant-Sized PCS with Logarithmic Verifier

We prove our main result on PCS in this section. To go from IPC in §3 to a PCS of constant size and logarithmic verification time, we need two main ideas. First, we use Kronecker products test vectors to improve verification time from linear to logarithmic. But this breaks extractability of the new test. To recover extractability, we prove a new extremal combinatorial bound that enables us to prove a structure theorem despite the exponentially smaller randomness in the verifier's test vectors.

4.1 Dew: Our Polynomial Commitment Scheme

To construct Dew, we use ideas based on Kronecker products to define new query vectors in the **TEST** and **IPP** protocols from §3 and call the modified protocols **logTEST** and **logIPP**. These changes are to bring the verifier complexity down to logarithmic in the degree of the polynomial. For notational simplicity, let the degree of the polynomial $d = l - 1$. We change the blue messages in the **TEST**, **IPP** protocols as below. In the **logTEST** protocol, the query vector $\mathbf{z}$ in Fig. 2 is now redefined using just $2 \log l$ random elements in $\mathbb{Z}_p$:

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,

$$z_k \equiv z_{k_0, \ldots, k_{\log l - 1}} := \prod_{j=1}^{\log l} x_{j, k_{j-1}}. \tag{10}$$

For **logIPP**, the query vector $\mathbf{q}$ in Fig. 3 defined by the evaluation point $x \in \mathbb{Z}_p$ is modified as follows

$$q_k := \prod_{0 \le j \le \log l - 1} (x^{k_j 2^j} \mod p). \tag{11}$$

Note that $0 < z_k, q_k < p^{\log l}$ and $q_k \bmod p = x^k \bmod p$ for all k.

Our PCS $\mathsf{Dew} = (\mathsf{setup}, \mathsf{commit}, \mathsf{open}, \mathsf{eval})$ 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 $g \leftarrow\$ \mathbb{G}$. Define $\alpha := p^{2D \log D}$ (p is a large prime such that $\mathsf{len}(p) = \mathsf{poly}(\kappa)$).
  `Return` $\mathsf{pp} = (\kappa, \mathbb{G}, g, p)$.

– $\mathsf{commit}(\mathsf{pp}, D, f(X) \in \mathbb{Z}_p[X], l-1)$: Define the commitment $C := g^{\sum_{i=0}^{l-1} f_i \alpha^{2i}}$, where f_i are the coefficients of the degree $(l-1)$ polynomial $f(X)$ considered as integers from $[0, p-1]$ and the sum in $\mathbb{Z}$. If $l-1 \le D$, return $(C, \mathbf{f})$, else return error.

– $\mathsf{open}(\mathsf{pp}, D, f(X) \in \mathbb{Z}_p[X], l-1, C, \tilde{\mathbf{f}})$: Check that

  (i) $l - 1 \le D$, $\tilde{f}_i = f_i \mod p$ where $f_i \in \mathbb{Z}_p$ are the coefficients of $f(X)$.

  (ii) $C = g^{\sum_{i=0}^{l-1} \tilde{f}_i \alpha^{2i}}$, $\sum_{i=0}^{l-1} \tilde{f}_i \alpha^{2i} \in \mathbb{Z}$, and $\tilde{\mathbf{f}} \in \mathbb{Q}(\log l + 1, l + 1)^l$.
  Recall that

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

  where $|a|$ denotes the absolute value of the integer a.

  `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**:

  - `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** are simply variants of **TEST** and **IPP** in Figures 2 and 3 by replacing the (blue messages) query vectors with Kronecker products of shorter vectors as in (10) and (11). We also replace all expensive group exponentiations for the verifier by invocations of Wesolowski's PoE protocol. The full protocols thus obtained are presented as figures in the Appendix of [1]. In the NI version, the random query vector is derived from the RO instead of being sent in **logTEST**.

Non-interactive Dew 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 [16] to obtain a non-interactive version in the ROM that has constant-sized proof and logarithmic verification. The prover applies the RO on the commitment C to obtain $\mathbf{x}_1, \ldots \mathbf{x}_{\log l}$, and the random query vector $\mathbf{z}$ is computed as described in Eq (10). 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 the figures in the appendix of [1]). It is easy to see that the Fiat-Shamir transformed NI transcript is succinct since the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_{\log l}$ are now generated using the RO.

Proof of completeness is analogous (taking care of the new choices of parameters) to that of **IPP** in Theorem 31 and is deferred to the appendix of the full version [1]. Since the commitment scheme remains unchanged, the proof of binding remains as in Theorem 32. Proof of Extractability is shown in § 4.2, and proof of succinctness is given in § 4.3. The appendix of the full version contains concrete estimates of proof sizes. It also contains a section on how to achieve hiding and zero-knowledge evaluation for the commitment scheme.

4.2 Proof of Extractability of Dew

Define

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

as a subset of $\mathbb{Z}$ 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 $S_{\log}^q$:

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

Theorem 41 *The polynomial commitment scheme Dew satisfies Extractability (Def. 27) in the Generic Group Model.*

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

Theorem 42 *If the Verifier in* **logTEST** *outputs accept with non-negligible probability over the choice of random* $\mathbf{z_1}, \ldots, \mathbf{z_{\log l}} \in \mathbb{Z}_p^2$*, there exists an efficient extractor that outputs* $\mathbf{c}, \mathbf{d} \in [\alpha]^l$ *such that* $C = g^{\sum_{i=0}^{l-1}(c_i + \alpha d_i)\alpha^{2i}}$ *and* $\mathbf{d} \in S_{log}^l$*.*

Theorem 43 *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* $v = \tilde{f}(x)$*.*

The proof of Theorem 42 is presented in Section 4.2.2. The proof of Theorem 43 is almost identical to that of Theorem 35 and we defer it to the full version. There are only two changes: $d_i \in S_{log}$ instead of S implies that the extracted vector $\in \mathbb{Q}(\log l+1, l+1)^l$ instead of $\mathbb{Q}(2,3)^l$, and the bounds on N, N', M' are different, leading to a lower bound $\alpha > p^{2l(\log l)}$.

4.2.1 d-cancellation structures Before we present the proof of Theorem 42 in Section 4.2.2, we define certain combinatorial structures and state an extremal bound about them that plays a crucial role in our extractability proof. These are d-cancellation structures and are generalizations of d-dimensional hyper-rectangles /boxes. For instance, a 2-cancellation structure is a parallelogram, while a 3-cancellation 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 general d, we have the following recursive definition.

Definition 44 (d-cancellation structure) *Given a d-tuple $(a_1, \ldots, a_d) \in [n]^d$, a d-cancellation structure is defined to be the set of 2^d points mapped to the leaves of a depth-d binary tree, where the mapping from $[n]^d$ to nodes of the tree is recursively defined as follows.*

– *Map $(a_1, \ldots, a_d)$ to the root (depth 0).*

– *Suppose $(b_1, \ldots, b_d)$ is mapped to a node u at depth $d-j+1$. Then, for some $y_{u,j} \in [n]$, map $(b_1, \ldots, b_{j-1}, y_{u,j} + b_j, \ldots, b_d)$ to u's left child and $(b_1, \ldots, b_{j-1,}, y_{u,j}, \ldots, b_d)$ to u's right child.*

Informally, when we start from the same d-tuple, we get "similar" d-cancellation structures (which form an equivalence class; see an equivalent definition in the full version [1]). This is useful in counting arguments about them such as the one below.

Our main result on d-cancellation structures states that in the $[n]^d$ integer lattice, we can choose at most $n^d - (n-1)^d \le dn^{d-1}$ points that do not contain a d-cancellation structure.

Theorem 45 *The maximum number of points in $[n]^d$ such that no subset of them forms a d-cancellation structure is $N_d \coloneqq n^d - (n-1)^d$. This bound is tight.*

In the extractability proof in the next section, we will argue that if the prover succeeds with non-negligible probability, then it must populate an appropriately chosen lattice with more points than this bound, leading to the existence of a log l-cancellation structure. We then use higher-dimensional/multilinear analogs of ideas in Theorem 34 to induce cancellations among the l equations at the points in this log l-cancellation structure to derive an equation with a single **d** coordinate, thereby deducing the required structure for it. Specifically, we traverse the corresponding tree bottom-up (from leaves to root) by "folding" equations from one level to the next – subtract them pairwise to reduce their number by half and eliminate half of the remaining terms in each of them. Details of this process appear in the next section and in the full version [1].

4.2.2 Proof of Theorem 42 – logTEST extraction

Similar to the process in Lemma 3, we define the extractor Ext_T to first invoke the PoKPE extractor for C, which outputs $c > 0$ 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 $c_i = x_i$, $0 \le c_i \le p-1$ for $i \in [l]$ 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 $0 \le c_i, d_i \le \alpha - 1$.

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 – $\mathbf{d} \in S^l_{\log}$. Suppose the prover succeeds with a non-negligible probability over the random choice of $\mathbf{z_1}, \ldots, \mathbf{z_{\log l}}$ from $\mathbb{Z}_p^2$.
Fix an arbitrary index $0 \le k \le l-1$, 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

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

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 $\log l p^{\log l - 1} \ge p^{\log l} - (p-1)^{\log l}$ accepting points, which implies by Theorem 45 that there exists a $\log \mathsf{l\text{-}cancellation\ structure}$ in this space consisting of l accepting points.

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

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

All the points in B can be considered as the leaves of a binary tree, with leaves indexed as $g_1 g_2 \ldots g_{\log l}$. 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 $11\ldots1$, and the rightmost leaf will have index $00\ldots0$.

Now, Lemma 3 gives us equations corresponding to each accepting point on the $\log \mathsf{l\text{-}cancellation\ structure}$ relating $\mathbf{c}, \mathbf{d}$ (given by Ext_T) and the random variables x_{j,i_j}. We recall the definition of the query vector $\mathbf{z}$ from Equation (10), where for each coordinate z_i if the binary representation of $i = i_1 \ldots i_{\log l}$, then

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

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

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

$$\langle \mathbf{d}, \mathbf{z} \rangle \mod \alpha = \sum_{i_1, i_2, \ldots, i_{\log l} \in \{0,1\}} d_{i_1, i_2, \ldots, i_{\log l}} \cdot \prod_{j=1,}^{\log l} x_{j, i_j} \mod \alpha$$

$$= \sum_{i_1, \ldots, i_{\log l} \in \{0,1\}} d_{i_1, \ldots, i_{\log l}} \cdot \prod_{j=1}^{\log l} \left(y_{j, g_1, \ldots, g_{\log l - j}} + g_{\log l - j + 1} a_j \right) \cdot \prod_{\substack{j=1 \\ i_j \neq k_j}}^{\log l} x_{j, i_j} \mod \alpha$$

This expansion holds at height $\log l$ (for all leaves $g_1 g_2 \ldots g_{\log l}$). 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 $d_{i_1 \ldots i_{\log l}}$ 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 :

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

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

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

Notice that at the root, i.e., at height 0, we are left with the single term $a_1 \ldots a_{\log l} \cdot d_{k_1, k_2, \ldots, k_{\log l}}$.

For the rest of the folded 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 Theorem 34 generalised to higher dimensions.
Specifically, indexing the l points/leaves by $\mathbf{z}_{g_1 g_2 \ldots g_{\log l}}$, we get the expression for the remaining two terms (call this expression n) as

$$\left(\sum_{g_1, g_2, \ldots, g_{\log l} \in \{0,1\}} (-1)^{\sum_{j=1}^{\log l} i_j} \cdot \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}_{\mathbf{g_1}, \mathbf{g_2}, \ldots, \mathbf{g}_{\log 1}} \rangle}{\alpha} \right\rfloor + \right.$$

$$\left.\sum_{g_1,g_2,\ldots,g_{\log l}\in\{0,1\}} (-1)^{\sum_{j=1}^{\log l} i_j} \cdot u_{g_1,g_2,\ldots,g_{\log l}}\right) \mod \alpha$$

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

$$\frac{c_{k_1,\ldots,k_{\log l}} \cdot \prod_{i=1}^{\log l} a_i}{\alpha} - 2^{\log l} < n < \frac{c_{k_1,\ldots,k_{\log l}} \cdot \prod_{i=1}^{\log l} a_i}{\alpha} + 2^{\log l}$$
$$\implies -l < n < \prod_{i=1}^{\log l} a_i + l$$

Hence, there exists m such that $d_{k_1,\ldots,k_{\log l}} = \frac{m\alpha - n}{a_1 \cdots a_{\log l}}$ where $m \le \prod_{i=1}^{\log l} a_i < p^{\log l}$ (as $d_{k_1,\ldots,k_{\log l}} < \alpha$) and $-l < n < \prod_{i=1}^{\log l} a_i + l$ as shown above.

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

4.3 Succinctness of Dew

Theorem 46 (Proof and Verifier Succinctness) *In Dew, 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, as mentioned before, in the NI version, the $2\log l$ *random* field elements are generated using a RO – this makes the proof size of the non-interactive version of the protocol 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 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 (for σ in both **logTEST** and **logIPP**).

In **logTEST**, using the definition of the test vector in (10) and direct manipulation implies that $\sigma \mod q$ can be computed in $O(\log l)$ time as follows

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

In **logIPP**, by a similar manipulation as above using the definition of the the query vector in (11), we obtain

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

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 $L = O(l)$, computing $\alpha \mod q = p^L \mod q$ can be efficiently done in $O(\log l)$ time using repeated squaring. Once this is found, $\alpha^{-1} \mod q$ can also be efficiently found using the Extended Euclidean algorithm.

5 Transparent zkSNARKs via Dew

As a corollary of our PCS, we get concrete instantiations of new *transparent succinct arguments* by compiling an information theoretic proof in an idealized model 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 [9, 12, 17] 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 [12] uses an IOP abstraction called algebraic holographic proofs (AHP), and [9] 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 [17] 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) [11] 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. We present the details and compare our zkSNARK Dew-SNARK to existing schemes in the full version [1].

References

1. Arasu Arun, Chaya Ganesh, Satya Lokam, Tushar Mopuri, and Sriram Sridhar. Dew: Transparent constant-sized zkSNARKs. Cryptology ePrint Archive, Report

2022/419, 2022. https://eprint.iacr.org/2022/419.
2. Eli Ben-Sasson, Alessandro Chiesa, Lior Goldberg, Tom Gur, Michael Riabzev, and Nicholas Spooner. Linear-size constant-query IOPs for delegating computation. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 494–521. Springer, Heidelberg, December 2019.
3. 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.
4. Béla Bollobás. *Extremal Graph Theory*. Reprint of the 1978 original. Dover Publications, Inc., Mineola, NY, ISBN: 0-486-43596-2, 2004.
5. 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.
6. 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.
7. 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.
8. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. Cryptology ePrint Archive, Report 2019/1229, 2019. https://eprint.iacr.org/2019/1229.
9. 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.
10. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, Heidelberg, August 2002.
11. Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2021.
12. 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.
13. Hien Chu, Dario Fiore, Dimitris Kolonelos, and Dominique Schröder. Inner product functional commitments with constant-size public parameters and openings. In Clemente Galdi and Stanislaw Jarecki, editors, *Security and Cryptography for Networks*, pages 639–662, Cham, 2022. Springer International Publishing.
14. 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.
15. Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 427–438. IEEE, 1987.

16. 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.
17. 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.
18. 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.
19. 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.
20. Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 1–34. Springer, Heidelberg, November 2021.
21. 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.
22. Silvio Micali, Michael Rabin, and Joe Kilian. Zero-knowledge sets. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 80–91. IEEE, 2003.
23. 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.
24. 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.
25. 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.
26. 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.
27. 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.