

Couplet: Multivariate Polynomial Commitments with Two-Group-Element Openings

Weijie Wang¹, Alin Tomescu², Rex Fernando², and Charalampos
Papamanthou¹

¹ Yale University

² Aptos Labs

Abstract. We present *Couplet*, a new pairing-based multivariate polynomial commitment scheme (PCS). *Couplet* is the first multivariate PCS whose opening proofs are solely two group elements. In addition, for the case of multilinear polynomials, *Couplet* is the first constant-proof PCS with a prover that requires only a sublinear number of group operations. We achieve those properties simultaneously at the expense of a quasilinear, polynomial-specific, one-time preprocessing phase, whose cost can be amortized over multiple subsequent openings. On BN254, our implementation opens multilinear polynomials with 20 variables about $20\times$ faster than Mercury after preprocessing, breaking even in total prover time after roughly 100 openings. Our approach generalizes to batched evaluation proofs, yielding a constant-size proof for multiple evaluation points without using random oracles. Applications include verifying repeated inferences of a committed model as well as more efficient auditable authenticated dictionaries. Our main technique involves compressing Papamanthou–Shi–Tamassia (PST) evaluation proofs, giving a single commitment which encodes all $\log n$ quotient polynomials which are normally given out separately in a PST proof. In doing this, we reduce the verification to a univariate sumcheck problem, which we solve with techniques from Groth16 using a single additional group element. We prove extractability in the algebraic group model under a parameterized discrete-logarithm assumption. We give hiding variants and prove perfect zero-knowledge of the opening protocols. All opening protocols are noninteractive and require no random oracle.

1 Introduction

Polynomial commitment (PC) schemes [26] allow a prover to commit to a polynomial f and later prove an evaluation claim $f(z) = v$ succinctly. They are a core component of many zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs). While earlier zkSNARKs such as Plonk [19] and Marlin [14] were built around *univariate* PC schemes, more recent protocols such as Spartan [38], HyperPlonk [13], and Jolt [3] instead apply the sumcheck protocol [31] to *multilinear extensions* (MLEs) of the instance being proven, and

thus require *multilinear* PCs. The sumcheck-based approach is attractive because it avoids the costly polynomial divisions (and associated FFTs) that quotient-based univariate protocols perform during proof generation, enabling provers whose work is linear in the computation size [45,38,13]. These prover savings come with a tradeoff: the sumcheck rounds enlarge the proof, and multilinear PCs are more expensive than their univariate counterparts in both opening-proof size and verifier time. Nonetheless many deployed systems have adopted or migrated to sumcheck-based proving with multilinear commitments, including Aztec’s Honk provers [4] (following HyperPlonk [13]), Succinct’s SP1 Hypercube zkVM [24], Jolt [3], and Binius [16].

Existing multilinear PCs offer differing tradeoffs between evaluation-proof size and opening group work. The original multilinear PC by Papamanthou–Shi–Tamassia (PST) [35] requires the prover to commit to one quotient polynomial per variable. For a multilinear polynomial with $n = 2^\ell$ coefficients, this gives logarithmic proof size, and has evaluation proof computation costs roughly 2x that of classic univariate KZG [26]. More recent multilinear PC constructions, such as Hyrax [42] and KZH [25], are able to achieve much more prover-efficient evaluation proofs. Specifically, the logarithmic variant of KZH [25] uses a per-commitment preprocessing phase which reduces opening group work to *sublinear* in the polynomial description length, and retains the logarithmic proof size of PST. Another recent line of PC constructions [29,49,17,20] achieve constant size but require a linear number of group operations for computing the proof.

Our contribution. Our work studies whether it is possible to achieve a multilinear PC scheme which allows for fast (amortized) eval proof computation while also having constant-sized proofs. To that end, we present **Couplet**, a multilinear polynomial commitment scheme which improves over previous work in several dimensions:

- *Smallest known evaluation proofs.* A proof consists of two $\mathbb{G}_1$ elements—96 bytes on BLS12-381. To our knowledge, these are the smallest evaluation proofs among pairing-based multilinear PCs (Table 2). This does not require a commitment-size tradeoff: the single-opening scheme retains the one-element PST commitment. Verification uses three pairings and $O(\log n)$ scalar multiplications (Theorems 1 and 2). Notably, our evaluation proofs do *not* use Fiat-Shamir, in contrast to every other constant-sized eval proof for multilinear polynomials in the literature.
- *Sublinear opening group work.* After preprocessing a polynomial, each single-point opening uses two multiscalar multiplications (MSMs) of total size $O(\sqrt{n} \log n)$ and $O(n)$ field operations. The preprocessing costs $O(n \log^2 n)$ MSM terms and stores $O(\sqrt{n} \log n)$ auxiliary group elements per commitment, reusable across evaluation points. Our implementation (Section 1.3) shows approximately $7\times$ to $40\times$ concrete speedups during opening over other popular PCs, when computing multiple openings per commitment.
- *Constant-sized batched evaluation proofs.* A batch variant opens one committed polynomial at up to s arbitrary points with the same two-element proof size. Verification uses four pairings and $O(s \log n)$ scalar multiplica-

tions. Setup fixes the capacity s , which determines the key and preprocessing sizes; no verifier challenge is needed to combine the claims (Theorem 3). These multi-evaluation proofs also allow for preprocessing, with $O(s\sqrt{n} \log n)$ opening MSM terms for batches (Remark 6).

Core idea. In order to achieve two-element evaluation proofs, we compress PST evaluation proofs by combining univariate sumcheck with techniques underlying the classic Pinocchio and Groth16 [36,22] zkSNARKs. Recall that a classic PST evaluation proof gives out ℓ group elements which encode the following identity:

$$f(\mathbf{X}) - v = \sum_{k \in [\ell]} q_k(\mathbf{X})(X_k - z_k)$$

Instead of giving out these ℓ group elements, we first interpolate two polynomials $q(\mathbf{X}, Y)$ and $h(\mathbf{X}, Y)$ which encode the two components of *every* part of the sum, along the roots of unity of Y . This turns the evaluation identity into a univariate sum identity over the roots of unity involving just q and h , with q committed to by the prover and h computed by the verifier. Univariate sumcheck then reduces validity of the identity to membership in a low-dimensional polynomial subspace. Previous instantiations of univariate sumcheck themselves usually rely on polynomial commitment evaluation proofs, which would introduce a circular problem. To avoid this, we apply techniques from [36,22]: we encode a basis for this subspace in the trusted setup, shifted by some $1/\delta$ with δ secret, and also give out the group element $[\delta]_2$. This enables proof of membership in the subspace with a *single* additional group element. The two elements of the evaluation proof are then the commitment to q and this additional subspace-constrained element. Note that this subspace basis is *independent of the committed polynomial and evaluation point*, which crucially allows us to include it in the setup. This same strategy allows us to achieve constant-size batch evaluation proofs, and we explain the strategy further in Section 1.1.

Further extensions. Both single and batch openings extend to ℓ -variate polynomials of individual degree at most d , with the same proof sizes (Theorems 1 and 3). We also obtain a maintainable vector commitment that supports changes to individual entries. The prover stores a data structure of $O(n \log n)$ size. After an entry changes, updating the commitment and this data structure takes $O(\log^2 n)$ group work. Producing a two-element proof for any entry requires $O(\log^2 n)$ group work as well (Corollary 2). Finally, a coefficient embedding yields a univariate PC with sublinear opening group work after preprocessing, providing an alternative to KZG’s linear opening MSM size for repeatedly opened polynomials (Remark 5).

We prove extractability under the (q_1, q_2) -**dlog** assumption in the algebraic group model (AGM). The single and batch openings are directly noninteractive and require neither Fiat-Shamir nor a random oracle.

Applications. Our preprocessing is useful when the same committed polynomial is opened repeatedly at different points. Two motivating applications:

Repeated inference with a committed model. The fixed weight matrices of a model can be committed and preprocessed once, with the preprocessing reused across inference proofs. Consider a dense $m \times m$ weight matrix $\mathbf{W}$, where $n = m^2$ is the number of weights. Sumcheck-based verification of a matrix–vector product $\mathbf{y} = \mathbf{W}\mathbf{x}$ reduces the computation claim to polynomial evaluation claims, including an opening of the MLE of $\mathbf{W}$ at a general field point, as in zkLLM [41]. Although this point may change between inference proofs, our preprocessing depends only on $\mathbf{W}$ and the public key. Each subsequent opening of the weight polynomial therefore uses $O(\sqrt{n} \log n)$ MSM terms and $O(n)$ field operations, with a proof of two group elements.

Auditable dictionaries. Auditable dictionaries such as IronDict [23] motivate repeated openings of committed MLEs for verifiable lookups, as discussed in KZH [25]. For lookups against the same dictionary commitment, the prover can reuse its preprocessing across the required evaluation points. Our scheme provides two-element individual MLE opening proofs, compared with the log-size proofs of KZH’s logarithmic variant, at the cost of larger public keys and prover state. For a dictionary with 2^{20} entries stored in a table of 2^{22} slots, a collision-free lookup requires two polynomial openings. On BLS12-381, these proofs total 4,160 bytes for the non-hiding logarithmic KZH variant and 192 bytes for Couplet, excluding public claims and other dictionary data.

Organization. After the technical overview and related work, Section 2 gives preliminaries, Section 3 presents the multivariate construction, Section 4 shows how preprocessing reduces opening work for multilinear polynomials, and Section 5 extends the construction to batch openings. The appendices contain auxiliary lemmas and deferred proofs; Section B.4 gives the maintainable vector commitment.

1.1 Technical overview

Let $f \in \mathbb{F}[X_1, \dots, X_\ell]$ have degree at most d in each variable. Write $[a]_b = a \cdot G_b$ where $b \in \{1, 2, T\}$ for a bilinear group with pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$; we use additive notation. We first explain how to prove an evaluation $f(\mathbf{z}) = v$ with two group elements, omitting hiding. We then show how preprocessing reduces the group work needed to compute the proof and how this idea can extend to batch openings.

Combining the PST quotients. PST [35] proves an evaluation by expressing $f - v$ as a sum of terms, each divisible by one of the linear factors $X_k - z_k$:

$$f(\mathbf{X}) - v = \sum_{k=1}^{\ell} q_k(\mathbf{X})(X_k - z_k). \quad (1)$$

The quotients q_k exist satisfying Eq. (1) if and only if $v = f(\mathbf{z})$. PST has the prover send a commitment for each quotient, and has the verifier check the identity at a secret point chosen during setup via a pairing.

We compress these ℓ quotient commitments by introducing a variable Y that indexes the terms of the sum. Let ω be a primitive ℓ -th root of unity, and let $\mathcal{L}_k$

be the Lagrange polynomial that is one at ω^k and zero at the other interpolation points. We interpolate the quotients and the linear factors separately:

$$q(\mathbf{X}, Y) = \sum_{k=1}^{\ell} \mathcal{L}_k(Y) q_k(\mathbf{X}), \quad h(\mathbf{X}, Y) = \sum_{k=1}^{\ell} \mathcal{L}_k(Y) (X_k - z_k).$$

At $Y = \omega^k$, their product is exactly the k -th term of the PST identity. Thus Eq. (1) can be written as

$$\sum_{k=1}^{\ell} q(\mathbf{X}, \omega^k) h(\mathbf{X}, \omega^k) = f(\mathbf{X}) - v,$$

or equivalently, denoting $\Phi(Y) := q \cdot h - (f - v)/\ell$ with coefficients in $\mathbb{F}[\mathbf{X}]$,

$$\sum_{k=1}^{\ell} \Phi(\omega^k) = 0. \quad (2)$$

One core observation is that any polynomial $\Phi(Y)$ with $\deg_Y \Phi \leq 2\ell - 2$ satisfies the univariate sumcheck in Eq. (2) if and only if it lies in the subspace

$$\mathcal{V} := \{p(Y)(Y^\ell - 1) + r(Y)Y : \deg_Y p, \deg_Y r \leq \ell - 2\},$$

where all coefficients are in $\mathbb{F}[\mathbf{X}]$. To see this briefly, divide Φ by $Y^\ell - 1$. The remainder has degree less than ℓ and agrees with Φ at every root. Its sum over these roots is ℓ times its constant coefficient, since every nonconstant monomial cancels in the sum. The sum is therefore zero exactly when the remainder is divisible by Y . This is the multiplicative-subgroup form of the univariate sumcheck identity in Aurora [7, Remark 5.6]; Lemma 1 states the formal version we use.

Enforcing the identity with two group elements. The verifier must check both the identity $qh = \Phi + (f - v)/\ell$ and the required form of Φ , namely $\Phi \in \mathcal{V}$. Without the latter restriction, a prover could choose $q = 0$ and $\Phi = -(f - v)/\ell$, making the identity hold for any claimed value v . Note that we cannot directly apply previous instantiations of univariate sumcheck in the literature to check $\Phi \in \mathcal{V}$, since these instantiations mostly use polynomial commitment evaluation proofs to do the check, whereas here *we are building an evaluation proof*.

To do this, we use ideas from [36, 22]. Suppose the setup algorithm samples trapdoors τ for $\mathbf{X}$ and μ for Y , together with a nonzero secret δ . We will have the setup output elements $\{[g_i/\delta]_1\}_i$, where $\{g_i\}_i$ has a span inside $\mathcal{V}$ and generates all possible Φ .

The honest prover then sends

$$Q = [q(\tau, \mu)]_1, \quad P = [\Phi(\tau, \mu)/\delta]_1,$$

where P is generated via the basis above. Given the commitment $C = [f(\tau)]_1$, the verifier checks

$$e(Q, [h(\tau, \mu)]_2) = e(P, [\delta]_2) + e(C - v[1]_1, [1/\ell]_2) \quad (3)$$

where the pairing with δ causes the $1/\delta$ components in P to vanish. This equation then naturally verifies the identity $qh = \Phi + (f - v)/\ell$, and the secret δ restricts the prover such that $\Phi \in \mathcal{V}$. Specifically, the only way that the prover can prevent the RHS of Eq. (3) from having a δ component is to generate P solely from the elements $\{[g_i/\delta]_1\}_i$, and if the RHS has a δ component then it is impossible for the equality to hold.

Thus, we obtain a two-element evaluation proof with three pairings in verification. In doing this, we also constructed a single-group-element univariate sumcheck proof which *does not rely on an evaluation proof*. We believe this technique could be of independent interest.

We can make the evaluation proof zero-knowledge by randomizing Q and adjusting P to preserve Eq. (3), leaving C unchanged. Moreover, masking C also independently hides the commitment. These modifications preserve the one-element commitment and two-element proof; we describe the construction in Section 3.

Computing the proof with sublinear group work. We now show, in the multilinear setting, how to compute the above evaluation proof with sublinear group work, assuming a per-commitment preprocessing phase. We reduce the problem of efficiently computing P and Q to efficiently computing the commitments for the partial evaluations $f(\mathbf{z}_{[:k]}, \mathbf{X}_{[k+1:]})$ for all values of $k \in [\ell]$. The naive way of deriving this commitment to $f(\mathbf{z}_{[:k]}, \mathbf{X}_{[k+1:]})$ involves up to n group operations; we use the similar ideas as in KZH [25] to do this derivation with $O(\sqrt{n})$ group operations.

We briefly review the ideas here. For multilinear polynomials with $n = 2^\ell$ hypercube values, $f(\mathbf{z}_{[:k]}, \mathbf{X}_{[k+1:]})$ may be expanded in two different ways:

$$\begin{aligned} f(\mathbf{z}_{[:k]}, \mathbf{X}_{[k+1:]}) &= \sum_{\mathbf{i} \in \{0,1\}^k} \text{eq}(\mathbf{i}; \mathbf{z}_{[:k]}) \cdot f(\mathbf{i}, \mathbf{X}_{[k+1:]}) \\ &= \sum_{\mathbf{j} \in \{0,1\}^{\ell-k}} f(\mathbf{z}_{[:k]}, \mathbf{j}) \cdot \text{eq}(\mathbf{j}; \mathbf{X}_{[k+1:]}) . \end{aligned}$$

Notice that if $k > \ell/2$, computing a commitment from the second expansion corresponds to a sum of at most $\sqrt{n}$ group elements $[\text{eq}(\mathbf{j}; \boldsymbol{\tau}_{[k+1:]})]_1$, which are completely independent of the polynomial and the value $\mathbf{z}$. So in this case, assuming these group elements are embedded in the setup, we easily obtain a commitment to $f(\mathbf{z}_{[:k]}, \mathbf{X}_{[k+1:]})$ using $O(\sqrt{n})$ group work. When $k \leq \ell/2$, computing a commitment from the first expansion is a sum of at most $\sqrt{n}$ group elements $[f(\mathbf{i}, \boldsymbol{\tau}_{[k+1:]})]_1$. These group elements depend on f , but *do not* depend on the evaluation point $\mathbf{z}$. So if we precompute these group elements at commitment time, we also can obtain a commitment to $f(\mathbf{z}_{[:k]}, \mathbf{X}_{[k+1:]})$ using at most $O(\sqrt{n})$ group work.

We now explain how to compute Q and P efficiently, starting with Q . Recall Q is a commitment involving the PST quotient polynomials $\{q_k(\mathbf{X})\}_{k \in [\ell]}$. For multilinear polynomials, we show in Lemma 4 that q_k from Eq. (1) has the

following closed form:

$$q_k(\mathbf{X}) = f(\mathbf{z}_{[:k-1]}, 1, \mathbf{X}_{[k+1:]}) - f(\mathbf{z}_{[:k-1]}, 0, \mathbf{X}_{[k+1:]}) .$$

This means that we can compute Q using a sum of $2\ell = 2 \log n$ group elements, each committing to terms of the form $f(\mathbf{z}_{[:k]}, \mathbf{X}_{[k+1:]})$. By the above argument, we achieve this using $O(\sqrt{n} \log n)$ total group operations.

We finally discuss how to compute P efficiently. Recall that P is a commitment to the univariate sumcheck polynomial Φ . We derive a closed form for the polynomial Φ in terms of the same polynomials q_k and $f(\mathbf{z}_{[:k]}, \mathbf{X}_{[k+1:]})$ discussed above (Lemma 3). Note that even without preprocessing, this closed form provides an efficiency gain because it avoids directly expanding qh and dividing it by $Y^\ell - 1$. Since as we just explained q_k also reduces to values $f(\mathbf{z}_{[:k]}, \mathbf{X}_{[k+1:]})$, and there are at most $\log n$ such q_k , we again get that we can commit to Φ using $O(\sqrt{n} \log n)$ group operations.

All in all, with preprocessing, an opening uses $O(n)$ field operations and two MSMs of total size $O(\sqrt{n} \log n)$ (Section 4).

Batching over cosets. The same idea extends from a polynomial that sums to zero on one domain to a polynomial that sums to zero on *every coset* of a larger domain. Fix a batch capacity s at setup. To prove $f(\mathbf{z}_i) = v_i$ for every $i \in [s]$, partition a domain of $s\ell$ roots of unity into s cosets of size ℓ , one for each claim. On coset i , interpolate the PST quotients $q_{i,k}$ and linear factors $X_k - z_{i,k}$; interpolating across all cosets gives the polynomials q and h . Let $\mathcal{L}_i^*(Y^\ell)$ equal 1 on coset i and 0 on the others. Applying univariate sumcheck separately on each coset expresses the s PST identities as

$$qh = p \cdot (Y^{s\ell} - 1) + \sum_{i \in [s]} \mathcal{L}_i^*(Y^\ell) \left(r_i Y + \frac{f - v_i}{\ell} \right),$$

where $\deg_Y p \leq s\ell - 2$ and $\deg_Y r_i \leq \ell - 2$. The term $p \cdot (Y^{s\ell} - 1)$ and the weighted $r_i Y$ terms have zero sum on every coset. They form a subspace $\mathcal{V}_s$, whose membership is enforced by the same mechanism as before, preserving the two-element proof.

To ensure that all claims refer to the same polynomial, we use an additional setup secret γ to separate the commitment key from the proof keys. The commitment key contains no Y -dependent polynomials, so a prover cannot use different polynomials on different cosets.

Verification uses four pairings. For multilinear polynomials, the same preprocessing method gives $O(s\sqrt{n} \log n)$ opening MSM terms (Section 5 and Remark 6).

1.2 Related work

Evaluation-proof size. Table 2 shows two ways prior pairing-based multilinear PCs trade proof size against opening cost. PST, Gemini [9] (including HyperKZG [39]), and Zeromorph [27] have logarithmic-size proofs. Mercury [17],

Table 1. Costs of Couplet: the general scheme (Section 3) for $N = (d+1)^\ell$ coefficients, the multilinear scheme (Section 4) for $n = 2^\ell$ with even ℓ and $\kappa = \ell/2$, and the batch scheme (Section 5) for s points, in the multilinear case. Counts include the hiding terms and assume supported unpadded domains (Remark 1). In the verifier row, $\mathbb{G}_i$ counts scalar multiplications; in proof and storage rows, it counts elements. Opening group work is total MSM size in $\mathbb{G}_1$, and P denotes a pairing term. The auxiliary information is computed once per commitment and is private to the prover (Theorem 2 and Remark 6).

	General	Multilinear	Batch of s points
Proof	$2 \mathbb{G}_1$	$2 \mathbb{G}_1$	$2 \mathbb{G}_1$
Verifier	$3 \text{ P}, \ell \mathbb{G}_2, 1 \mathbb{G}_1$	$3 \text{ P}, \ell \mathbb{G}_2, 1 \mathbb{G}_1$	$4 \text{ P}, (s\ell + s) \mathbb{G}_2$
Opening, group	$N(\ell + 3) + 2\ell + 2$	$(2\ell + 5)\sqrt{n}$	$\approx 2(s\ell + 4s)\sqrt{n}$
Opening, field mult.	$O(N\ell)$	$n + O(\ell\sqrt{n} + \ell^2)$	$O(sn + s^2\ell\sqrt{n})$
Prover key	$4N\ell + 2\ell + 1 \mathbb{G}_1$	$(\ell + 3)n + \ell - 1 \mathbb{G}_1$	$\approx (s\ell + 4s)n \mathbb{G}_1$
Aux. per commitment	$1 \mathbb{G}_1 (D)$	$< (\ell + 3)\sqrt{n} \mathbb{G}_1$	$\approx (s\ell + 4s)\sqrt{n} \mathbb{G}_1$

SamaritanPCS [20], Chopin [6], Dolphin [29], Vela and Carina [49] achieve constant size, but all have larger proof payloads and linear total MSM size in their reported opening algorithms. Vela is the closest in proof size: its $2\mathbb{G}_1 + 3\mathbb{F}$ payload is 192 bytes on BLS12-381, twice our 96 bytes. Vela and Carina are non-hiding, whereas our two-element proof also supports hiding. Vela uses $O(n \log n)$ opening field work; our prover uses $O(n)$. Several competitors, including Vela and Dolphin, use only two pairing terms for verification and smaller structured keys.

Preprocessing and opening work. KZH [25] is the closest comparison for reusable preprocessing. Its logarithmic variant precomputes commitments to partial evaluations and achieves $O(\sqrt{n})$ opening group work with $O(\log n)$ proof size. We retain this prefix/suffix approach while compressing the proof to two elements. The trade-off is an additional $O(\log n)$ factor in opening MSM size and $O(n \log n)$ commitment and prover key elements, compared with KZH’s $O(n)$ keys. Both schemes retain auxiliary data at the prover, and both use $O(n)$ opening field work.

Multipoint batching. The standard method for batching multilinear polynomial evaluation proofs is to use sumcheck, as in HyperPlonk [13]. This method randomly combines multilinear evaluation claims and reduces them to one opening, with a logarithmic additional overhead. SamaritanPCS [20] instead amortizes the opening group work for s evaluations to $O(n + s\sqrt{n})$, while its batch proof grows linearly with s . In contrast, our coset construction keeps the proof at two group elements for up to the capacity fixed at setup, with no verifier challenge to combine the claims. Its prover key and preprocessing sizes grow with this capacity, but the evaluation points and claimed values remain public inputs.

Maintainable vector commitments. Hyperproofs [40] has logarithmic-size individual proofs and logarithmic update and query costs. BalanceProofs [44] reduces the individual proof to three group elements in its two-layer variant,

Table 2. Pairing-based multilinear PCs for $n = 2^\ell$ with even ℓ ; single opening. Proof sizes exclude the commitment and public evaluations. Opening MSM counts measure total scalar–base pairs, exclude commitment and preprocessing, and suppress lower-order terms. Verifier counts list pairing terms (P) and scalar multiplications, omitting constant $\mathbb{G}_1$ work. SRS counts give leading $\mathbb{G}_1$ sizes for commitment and prover keys. The balanced prover in **Couplet** additionally stores $O(\sqrt{n} \log n)$ auxiliary group elements per commitment (Theorem 2).

Scheme	Proof	Opening MSM	Verifier Group Work	SRS
PST [35]	$\ell \mathbb{G}_1$	$n - 1$	$(\ell+1) \text{ P}, \ell \mathbb{G}_2$	n
Gemini [9]	$(\ell+1) \mathbb{G}_1 + 2\ell \mathbb{F}$	$3n$	$2 \text{ P}, 1 \mathbb{G}_2, O(\ell) \mathbb{G}_1$	n
Zeromorph [†] [27]	$(\ell+2) \mathbb{G}_1$	$3n$	$2 \text{ P}, 1 \mathbb{G}_2, O(\ell) \mathbb{G}_1$	n
Mercury [17]	$8 \mathbb{G}_1 + 6 \mathbb{F}$	$2n + O(\sqrt{n})$	2 P	n
SamaritanPCS [20]	$7 \mathbb{G}_1 + 1 \mathbb{F}$	$2n + O(\sqrt{n})$	3 P	n
Chopin [6]	$7 \mathbb{G}_1 + 7 \mathbb{F}$	$n + 6\sqrt{n}$	$5 \text{ P}, 2 \mathbb{G}_2$	n
Dolphin [29]	$6 \mathbb{G}_1 + 1 \mathbb{F}$	$2n + 3\sqrt{n} - 3$	2 P	n
Dolphin _{AGM} [29]	$6 \mathbb{G}_1$	$2n + 3\sqrt{n} - 3$	2 P	n
Vela [49]	$2 \mathbb{G}_1 + 3 \mathbb{F}$	$2n - 3$	$2 \text{ P}, 1 \mathbb{G}_2$	n
Carina [49]	$4 \mathbb{G}_1 + 8 \mathbb{F}$	$n + 2\sqrt{n} - 6$	$3 \text{ P}, 2 \mathbb{G}_2$	n
KZH [25]	$\sqrt{n} \mathbb{G}_1 + \sqrt{n} \mathbb{F}$	0	$(\sqrt{n}+1) \text{ P}, 2\sqrt{n} \mathbb{G}_1$	n
KZH-log n [25]	$2(\ell-1) \mathbb{G}_1 + 2 \mathbb{F}$	$O(\sqrt{n})$	$3(\ell-1) \text{ P}, O(\ell) \mathbb{G}_1$	$O(n)$
Couplet , $\kappa = \ell/2$	$2 \mathbb{G}_1$	$(2\ell+5)\sqrt{n}$	$3 \text{ P}, \ell \mathbb{G}_2$	$(\ell+4)n$

Notes. Zeromorph[†] uses the non-hiding implementation analyzed in [49, Table 1]; the original hiding construction [27, Section 6] has an $(\ell+3)\mathbb{G}_1$ proof, approximately $5n/2$ opening MSM terms, and three pairings. SamaritanPCS and Dolphin use the optimized counts in [29, Table 2]. KZH follows [25, Figs. 2 and 12]: its square-root variant reuses cached row commitments, and its logarithmic verifier checks $\ell - 1$ three-term pairing equations.

whose update cost is $O(n^{1/4} \log n)$. Both schemes also support proof aggregation. Our specialization gives two-element individual proofs with polylogarithmic maintenance and query costs, at the cost of $O(n \log n)$ keys and auxiliary storage (Table 4). Other trade-offs appear in matrix and linear-map VCs [30, 46, 12]. Cauchyproofs [32] updates selected user proofs, a different interface from maintaining shared prover state for queries at any position.

Setup and security trade-offs. Transparent PCs avoid the trusted setup required by our construction. Inner-product approaches include Bulletproofs [11], Hyrax [42], and Dory [28]; Dory gives logarithmic proofs and verification using pairings without a structured reference string. Hash-based schemes such as Brakedown [21], BaseFold [47], WHIR [1], and Binius [16] also offer plausible post-quantum security and a range of field choices. Lattice-based PCs [34, 33] and unknown-order approaches [10, 2] provide further alternatives. Our construction requires a structured setup and pairing-friendly fields, and offers no post-quantum guarantee.

The noninteractive versions of Mercury, SamaritanPCS, Chopin, Vela, and Carina use Fiat–Shamir and the random-oracle model. Dolphin [29] establishes

computational special soundness for its interactive protocol under KZG binding and interpolation binding, with a random-oracle argument after Fiat–Shamir. Our opening protocols need no random oracle.

1.3 Implementation and benchmarks

We benchmark the implementation of Section 4 for one opening at $\kappa = \ell/2$, after polynomial preprocessing. Mercury [17] provides a constant-proof baseline; PST [35], HyperKZG [39], and Zeromorph [27] cover logarithmic-size proofs. For the other constant-size schemes in Table 2—SamaritanPCS, Chopin, both Dolphin variants, Vela and Carina—we did not locate public optimized implementations suitable for this same-machine benchmark.³

All runs use BN254, eight Rayon threads, and release builds on a Snapdragon X X1P64100 under Ubuntu 24.04/WSL2.⁴ All baselines run without hiding. We use padding parameter $t = 16, 18, 24$ for $\ell = 16, 18, 20$ (Remark 1).

Table 3. Single-opening time on BN254. Proof bytes exclude the commitment and evaluation claim.

Implementation	Opening time (ms)			Proof (bytes)
	$\ell = 16$	$\ell = 18$	$\ell = 20$	$\ell = 20$
Couplet	20.2	37.4	97.1	64
Mercury (Nova)	168.4	495.8	1 940.1	448
PST (HyperPlonk)	141.0	414.2	1 464.5	648
HyperKZG (Nova)	232.2	756.0	2 803.6	2 640
Zeromorph (Arecibo)	279.1	826.1	3 728.7	712

Results. At $\ell = 20$, the opening speedups are $20.0\times$ over Mercury and $38.4\times$ over Zeromorph. Our proof is one seventh the size of Mercury’s. KZH2 [25], using cached row commitments, opens faster in our measurements but produces a 64 KB proof at $\ell = 20$, compared with our 64 bytes. These comparisons include differences between arithmetic libraries.

Preparation costs. At $n = 2^{18}$, our preprocessing takes 28.5s and produces 0.69 MB of auxiliary information; the public commitment and prover keys occupy 369.1 MB uncompressed. At $n = 2^{20}$, the corresponding costs are 184.3s, 1.77 MB, and 1.88 GB. An opening uses the polynomial table, auxiliary information, and only the necessary keys.

Break-even openings. Let T_{commit} include commitment and any polynomial-specific preprocessing time, so that the total prover time for a openings is

³ A public [educational Python prototype of Samaritan](#) is available.

⁴ We use arkworks 0.5; Nova v0.76.0 (halo2curves 0.9); HyperPlonk 2a3b55c (arkworks 0.4); and Arecibo 8997d73 (halo2curves 0.6.1). Arecibo uses its pure-Rust MSM because its C++ backend did not build on ARM64.

$T_{\text{commit}} + aT_{\text{open}}$. Writing C for Couplet and B for a baseline, equating their total times and rounding up gives

$$a_{\min} = \left\lceil \frac{T_{\text{commit}}^C - T_{\text{commit}}^B}{T_{\text{open}}^B - T_{\text{open}}^C} \right\rceil.$$

For $\ell = 16, 18, 20$, respectively, Couplet breaks even after (46, 63, 101) openings against Mercury, (56, 76, 135) against PST, (32, 40, 69) against HyperKZG, and (26, 37, 51) against Zeromorph.

2 Preliminaries

Sets and vectors. For integers $m \leq n$, let $[n] = \{1, 2, \dots, n\}$ and $[m, n] = \{m, m+1, \dots, n\}$ ⁵. We use bold uppercase letters for vectors of variables, such as $\mathbf{X} = (X_1, \dots, X_\ell)$, and bold lowercase letters for vectors of field elements, such as $\mathbf{r} = (r_1, \dots, r_\ell) \in \mathbb{F}^\ell$. Inclusive bracket subscripts denote subvectors: $\mathbf{X}_{[i:k]} = (X_i, \dots, X_k)$ and $\mathbf{r}_{[i:j]} = (r_i, \dots, r_j)$. For two vectors $\mathbf{a}, \mathbf{b}$ of the same length, we write their inner product as $\mathbf{a}^\top \mathbf{b} := \sum_{i \in [|\mathbf{a}|]} a_i \cdot b_i$ and their product of powers as $\mathbf{a}^{\mathbf{b}} = \prod_{i \in [|\mathbf{a}|]} a_i^{b_i}$.

In particular, for a multi-index $\mathbf{i} \in [0, d]^\ell$ we write $\mathbf{X}^{\mathbf{i}} = \prod_{k \in [\ell]} X_k^{i_k}$. $\mathbb{F}_{\leq d}[\mathbf{X}]$ denotes the set of polynomials in $\mathbf{X}$ where each X_k has degree at most d .

Pairings. Let $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p)$ denote a *bilinear group* where

- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p .
- e is a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Using additive notation, for all $a, b \in \mathbb{Z}_p$ and all generators G_1 of $\mathbb{G}_1$ and G_2 of $\mathbb{G}_2$, the element $G_T := e(G_1, G_2)$ generates $\mathbb{G}_T$ and $e(a \cdot G_1, b \cdot G_2) = (ab) \cdot e(G_1, G_2) = (ab) \cdot G_T$.

We write $\mathbb{F} = \mathbb{Z}_p$ and $[a]_b := a \cdot G_b$ for $b \in \{1, 2, T\}$. We require an efficient algorithm $\text{GroupGen}(\mathcal{G})$ that returns independent uniformly random generators G_1 of $\mathbb{G}_1$ and G_2 of $\mathbb{G}_2$. Every setup algorithm below first samples $(G_1, G_2) \leftarrow \text{GroupGen}(\mathcal{G})$ independently of its trapdoors; these generators are public and implicit in the SRS. A multi-scalar multiplication $\text{MSM}(k)$ computes $\sum_{j=1}^k z_j \cdot U_j$ for scalars $z_j \in \mathbb{F}$ and elements U_j of one source group. We use *total MSM size* for the sum of the lengths of several MSMs. Unless stated otherwise, prover MSMs are in $\mathbb{G}_1$.

Lagrange polynomials. We assume that ℓ divides $p-1$, so that $\mathbb{F}$ contains a primitive ℓ -th root of unity ω . Batch openings of s points (Section 5) require

⁵ We interpret an index range whose upper endpoint is smaller than its lower endpoint as empty; empty slices are empty vectors, and empty sums and products are 0 and 1, respectively.

$s\ell \mid p-1$. See Remark 1 for padding when the required order is unavailable. The Lagrange basis polynomials for the ℓ points $\omega^1, \dots, \omega^\ell = 1$ are

$$\mathcal{L}_k(Y) = \frac{\omega^k(Y^\ell - 1)}{\ell(Y - \omega^k)} = \frac{1}{\ell} \sum_{j=0}^{\ell-1} \omega^{-jk} \cdot Y^j, \text{ for } k \in [\ell].$$

We use the following standard facts: $\mathcal{L}_k(\omega^{k'}) = 1$ if $k = k'$ and 0 otherwise; $\mathcal{L}_k(0) = 1/\ell$; $\sum_{k \in [\ell]} \mathcal{L}_k(Y) = 1$; and consequently, for every polynomial $g(Y)$ of degree at most $\ell - 1$ (over any commutative ring that contains $\mathbb{F}$), $g(Y) = \sum_{k \in [\ell]} g(\omega^k) \cdot \mathcal{L}_k(Y)$ and $\sum_{k \in [\ell]} g(\omega^k) = \ell \cdot g(0)$.

Univariate sumcheck. Let $\mathbb{R} = \mathbb{F}[\mathbf{X}]$, and write $\mathbb{R}_{\leq m}[Y]$ for the polynomials in $\mathbb{R}[Y]$ of degree at most m in Y , with $\mathbb{R}_{\leq m}[Y] = \{0\}$ for $m < 0$. For vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^\ell$ and their univariate interpolations $a(Y), b(Y)$, namely,

$$a(Y) := \sum_{k \in [\ell]} a_k \cdot \mathcal{L}_k(Y), \quad b(Y) := \sum_{k \in [\ell]} b_k \cdot \mathcal{L}_k(Y),$$

univariate sumcheck (a variant of [7]) checks whether

$$\mathbf{a}^\top \mathbf{b} = \sum_{k \in [\ell]} a(\omega^k) \cdot b(\omega^k) \stackrel{?}{=} t$$

for a given $t \in \mathbb{R}$, by checking membership of $a \cdot b - t/\ell$ in the subspace

$$\mathcal{V} := \{p(Y) \cdot (Y^\ell - 1) + r(Y) \cdot Y : p, r \in \mathbb{R}_{\leq \ell-2}[Y]\} \subseteq \mathbb{R}_{\leq 2\ell-2}[Y], \quad (4)$$

which has the following characterization:

Lemma 1. *Let ω be a primitive ℓ -th root of unity and $\mathbb{R}$ be the polynomial ring $\mathbb{F}[\mathbf{X}]$.*

1. *A polynomial $\Phi \in \mathbb{R}_{\leq 2\ell-2}[Y]$ lies in $\mathcal{V}$ if and only if $\sum_{k \in [\ell]} \Phi(\omega^k) = 0$, and every $\Phi \in \mathcal{V}$ has a unique representation $\Phi = p \cdot (Y^\ell - 1) + r \cdot Y$ with $p, r \in \mathbb{R}_{\leq \ell-2}[Y]$.*
2. *Consequently, for polynomials $a(Y), b(Y) \in \mathbb{R}[Y]$ of degree at most $\ell - 1$ and $t \in \mathbb{R}$, $\sum_{k \in [\ell]} a(\omega^k) \cdot b(\omega^k) = t$ if and only if there exist unique polynomials $p(Y), r(Y) \in \mathbb{R}[Y]$ such that*

$$a(Y) \cdot b(Y) = p(Y) \cdot (Y^\ell - 1) + r(Y) \cdot Y + \frac{1}{\ell} \cdot t,$$

where both $p(Y)$ and $r(Y)$ have degree at most $\ell - 2$.

Dividing by $Y^\ell - 1$ gives a remainder whose sum over the roots of unity depends only on its constant coefficient. The full proof is in Section B.1.

Multilinear extensions. Let $n = 2^\ell$. The MLE of a vector $\mathbf{v} = (v_1, \dots, v_n) \in \mathbb{F}^n$ is the ℓ -variate multilinear polynomial

$$f(\mathbf{X}) = \sum_{\mathbf{i} \in \{0,1\}^\ell} v_{\mathbf{i}} \cdot \text{eq}(\mathbf{i}; \mathbf{X})$$

where $\mathbf{i} = (i_1, \dots, i_\ell)$ is the binary representation of $i - 1 \in [0, n - 1]$, most significant bit first, and

$$\text{eq}(\mathbf{i}; \mathbf{X}) = \prod_{j \in [\mathbf{X}]} (i_j X_j + (1 - i_j)(1 - X_j)).$$

Polynomial commitments.

Definition 1 ((Multivariate) Polynomial Commitments). A polynomial commitment (PC) is a tuple of algorithms $\text{PC} = (\text{Setup}, \text{Commit}, \text{Eval}, \text{Verify})$ such that:

- $\text{PC.Setup}(1^\lambda, \ell, d) \rightarrow \text{srs} = (\text{ck}, \text{pk}, \text{vk})$: On input the security parameter, number of variables ℓ and degree bound d for each variable, output the commitment key ck , the prover key (for openings) pk and the verification key vk .
- $\text{PC.Commit}(\text{ck}, f) \rightarrow (C, \text{aux})$: On input the commitment key ck and a polynomial $f \in \mathbb{F}_{\leq d}[\mathbf{X}]$, output a commitment C and auxiliary information aux .
- $\text{PC.Eval}(\text{pk}, f, \mathbf{z}, v; \text{aux}) \rightarrow \pi$: On input the prover key pk , the polynomial f , the evaluation point $\mathbf{z} \in \mathbb{F}^\ell$, the claimed value $v \in \mathbb{F}$ and the auxiliary information aux , output an evaluation proof π that $v = f(\mathbf{z})$.
- $\text{PC.Verify}(\text{vk}, C, \mathbf{z}, v, \pi) \rightarrow 0/1$: On input the verification key vk , the commitment C , the evaluation point $\mathbf{z} \in \mathbb{F}^\ell$, the value v , and the proof π , output a bit indicating acceptance or rejection.

The scheme must satisfy the following properties:

Completeness. An honest prover always convinces the verifier. Formally, for any efficient adversary $\mathcal{A}$ that outputs $(f, \mathbf{z}) \in \mathbb{F}_{\leq d}[\mathbf{X}] \times \mathbb{F}^\ell$, we have

$$\Pr \left[\text{PC.Verify}(\text{vk}, C, \mathbf{z}, v, \pi) = 1 \mid \begin{array}{l} \text{srs} \leftarrow \text{PC.Setup}(1^\lambda, \ell, d) \\ (f(\mathbf{X}), \mathbf{z}) \leftarrow \mathcal{A}(\text{srs}), v := f(\mathbf{z}) \\ (C, \text{aux}) \leftarrow \text{PC.Commit}(\text{ck}, f) \\ \pi \leftarrow \text{PC.Eval}(\text{pk}, f, \mathbf{z}, v; \text{aux}) \end{array} \right] = 1.$$

Extractability. Whenever the prover outputs a commitment C and later provides a valid opening $(\mathbf{z}, v)$ of C , it knows, already at commitment time, a polynomial $f \in \mathbb{F}_{\leq d}[\mathbf{X}]$ such that $f(\mathbf{z}) = v$ (following [14, 19, 25]). Formally, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ there exists an efficient extractor $\mathcal{E}$ such that the following probability is negligible:

$$\Pr \left[\begin{array}{l} \text{PC.Verify}(\text{vk}, C, \mathbf{z}, v, \pi) = 1 \\ \wedge f(\mathbf{z}) \neq v \end{array} \mid \begin{array}{l} \text{srs} \leftarrow \text{PC.Setup}(1^\lambda, \ell, d) \\ (C, \text{st}; f) \leftarrow (\mathcal{A}_0 || \mathcal{E})(\text{srs}) \\ (\mathbf{z}, v, \pi) \leftarrow \mathcal{A}_1(\text{st}) \end{array} \right],$$

where $(\mathcal{A}_0 || \mathcal{E})$ denotes that $\mathcal{E}$ runs on the same input and random coins as $\mathcal{A}_0$ (in the AGM, it also receives the representation of C) and outputs $f \in \mathbb{F}_{\leq d}[\mathbf{X}]$.

In particular, $\mathcal{E}$ extracts f before receiving the opening point $\mathbf{z}$, claimed value v , or proof π .

Zero knowledge. The commitment and evaluation proofs reveal no information about the polynomial beyond the claimed evaluations (following the simulation-based formulation of [48, 45, 14]). Formally, there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$ such that for every PPT adversary $\mathcal{A}$ and every polynomial f with at most ℓ variables of degree at most d each, the outputs of the following two experiments are computationally indistinguishable. The scheme is perfectly zero-knowledge if these outputs are identically distributed. In both experiments, $\mathcal{A}$ adaptively queries evaluation points $\mathbf{z}_1, \mathbf{z}_2, \dots \in \mathbb{F}^\ell$ and receives, for each query, the value $v_i := f(\mathbf{z}_i)$ together with a proof π_i .

- $\text{Real}_{\mathcal{A},f}$: $\text{srs} \leftarrow \text{PC.Setup}(1^\lambda, \ell, d)$; $(C, \text{aux}) \leftarrow \text{PC.Commit}(\text{ck}, f)$; the i -th query $\mathbf{z}_i$ is answered with $\pi_i \leftarrow \text{PC.Eval}(\text{pk}, f, \mathbf{z}_i, v_i; \text{aux})$; output $b \leftarrow \mathcal{A}(\text{srs}, C, ((\mathbf{z}_i, v_i, \pi_i))_i)$.
- $\text{Ideal}_{\mathcal{A},f,\mathcal{S}}$: $(\text{srs}, \text{td}) \leftarrow \mathcal{S}_0(1^\lambda, \ell, d)$; $(C, \text{st}) \leftarrow \mathcal{S}_1(\text{td})$; the i -th query $\mathbf{z}_i$ is answered by $(\pi_i, \text{st}) \leftarrow \mathcal{S}_2(\text{td}, \text{st}, C, \mathbf{z}_i, v_i)$; output $b \leftarrow \mathcal{A}(\text{srs}, C, ((\mathbf{z}_i, v_i, \pi_i))_i)$.

The simulator never sees f and learns only the values v_i . With no queries, the definition requires the commitment alone to be hiding.

For a univariate polynomial $f(X)$, we write its KZG commitment [26] as

$$[f(\tau)]_b = f(\tau) \cdot G_b$$

where $b \in \{1, 2\}$ and τ is the trapdoor for X . Similarly, for a multivariate polynomial $f(\mathbf{X})$, we write its PST commitment [35] as $[f(\boldsymbol{\tau})]_b = f(\boldsymbol{\tau}) \cdot G_b$ where $\boldsymbol{\tau} = (\tau_1, \dots, \tau_\ell)$ is the vector of trapdoors for $\mathbf{X} = (X_1, \dots, X_\ell)$. The unmasked KZG and PST commitments are not hiding and do not satisfy the zero-knowledge definition above.

Cryptographic assumptions. We prove security in the algebraic group model (AGM) [18], using the (q_1, q_2) -**dlog** assumption. In the AGM, adversaries are *algebraic*: whenever $\mathcal{A}$ outputs a group element, it also outputs a representation as a linear combination of previously received group elements. In our asymmetric bilinear group setting, let $\mathbf{U}_1 = (U_{1,1}, \dots, U_{1,m_1}) \in \mathbb{G}_1^{m_1}$ and $\mathbf{U}_2 = (U_{2,1}, \dots, U_{2,m_2}) \in \mathbb{G}_2^{m_2}$ contain all group elements received by $\mathcal{A}$ so far. For each output $V \in \mathbb{G}_b$ ($b \in \{1, 2\}$), the adversary provides a vector $\boldsymbol{\mu}_V \in \mathbb{F}^{m_b}$ such that $V = \sum_{i \in [m_b]} (\mu_V)_i \cdot U_{b,i}$. The following (q_1, q_2) -**dlog** assumption is the *bilinear* version of the q -**dlog** assumption.

Assumption 2.1 $((q_1, q_2)$ -**dlog**) *Sample*

$$(G_1, G_2) \xleftarrow{\$} \text{GroupGen}(\mathcal{G}), \quad \alpha \xleftarrow{\$} \mathbb{F}.$$

For any PPT adversary $\mathcal{A}$ given

$$(G_1, \alpha \cdot G_1, \dots, \alpha^{q_1} \cdot G_1, G_2, \alpha \cdot G_2, \dots, \alpha^{q_2} \cdot G_2),$$

the probability that $\mathcal{A}$ outputs α , denoted $\text{Adv}_{\mathcal{A}}^{(q_1, q_2)\text{-dlog}}$, is negligible.

3 Multivariate polynomial commitments with two-element proofs

We construct **Couplet**, a commitment scheme for polynomials in ℓ variables of individual degree at most d , with proofs of two $\mathbb{G}_1$ elements verified by three pairings. Section 3.1 folds the ℓ divisibility conditions of PST [35] into a single polynomial identity using Lemma 1.

3.1 Folding the PST identities into one

Starting point: PST. PST certifies an evaluation $f(\mathbf{z}) = v$ through ℓ quotient polynomials, one per variable.

Lemma 2 (PST quotients). *Let $f \in \mathbb{F}_{\leq d}[\mathbf{X}]$ and $\mathbf{z} \in \mathbb{F}^\ell$. For $k \in [0, \ell]$, let*

$$F_k(\mathbf{X}) := f(\mathbf{z}_{[:k]}, \mathbf{X}_{[k+1:]})$$

denote the partial evaluation of f at the first k coordinates of $\mathbf{z}$, so that $F_0 = f$ and $F_\ell = f(\mathbf{z})$. For every $k \in [\ell]$, the polynomial $F_{k-1} - F_k$ is divisible by $X_k - z_k$, and the quotients

$$q_k(\mathbf{X}) := \frac{F_{k-1}(\mathbf{X}) - F_k(\mathbf{X})}{X_k - z_k}, \quad k \in [\ell], \quad (5)$$

satisfy

$$f(\mathbf{X}) - f(\mathbf{z}) = \sum_{k \in [\ell]} q_k(\mathbf{X}) \cdot (X_k - z_k). \quad (6)$$

The quotient q_k does not involve $X_1, \dots, X_{k-1}$, has degree at most $d-1$ in X_k , and degree at most d in every other variable.

Proof. Substituting $X_k := z_k$ in F_{k-1} gives F_k , so $F_{k-1} - F_k$ vanishes at $X_k = z_k$. Viewed as a polynomial in X_k over $\mathbb{F}[\mathbf{X}_{[k+1:]}]$, it is therefore divisible by the monic polynomial $X_k - z_k$, with quotient of degree at most $d-1$ in X_k . Eq. (6) is the telescoping sum $f - f(\mathbf{z}) = \sum_{k \in [\ell]} (F_{k-1} - F_k)$. $\square$

The PST proof for $f(\mathbf{z}) = v$ consists of the ℓ commitments $[q_k(\boldsymbol{\tau})]_1$, and the verifier checks Eq. (6) at the trapdoor with $\ell + 1$ pairings, namely

$$e(C - v \cdot [1]_1, [1]_2) = \sum_{k \in [\ell]} e([q_k(\boldsymbol{\tau})]_1, [\tau_k - z_k]_2),$$

where $C = [f(\boldsymbol{\tau})]_1$.

One variable for ℓ polynomials. We interpolate the ℓ quotients and the ℓ linear factors $X_k - z_k$ in a new variable Y to obtain q and h . We also define an auxiliary polynomial u :

$$\begin{aligned} q(\mathbf{X}, Y) &:= \sum_{k \in [\ell]} \mathcal{L}_k(Y) \cdot q_k(\mathbf{X}), \\ u(\mathbf{X}, Y) &:= \sum_{k \in [\ell]} \mathcal{L}_k(Y) \cdot X_k, \quad h(\mathbf{X}, Y) := u(\mathbf{X}, Y) - u(\mathbf{z}, Y). \end{aligned} \quad (7)$$

By construction, $q(\mathbf{X}, \omega^k) = q_k(\mathbf{X})$ and $h(\mathbf{X}, \omega^k) = X_k - z_k$, so Eq. (6) becomes

$$\sum_{k \in [\ell]} q(\mathbf{X}, \omega^k) \cdot h(\mathbf{X}, \omega^k) = f(\mathbf{X}) - v. \quad (8)$$

Both q and h have degree at most $\ell - 1$ in Y , so the univariate sumcheck of Lemma 1 (with $t = f - v$) applies: Eq. (8) holds if and only if

$$q(\mathbf{X}, Y) \cdot h(\mathbf{X}, Y) - \frac{1}{\ell} \cdot (f(\mathbf{X}) - v) \in \mathcal{V}, \quad \text{i.e., } q \cdot h = p \cdot (Y^\ell - 1) + r \cdot Y + \frac{1}{\ell} \cdot (f - v) \quad (9)$$

for unique $p(\mathbf{X}, Y), r(\mathbf{X}, Y) \in \mathbb{R}_{\leq \ell-2}[Y]$.

Lemma 3 (Closed forms). *Let $f, \mathbf{z}, F_k$ and q_k be as in Lemma 2, $v := f(\mathbf{z})$, and q, u, h as in Eq. (7). Define, for $k \in [\ell]$,*

$$U_k(\mathbf{X}, Y) := \mathcal{L}_k(Y) \cdot (u(\mathbf{X}, Y) - X_k), \quad Z_k(Y) := \mathcal{L}_k(Y) \cdot (u(\mathbf{z}, Y) - z_k), \quad (10)$$

and $M_0(Y) := \mathcal{L}_1(Y) - \frac{1}{\ell}$, $M_k(Y) := \mathcal{L}_{k+1}(Y) - \mathcal{L}_k(Y)$ for $k \in [\ell - 1]$, $M_\ell(Y) := \frac{1}{\ell} - \mathcal{L}_\ell(Y)$. Then:

1. U_k and Z_k are divisible by $Y^\ell - 1$ with quotients in $\mathbb{R}_{\leq \ell-2}[Y]$, and M_k is divisible by Y with quotient of degree at most $\ell - 2$. Moreover, $U_k, Z_k, M_k \in \mathcal{V}$.
2. The polynomials p, r of Eq. (9) satisfy

$$\begin{aligned} p(\mathbf{X}, Y) \cdot (Y^\ell - 1) &= \sum_{k \in [\ell]} q_k(\mathbf{X}) \cdot (U_k(\mathbf{X}, Y) - Z_k(Y)), \\ r(\mathbf{X}, Y) \cdot Y &= \sum_{k=0}^{\ell} F_k(\mathbf{X}) \cdot M_k(Y). \end{aligned} \quad (11)$$

3. Every variable X_k has degree at most $d + 1$ in p and at most d in r .

The divisibility claims follow because U_k and Z_k vanish on the interpolation domain and $M_k(0) = 0$. The formula for $r \cdot Y$ follows by summation by parts. The full proof is in Section B.1.

By Lemma 3(1), U_k, Z_k, M_k and their products with monomials in $\mathbf{X}$ lie in $\mathcal{V}$, as required for the numerators of the prover-key elements carrying $1/\delta$. The factors U_k and M_k are independent of $\mathbf{z}$. To express Z_k using fixed polynomials as well, we write

$$Z_k(Y) = (Y^\ell - 1) \cdot \sum_{m=0}^{\ell-2} w_{k,m} \cdot Y^m, \quad \text{i.e., } \sum_{m=0}^{\ell-2} w_{k,m} Y^m = \frac{\omega^k}{\ell} \cdot \frac{u(\mathbf{z}, Y) - z_k}{Y - \omega^k}. \quad (12)$$

To compute the coefficients $w_{k,m}$, first expand $u(\mathbf{z}, Y)$ in the monomial basis, then divide $u(\mathbf{z}, Y) - z_k$ by $Y - \omega^k$ for each k and scale the quotient by ω^k/ℓ . The expansion takes $O(\ell^2)$ field operations directly, and each division takes $O(\ell)$, giving $O(\ell^2)$ overall.

3.2 The scheme

Design. Let $\tau \in \mathbb{F}^\ell$ and μ be trapdoors for $\mathbf{X}$ and Y , and let δ be an additional trapdoor used as a tag. The commitment is the PST commitment $C = [f(\tau)]_1$, and the verification key lets the verifier compute $[h(\tau, \mu)]_2 = [u(\tau, \mu)]_2 - \sum_k z_k \cdot [\mathcal{L}_k(\mu)]_2$ with ℓ scalar multiplications in $\mathbb{G}_2$. The proof consists of

$$Q = [q(\tau, \mu)]_1 \quad \text{and} \quad P = [\Phi(\tau, \mu)/\delta]_1 \quad \text{where} \quad \Phi := p \cdot (Y^\ell - 1) + r \cdot Y \in \mathcal{V},$$

and the verifier checks Eq. (9) at the trapdoors through

$$e(Q, [h(\tau, \mu)]_2) \stackrel{?}{=} e(P, [\delta]_2) + e(C - v \cdot [1]_1, [1/\ell]_2),$$

where $[1]_1 = G_1$ and $[1/\ell]_2 = \ell^{-1} \cdot G_2$ are public.

For zero knowledge, the prover adds a random multiple $\eta \cdot [\delta]_1$ to Q and compensates by adding $\eta \cdot [h(\tau, \mu)]_1$ to P . Following vSQL [48], the committer adds a random multiple $\eta' \cdot [\delta]_1$ to C and retains the hint $D = \frac{\eta'}{\ell} \cdot [1]_1$ as auxiliary information. The prover subtracts this hint from P in every opening. In the exponent, the verification equation becomes

$$(q(\tau, \mu) + \eta \delta) \cdot h(\tau, \mu) = \left(\frac{\Phi(\tau, \mu)}{\delta} + \eta \cdot h(\tau, \mu) - \frac{\eta'}{\ell} \right) \cdot \delta + \frac{1}{\ell} (f(\tau) - v + \eta' \delta), \quad (13)$$

and both sides equal $q \cdot h + \eta \delta h$ at (τ, μ) by Eq. (9). **Orange** marks the terms that are only needed for zero knowledge; dropping them yields a non-hiding scheme with the same extractability proof.

Keys. The prover-key families in Fig. 1 encode the Lagrange basis for Q and the U_k , Z_k and M_k contributions to P in Eq. (11); the Z_k contributions use the monomial basis in Eq. (12). The monomials in p that have degree $d + 1$ in one variable (Lemma 3) are accounted for by U_k , so the prover key need not list them separately.

3.3 Security

Theorem 1. *Let $\ell, d \geq 1$ with $\ell \mid p - 1$. Couplet in Fig. 1 is a polynomial commitment for $\mathbb{F}_{\leq d}[\mathbf{X}]$ (Definition 1) with the following properties:*

1. *it is complete;*
2. *it is extractable in the AGM under the $(d\ell + 2\ell - 1, \ell + 1)$ -dlog assumption (Assumption 2.1). For every algebraic PPT adversary $\mathcal{A}$, there are an efficient extractor $\mathcal{E}$, which uses only the representation supplied with C , and an expected-polynomial-time discrete-log adversary $\mathcal{B}$ such that the extractability error in Definition 1 is at most*

$$\text{Adv}_{\mathcal{B}}^{(d\ell + 2\ell - 1, \ell + 1)\text{-dlog}} + \frac{d\ell + 3\ell + 1}{p}.$$

The reduction $\mathcal{B}$ runs $\mathcal{A}$ once, simulates the SRS, and finds the roots of a univariate polynomial of degree at most $d\ell + 3\ell$;

Setup($1^\lambda, \ell, d$) $\rightarrow$ (ck, pk, vk):

- Pick $\tau = (\tau_1, \dots, \tau_\ell) \xleftarrow{\$} \mathbb{F}^\ell$, $\mu \xleftarrow{\$} \mathbb{F}$ and $\delta \xleftarrow{\$} \mathbb{F}^*$. Let U_k, M_k be as in Lemma 3.
- Output, with i ranging over $[0, d]^\ell$ and k over $[\ell]$ unless stated otherwise,

$$\text{ck} = \left(([\tau^i]_1)_i, [\delta]_1 \right),$$

$$\text{pk} = \left(\begin{array}{c} [\delta]_1, ([\mathcal{L}_k(\mu)]_1, [\tau_k \mathcal{L}_k(\mu)]_1)_k, \\ ([\tau^i \mathcal{L}_k(\mu)]_1)_{i,k}, \left(\left[\frac{\tau^i U_k(\tau, \mu)}{\delta} \right]_1 \right)_{i,k}, \\ \left(\left[\frac{\tau^i \mu^m (\mu^\ell - 1)}{\delta} \right]_1 \right)_{i,m}, \left(\left[\frac{\tau^i M_{k'}(\mu)}{\delta} \right]_1 \right)_{i,k'} \end{array} \right),$$

$$\text{vk} = \left([\delta]_2, ([\mathcal{L}_k(\mu)]_2)_k, H_\tau := [u(\tau, \mu)]_2 = \left[\sum_{k \in [\ell]} \tau_k \mathcal{L}_k(\mu) \right]_2 \right);$$

where $m \in [0, \ell - 2]$ and $k' \in [0, \ell]$. The verifier also uses the public elements $[1]_1 = G_1$ and $[1/\ell]_2 = \ell^{-1} \cdot G_2$.

Commit(ck, f) $\rightarrow$ (C , aux):

- Write $f = \sum_i f_i \mathbf{X}^i$. Pick $\eta' \xleftarrow{\$} \mathbb{F}$.
- Output $C = \sum_i f_i \cdot [\tau^i]_1 + \eta' \cdot [\delta]_1$ and $\text{aux} = D := \frac{\eta'}{\ell} \cdot [1]_1$.

Eval(pk, $f, \mathbf{z}, v$; aux) $\rightarrow \pi$:

- Compute $F_k = \sum_i F_{k,i} \mathbf{X}^i$ ($k \in [0, \ell]$) and $q_k = \sum_i q_{k,i} \mathbf{X}^i$ ($k \in [\ell]$) as in Lemma 2, and the coefficients $w_{k,m}$ of Eq. (12). Parse D from aux and pick $\eta \xleftarrow{\$} \mathbb{F}$.
- Output $\pi = (Q, P)$ with

$$Q = \sum_{k \in [\ell]} \sum_i q_{k,i} \cdot [\tau^i \mathcal{L}_k(\mu)]_1 + \eta \cdot [\delta]_1,$$

$$\begin{aligned} P = & \sum_{k \in [\ell]} \sum_i q_{k,i} \cdot \left(\left[\frac{\tau^i U_k(\tau, \mu)}{\delta} \right]_1 - \sum_{m=0}^{\ell-2} w_{k,m} \cdot \left[\frac{\tau^i \mu^m (\mu^\ell - 1)}{\delta} \right]_1 \right) \\ & + \sum_{k=0}^{\ell} \sum_i F_{k,i} \cdot \left[\frac{\tau^i M_k(\mu)}{\delta} \right]_1 \\ & + \eta \cdot \sum_{k \in [\ell]} ([\tau_k \mathcal{L}_k(\mu)]_1 - z_k \cdot [\mathcal{L}_k(\mu)]_1) - D, \end{aligned}$$

i.e., $Q = [q(\tau, \mu) + \eta \delta]_1$ and $P = [\Phi(\tau, \mu)/\delta + \eta h(\tau, \mu) - \eta'/\ell]_1$.

Verify(vk, $C, \mathbf{z}, v, \pi$) $\rightarrow$ 0/1:

- Compute $H_z = \sum_{k \in [\ell]} z_k \cdot [\mathcal{L}_k(\mu)]_2$, so that $H_\tau - H_z = [h(\tau, \mu)]_2$.
- Accept if and only if

$$e(Q, H_\tau - H_z) = e(P, [\delta]_2) + e(C - v \cdot [1]_1, [1/\ell]_2).$$

Fig. 1. Couplet for multivariate polynomials, with two-element opening proofs. Orange marks the terms needed only for zero knowledge.

3. *it is perfectly zero-knowledge.*

Proofs consist of two $\mathbb{G}_1$ elements. Verification takes ℓ scalar multiplications in $\mathbb{G}_2$, one in $\mathbb{G}_1$, and three pairings.

Completeness follows from Eq. (13). For zero knowledge, the masks make C and each fresh Q uniform. Given these elements, the public claim and the trapdoors, the simulator computes the unique valid P . The same simulation applies to adaptive repeated openings of a fixed polynomial. Full proofs are in Section B.2.

Proof overview. Let $c_E(\mathbf{X}, Y)$ and $q_E(\mathbf{X}, Y)$ be the δ -free components of the formal representations of C and Q , as in Eq. (27). The extractor outputs $f := c_E(\mathbf{X}, 0) \in \mathbb{F}_{\leq d}[\mathbf{X}]$ from the representation supplied with C , before receiving an opening. Every key element carrying $1/\delta$ has its numerator in $\mathcal{V}$, so the numerator p_D of the $1/\delta$ component of P also lies in $\mathcal{V}$. By Lemma 8 and Corollary 1, acceptance implies the corresponding Laurent-polynomial identity, except with the error bounded in Theorem 1. Comparing its δ -free terms gives

$$q_E \cdot h = p_D + \frac{1}{\ell}(c_E - v).$$

Summing over the roots of unity eliminates p_D by Lemma 1. Since c_E has degree at most $\ell - 1$ in Y , its average over these roots is $c_E(\mathbf{X}, 0) = f(\mathbf{X})$. Thus

$$\sum_{k \in [\ell]} q_E(\mathbf{X}, \omega^k) \cdot (X_k - z_k) = f(\mathbf{X}) - v.$$

Substituting $\mathbf{X} = \mathbf{z}$ yields $f(\mathbf{z}) = v$.

3.4 Efficiency and remarks

Costs. Let $N = (d+1)^\ell$ be the number of coefficients of f . In the hiding scheme of Fig. 1, the commitment key has $N+1$ elements of $\mathbb{G}_1$, and the verification key has $\ell+2$ elements of $\mathbb{G}_2$. The prover key lists $4N\ell + 2\ell + 1$ elements of $\mathbb{G}_1$, with at most $4N\ell + 1$ remaining after duplicate entries are removed. Computing the commitment C and the hint D takes one $\text{MSM}(N)$ and two scalar multiplications in $\mathbb{G}_1$.

The partial evaluations F_k and quotients q_k have $O(N)$ coefficients in total and can be computed with $O(N)$ field operations (Lemma 2). Forming Q and P takes at most $N(\ell+3) + 2\ell + 2$ MSM terms in $\mathbb{G}_1$ and $O(N\ell)$ field operations. For multilinear polynomials, preprocessing reduces the opening group work to $O(\sqrt{N} \log N)$ MSM terms (Section 4).

Remark 1 (Interpolation domains). Single openings require $\ell \mid p-1$, and batches of capacity s require $s\ell \mid p-1$ (Section 5). If ℓ is unsupported, choose $t \geq \ell$ with $t \mid p-1$ and pad the interpolation domain: use size- t Lagrange polynomials, set the quotients and linear factors to zero for $k > \ell$, and extend $F_k = v$ to

those indices. Replace $1/\ell$ by $1/t$ and define M_k on the padded domain. No polynomial variables are added. The prover key, the work involving Y , and the reduction degree bounds grow with t ; the commitment key, proof size and number of pairings are unchanged. The verifier still uses only ℓ nonzero linear factors. When supported, padding to the next power of two gives $t < 2\ell$.

For batches, choose $s' \geq s$ and $t \geq \ell$ with $s't \mid p-1$, repeat an existing claim in the extra slots, and apply the same zero padding within each size- t coset. Costs and reduction bounds then depend on s' and t .

Remark 2 (Bases of the prover key). The prover-key families in Fig. 1 may be replaced by any collection of $\mathbb{G}_1$ elements of the three types in Eq. (26) that spans the honest prover's outputs. If the total-degree bounds after clearing the common denominator do not increase and the $\mathbb{G}_2$ elements are unchanged, the extractability bound in Theorem 1 is unchanged. Retaining the masking elements also preserves zero knowledge. The multilinear prover in Section 4 uses such a basis.

Remark 3 (Same-point aggregation). Fix a claims $(C_j, \mathbf{z}, v_j)$ at the same point before sampling $\rho \xleftarrow{\$} \mathbb{F}$. By linearity of the verification equation, their proofs (Q_j, P_j) combine into $(\sum_{j=1}^a \rho^{j-1} Q_j, \sum_{j=1}^a \rho^{j-1} P_j)$, which opens $\sum_{j=1}^a \rho^{j-1} C_j$ at $\mathbf{z}$ to $\sum_{j=1}^a \rho^{j-1} v_j$. A false collection of fixed claims passes this check with probability at most $(a-1)/p$, apart from the underlying extractability error.

4 A prover with sublinear group work for multilinear polynomials

Many SNARKs commit to MLEs $f \in \mathbb{F}_{\leq 1}[\mathbf{X}]$ given by their $n = 2^\ell$ values on the hypercube, $f(\mathbf{X}) = \sum_{\mathbf{i} \in \{0,1\}^\ell} f(\mathbf{i}) \cdot \text{eq}(\mathbf{i}; \mathbf{X})$. For such polynomials, the general prover for Couplet in Fig. 1 uses opening MSMs of total size at most $n(\ell+3) + 2\ell + 2$. We use the structure of MLEs to reduce this cost to $O(\sqrt{n} \log n)$ MSM terms per opening, with $O(\sqrt{n} \log n)$ auxiliary group elements computed once per commitment.

4.1 Quotients and partial evaluations of MLEs

Lemma 4 (Quotients of MLEs). *Let $f \in \mathbb{F}_{\leq 1}[\mathbf{X}]$, $\mathbf{z} \in \mathbb{F}^\ell$, and let F_k and q_k be as in Lemma 2. Then, for $k \in [\ell]$,*

$$q_k(\mathbf{X}) = f(\mathbf{z}_{[:k-1]}, 1, \mathbf{X}_{[k+1:]}) - f(\mathbf{z}_{[:k-1]}, 0, \mathbf{X}_{[k+1:]}) , \quad (14)$$

and F_k and q_k are multilinear polynomials in $\mathbf{X}_{[k+1:]}$.

Proof. F_{k-1} is multilinear in X_k , so $F_{k-1} = (1 - X_k) \cdot F_{k-1}|_{X_k=0} + X_k \cdot F_{k-1}|_{X_k=1}$ and $F_k = (1 - z_k) \cdot F_{k-1}|_{X_k=0} + z_k \cdot F_{k-1}|_{X_k=1}$. Subtracting, $F_{k-1} - F_k = (X_k - z_k) \cdot (F_{k-1}|_{X_k=1} - F_{k-1}|_{X_k=0})$, which is Eq. (14) by Eq. (5). $\square$

We call a polynomial in $\mathbf{X}_{[k+1:]}$ a *level- k* polynomial. Each level- k polynomial used by the prover admits two expansions. The prefix expansion uses 2^{k-1} polynomials for q_k and 2^k for F_k ; these polynomials depend on f but not on $\mathbf{z}$. The suffix expansion uses $2^{\ell-k}$ Boolean basis polynomials that depend on neither f nor $\mathbf{z}$. Below, $\mathbf{i}$ denotes a *prefix* in $\{0, 1\}^{k-1}$ or $\{0, 1\}^k$, and $\mathbf{j}$ a *suffix* in $\{0, 1\}^{\ell-k}$.

Lemma 5 (Two-sided expansions). *Let $f \in \mathbb{F}_{\leq 1}[\mathbf{X}]$ and $\mathbf{z} \in \mathbb{F}^\ell$. For $k \in [\ell]$ define the $\mathbf{z}$ -independent level- k polynomials*

$$\hat{q}_{k,\mathbf{i}}(\mathbf{X}) := f(\mathbf{i}, 1, \mathbf{X}_{[k+1:]}) - f(\mathbf{i}, 0, \mathbf{X}_{[k+1:]}) , \quad \mathbf{i} \in \{0, 1\}^{k-1} ,$$

and for $k \in [0, \ell]$ the polynomials $\hat{F}_{k,\mathbf{i}}(\mathbf{X}) := f(\mathbf{i}, \mathbf{X}_{[k+1:]})$, $\mathbf{i} \in \{0, 1\}^k$ (so $\hat{F}_{0,()}(f) = f$). Then

$$q_k(\mathbf{X}) = \sum_{\mathbf{i} \in \{0, 1\}^{k-1}} \text{eq}(\mathbf{i}; \mathbf{z}_{[:k-1]}) \cdot \hat{q}_{k,\mathbf{i}}(\mathbf{X}) = \sum_{\mathbf{j} \in \{0, 1\}^{\ell-k}} q_k(\mathbf{j}) \cdot \text{eq}(\mathbf{j}; \mathbf{X}_{[k+1:]}) , \quad (15)$$

$$F_k(\mathbf{X}) = \sum_{\mathbf{i} \in \{0, 1\}^k} \text{eq}(\mathbf{i}; \mathbf{z}_{[:k]}) \cdot \hat{F}_{k,\mathbf{i}}(\mathbf{X}) = \sum_{\mathbf{j} \in \{0, 1\}^{\ell-k}} F_k(\mathbf{j}) \cdot \text{eq}(\mathbf{j}; \mathbf{X}_{[k+1:]}) , \quad (16)$$

where $q_k(\mathbf{j})$ and $F_k(\mathbf{j})$ denote the values of these level- k polynomials at $\mathbf{X}_{[k+1:]} = \mathbf{j}$.

Proof. Both expansions are instances of multilinear interpolation: a polynomial g that is multilinear in a set $\mathbf{X}_S$ of variables, with coefficients in a commutative ring containing $\mathbb{F}$, equals $\sum_{\mathbf{i} \in \{0, 1\}^{|S|}} \text{eq}(\mathbf{i}; \mathbf{X}_S) \cdot g|_{\mathbf{X}_S = \mathbf{i}}$, since both sides are multilinear in $\mathbf{X}_S$ and agree on $\{0, 1\}^{|S|}$. For the prefix expansions, apply this to f with $S = [k-1]$ (resp. $S = [k]$), substitute $\mathbf{z}_{[:k-1]}$ (resp. $\mathbf{z}_{[:k]}$) for $\mathbf{X}_S$, and use Eq. (14); for the suffix expansions, apply it to the level- k polynomials q_k and F_k with $S = [k+1, \ell]$. $\square$

Tables. For $k \in [0, \ell]$, let $T_k := (F_k(\mathbf{j}))_{\mathbf{j} \in \{0, 1\}^{\ell-k}}$ be the table of values of F_k on the hypercube; T_0 is the table of f . The recurrence $T_k(\mathbf{j}) = T_{k-1}(0, \mathbf{j}) + z_k(T_{k-1}(1, \mathbf{j}) - T_{k-1}(0, \mathbf{j}))$ computes all tables from T_0 with fewer than n multiplications. The values in Eqs. (15) and (16) are then $q_k(\mathbf{j}) = T_{k-1}(1, \mathbf{j}) - T_{k-1}(0, \mathbf{j})$ and $F_k(\mathbf{j}) = T_k(\mathbf{j})$. The hypercube values of the $\mathbf{z}$ -independent polynomials are also obtained directly from T_0 : $\hat{q}_{k,\mathbf{i}}(\mathbf{j}) = T_0(\mathbf{i}, 1, \mathbf{j}) - T_0(\mathbf{i}, 0, \mathbf{j})$ and $\hat{F}_{k,\mathbf{i}}(\mathbf{j}) = T_0(\mathbf{i}, \mathbf{j})$.

4.2 Level-structured keys and auxiliary information

By Lemma 3, the elements Q and P of Fig. 1, apart from the masks, are linear combinations of $[q_k(\boldsymbol{\tau})\mathcal{L}_k(\mu)]_1$, $[q_k(\boldsymbol{\tau})U_k(\boldsymbol{\tau}, \mu)/\delta]_1$, $[q_k(\boldsymbol{\tau})\mu^m(\mu^\ell - 1)/\delta]_1$ and $[F_k(\boldsymbol{\tau})M_k(\mu)/\delta]_1$. Each q_k and F_k is a level- k polynomial, so we write the

prover key in the suffix basis $\text{eq}(\mathbf{j}; \mathbf{X}_{[k+1:]})$. For $k \in [\ell]$, $\mathbf{j} \in \{0, 1\}^{\ell-k}$ and $m \in [0, \ell - 2]$, let

$$\begin{aligned} S_{k,\mathbf{j}}^{\text{Q}} &:= [\text{eq}(\mathbf{j}; \boldsymbol{\tau}_{[k+1:]}) \cdot \mathcal{L}_k(\mu)]_1, \quad S_{k,\mathbf{j}}^{\text{U}} := [\text{eq}(\mathbf{j}; \boldsymbol{\tau}_{[k+1:]}) \cdot U_k(\boldsymbol{\tau}, \mu)/\delta]_1, \\ S_{k,\mathbf{j},m}^{\text{Z}} &:= [\text{eq}(\mathbf{j}; \boldsymbol{\tau}_{[k+1:]}) \cdot \mu^m(\mu^\ell - 1)/\delta]_1, \quad S_{k,\mathbf{j}}^{\text{M}} := [\text{eq}(\mathbf{j}; \boldsymbol{\tau}_{[k+1:]}) \cdot M_k(\mu)/\delta]_1, \end{aligned} \quad (17)$$

where $S_{k,\mathbf{j}}^{\text{M}}$ is defined for $k \in [0, \ell]$ (for $k = \ell$, $\mathbf{j}$ is the empty string and $\text{eq}(\mathbf{j}; \cdot) = 1$). By the suffix expansions of Lemma 5, for every multilinear level- k polynomial g ,

$$[g(\boldsymbol{\tau})\mathcal{L}_k(\mu)]_1 = \sum_{\mathbf{j}} g(\mathbf{j}) \cdot S_{k,\mathbf{j}}^{\text{Q}}, \quad [g(\boldsymbol{\tau})U_k(\boldsymbol{\tau}, \mu)/\delta]_1 = \sum_{\mathbf{j}} g(\mathbf{j}) \cdot S_{k,\mathbf{j}}^{\text{U}}, \quad (18)$$

and likewise for S^{Z} and S^{M} . The four families have $n - 1$, $n - 1$, $(n - 1)(\ell - 1)$ and $2n - 1$ elements, respectively. We use the basis $\{[\text{eq}(\mathbf{i}; \boldsymbol{\tau})]_1\}_{\mathbf{i} \in \{0,1\}^\ell}$ for the commitment key. The masked PST commitment $C = \sum_{\mathbf{i}} f(\mathbf{i}) \cdot [\text{eq}(\mathbf{i}; \boldsymbol{\tau})]_1 + \eta'[\delta]_1$ can therefore be computed directly from T_0 . All these elements are linear combinations of the elements of Fig. 1 (for $d = 1$).

Auxiliary information. Fix a *level parameter* $\kappa \in [0, \ell]$. For levels $k \leq \kappa$, the prefix expansions of Lemma 5 express q_k and F_k as linear combinations of at most $2^k \leq 2^\kappa$ polynomials that do not depend on $\mathbf{z}$. Given pk and f , preprocessing computes the corresponding group elements once. This step is separate from computing C , which needs only ck :

$$\begin{aligned} A_{k,\mathbf{i}} &:= [\hat{q}_{k,\mathbf{i}}(\boldsymbol{\tau}) \cdot \mathcal{L}_k(\mu)]_1, \quad B_{k,\mathbf{i}} := [\hat{q}_{k,\mathbf{i}}(\boldsymbol{\tau}) \cdot U_k(\boldsymbol{\tau}, \mu)/\delta]_1, \\ B_{k,\mathbf{i},m} &:= [\hat{q}_{k,\mathbf{i}}(\boldsymbol{\tau}) \cdot \mu^m(\mu^\ell - 1)/\delta]_1, \quad R_{k',\mathbf{i}'} := [\hat{F}_{k',\mathbf{i}'}(\boldsymbol{\tau}) \cdot M_{k'}(\mu)/\delta]_1, \end{aligned} \quad (19)$$

for $k \in [\kappa]$, $\mathbf{i} \in \{0, 1\}^{k-1}$, $m \in [0, \ell - 2]$, $k' \in [0, \kappa]$ and $\mathbf{i}' \in \{0, 1\}^{k'}$. By Eq. (18), each auxiliary element at level k is computed with one $\text{MSM}(2^{\ell-k})$, using entries of T_0 or their differences as scalars. For levels $k > \kappa$, each weighted suffix encoding contributes $2^{\ell-k}$ terms to the final MSMs, with a separate contribution for each correction index m . Choosing $\kappa = \ell/2$ gives at most $\sqrt{n}$ terms in each prefix or suffix sum. We take ℓ even for simplicity; for odd ℓ , $\kappa = \lfloor \ell/2 \rfloor$ gives the same bounds up to a factor $\sqrt{2}$.

Fig. 2 gives the algorithms. The public parameter κ is fixed for preprocessing and subsequent openings. To fit the interface of Definition 1, Eval may run preprocessing itself when it receives only D . The opening costs below assume preprocessing is complete.

Lemma 6. *For $d = 1$ and every $\kappa \in [0, \ell]$, the algorithms of Fig. 2 output the same commitment $C = [f(\boldsymbol{\tau}) + \eta'\delta]_1$ and the same proof $(Q, P) = ([q(\boldsymbol{\tau}, \mu) + \eta\delta]_1, [\Phi(\boldsymbol{\tau}, \mu)/\delta + \eta h(\boldsymbol{\tau}, \mu) - \eta'/\ell]_1)$ as Fig. 1 (for the same randomness η, η'). In particular, Theorem 1 applies to Fig. 2 with $d = 1$, i.e., under the $(3\ell - 1, \ell + 1)$ -dlog assumption.*

The new keys preserve the SRS types and degree bounds used for extraction (Remark 2); the full proof is in Section B.3.

$\text{Setup}(1^\lambda, \ell, 1) \rightarrow (\text{ck}, \text{pk}, \text{vk})$: pick the trapdoors as in Fig. 1 and output

$$\text{ck} = \left(([\text{eq}(\mathbf{i}; \boldsymbol{\tau})]_1)_{\mathbf{i} \in \{0,1\}^\ell}, [\delta]_1 \right),$$

$$\text{pk} = \left([\delta]_1, ([\mathcal{L}_k(\mu)]_1, [\tau_k \mathcal{L}_k(\mu)]_1)_k, (S_{k,j}^Q)_{k,j}, (S_{k,j}^U)_{k,j}, (S_{k,j,m}^Z)_{k,j,m}, (S_{k',j'}^M)_{k',j'} \right),$$

with the index ranges of Eq. (17), and vk as in Fig. 1.

$\text{Commit}(\text{ck}, f) \rightarrow (C, D)$: on input $T_0 = (f(\mathbf{i}))_{\mathbf{i} \in \{0,1\}^\ell}$, pick $\eta' \xleftarrow{\$} \mathbb{F}$ and set $D := \frac{\eta'}{\ell} [1]_1$; output

$$C = \sum_{\mathbf{i} \in \{0,1\}^\ell} T_0(\mathbf{i}) \cdot [\text{eq}(\mathbf{i}; \boldsymbol{\tau})]_1 + \eta' \cdot [\delta]_1$$

together with D (take $D = 0$ without hiding).

$\text{Preprocess}(\text{pk}, f, \kappa; D) \rightarrow \text{aux}$:

- for $k \in [\kappa]$ and $\mathbf{i} \in \{0,1\}^{k-1}$, with $c_j := T_0(\mathbf{i}, 1, j) - T_0(\mathbf{i}, 0, j)$ for $j \in \{0,1\}^{\ell-k}$, compute

$$A_{k,\mathbf{i}} = \sum_j c_j \cdot S_{k,j}^Q, \quad B_{k,\mathbf{i}} = \sum_j c_j \cdot S_{k,j}^U, \quad B_{k,\mathbf{i},m} = \sum_j c_j \cdot S_{k,j,m}^Z,$$

where $m \in [0, \ell-2]$ indexes only the last family;

- for $k \in [0, \kappa]$ and $\mathbf{i} \in \{0,1\}^k$, compute $R_{k,\mathbf{i}} = \sum_{j \in \{0,1\}^{\ell-k}} T_0(\mathbf{i}, j) \cdot S_{k,j}^M$;
- output $\text{aux} = (D, (A_{k,\mathbf{i}}, B_{k,\mathbf{i}})_{k,\mathbf{i}}, (B_{k,\mathbf{i},m})_{k,\mathbf{i},m}, (R_{k,\mathbf{i}})_{k,\mathbf{i}})$.

$\text{Eval}(\text{pk}, f, \mathbf{z}, v; \text{aux}) \rightarrow \pi$:

- Compute the tables $T_1, \dots, T_\ell$, the prefix weights $e_{k,\mathbf{i}} := \text{eq}(\mathbf{i}; \mathbf{z}_{[k]})$ for $k \in [0, \kappa]$ and $\mathbf{i} \in \{0,1\}^k$, and the coefficients $w_{k,m}$ of Eq. (12). Parse D from aux and pick $\eta \xleftarrow{\$} \mathbb{F}$. For $k > \kappa$ write $q_k(j) = T_{k-1}(1, j) - T_{k-1}(0, j)$.
- Compute and output $\pi = (Q, P)$ where

$$Q = \sum_{k \in [\kappa]} \sum_{\mathbf{i} \in \{0,1\}^{k-1}} e_{k-1,\mathbf{i}} \cdot A_{k,\mathbf{i}} + \sum_{k \in [\kappa+1, \ell]} \sum_{j \in \{0,1\}^{\ell-k}} q_k(j) \cdot S_{k,j}^Q + \eta \cdot [\delta]_1,$$

$$P = \sum_{k \in [\kappa]} \sum_{\mathbf{i} \in \{0,1\}^{k-1}} e_{k-1,\mathbf{i}} \cdot \left(B_{k,\mathbf{i}} - \sum_{m=0}^{\ell-2} w_{k,m} \cdot B_{k,\mathbf{i},m} \right)$$

$$+ \sum_{k \in [\kappa+1, \ell]} \sum_{j \in \{0,1\}^{\ell-k}} q_k(j) \cdot \left(S_{k,j}^U - \sum_{m=0}^{\ell-2} w_{k,m} \cdot S_{k,j,m}^Z \right)$$

$$+ \sum_{k \in [0, \kappa]} \sum_{\mathbf{i} \in \{0,1\}^k} e_{k,\mathbf{i}} \cdot R_{k,\mathbf{i}} + \sum_{k \in [\kappa+1, \ell]} \sum_{j \in \{0,1\}^{\ell-k}} T_k(j) \cdot S_{k,j}^M$$

$$+ \eta \cdot \sum_{k \in [\ell]} ([\tau_k \mathcal{L}_k(\mu)]_1 - z_k \cdot [\mathcal{L}_k(\mu)]_1) - D.$$

$\text{Verify}(\text{vk}, C, \mathbf{z}, v, \pi)$: as in Fig. 1.

Fig. 2. The multilinear prover for Couplet, with auxiliary information for the levels $k \leq \kappa$.

4.3 Costs

Theorem 2. *Let $\ell \geq 1$ with $\ell \mid p-1$, $n = 2^\ell$, and $\kappa \in [0, \ell]$. The multilinear instantiation of Couplet in Fig. 2 has the following costs, including the hiding terms.*

1. Setup outputs $n+1$ elements of $\mathbb{G}_1$ as ck , $(\ell+3)n + \ell - 1$ elements of $\mathbb{G}_1$ as pk , and $\ell+2$ elements of $\mathbb{G}_2$ as vk .
2. Computing the commitment C and the hint D requires one $\text{MSM}(n)$ and two scalar multiplications in $\mathbb{G}_1$. Preprocessing produces aux with $(\ell+3) \cdot 2^\kappa - \ell - 1$ elements of $\mathbb{G}_1$, including D . Its total MSM size is $\frac{\kappa(\ell+3)}{2} \cdot n + n$, with each MSM having size at least $2^{\ell-\kappa}$.
3. Given aux , each opening uses at most $n + (\ell+1)(2^\kappa + 2^{\ell-\kappa}) + O(\ell^2)$ field multiplications and two MSMs in $\mathbb{G}_1$ of total size at most $(\ell+3) \cdot 2^\kappa + (\ell+2) \cdot 2^{\ell-\kappa}$.
4. The proof consists of two $\mathbb{G}_1$ elements. Verification uses ℓ scalar multiplications in $\mathbb{G}_2$, one in $\mathbb{G}_1$, and three pairings.

For even ℓ , choosing $\kappa = \ell/2$ gives fewer than $(\ell+3)\sqrt{n}$ auxiliary elements. Preprocessing then has total MSM size $\frac{\ell(\ell+3)}{4} \cdot n + n = O(n \log^2 n)$, with each MSM having size at least $\sqrt{n}$. Each opening uses at most $n + 2(\ell+1)\sqrt{n} + O(\ell^2)$ field multiplications and two MSMs in $\mathbb{G}_1$ of total size at most $(2\ell+5)\sqrt{n}$.

The proof is in Section B.3.

Remark 4 (Choosing the level parameter). At $\kappa = 0$, aux contains only D and $R_{0,()}\leftarrow [f(\tau)M_0(\mu)/\delta]_1$. Computing $R_{0,()}$ takes one $\text{MSM}(n)$, after which each opening uses at most $(\ell+2)n + \ell + 1$ MSM terms. To avoid this preprocessing, compute $R_{0,()}$ during each opening instead, adding n MSM terms per opening.

For $\kappa = \ell/2$, the preprocessing MSMs have size at least $\sqrt{n}$, so Pippenger's algorithm evaluates them using $O(n \log n \cdot \log p)$ group additions.

For example, at $n = 2^{16}$, balanced preprocessing uses 5,046,272 MSM terms and each opening uses at most 9,472. Using $2n$ MSM terms per opening as a leading-term baseline for Mercury [17], these bounds offset the preprocessing cost after 42 openings. This comparison excludes the common commitment cost and measures total MSM size; actual running times also depend on the MSM lengths and implementation.

Remark 5 (Univariate polynomial commitments). Let $g(X) = \sum_{i=0}^{n-1} c_i X^i$ with $n = 2^\ell$. Identify each index i with its binary vector $\mathbf{i} = (i_1, \dots, i_\ell)$, where $i = \sum_{j=1}^\ell i_j 2^{\ell-j}$, and define $f(\mathbf{X}) := \sum_{i=0}^{n-1} c_i \cdot \mathbf{X}^{\mathbf{i}}$. Then

$$g(x) = f(x^{2^{\ell-1}}, \dots, x^2, x), \quad x \in \mathbb{F}.$$

To use Fig. 2, first convert the coefficients of g into the hypercube table of f . Its entries are the subset sums $f(\mathbf{b}) = \sum_{\mathbf{i}: \mathbf{i} \leq \mathbf{b}} c_{\mathbf{i}}$, where $\mathbf{i} \leq \mathbf{b}$ is interpreted coordinatewise. All entries can be computed with $n\ell/2$ field additions.

Commit to f and open it at $(x^{2^{\ell-1}}, \dots, x)$ to certify $g(x)$. With balanced preprocessing, this gives two-element proofs, verification with three pairings, and $O(\sqrt{n} \log n)$ opening MSM terms. Compared with KZG's linear opening MSM size, the smaller opening cost requires $O(n \log n)$ key elements and $O(n \log^2 n)$ preprocessing MSM terms.

5 Constant-size batch openings

We extend Couplet to prove $f(z_i) = v_i$ for all $i \in [s]$ with two $\mathbb{G}_1$ elements, four pairings and no verifier challenge. Each claim is assigned a coset, and a commitment tag ensures that all claims refer to the same polynomial. The batch capacity s is fixed at setup. A batch is an ordered tuple; to fill a nonempty batch of fewer than s claims, repeat one claim in the remaining slots. The formal interface and security definitions are in Definition 3.

The batch scheme uses a different commitment key from Section 3. To support both single and batch openings of the same commitment, use the batch scheme from setup onward.

5.1 Univariate sumcheck over cosets

Interpolation domain. Let $s \geq 1$ with $s\ell \mid p-1$, let σ be a primitive $s\ell$ -th root of unity, and set $\nu := \sigma^s$ and $\theta := \sigma^\ell$. Thus ν has order ℓ and θ has order s . The $s\ell$ -th roots of unity form s disjoint cosets

$$\mathcal{H}_i := \{\rho_{i,k} : k \in [\ell]\} = \sigma^{i-1} \langle \nu \rangle, \quad \rho_{i,k} := \sigma^{i-1} \nu^k, \quad i \in [s].$$

Every point $\rho \in \mathcal{H}_i$ satisfies $\rho^\ell = \theta^{i-1}$. The map $\rho \mapsto \rho^\ell$ therefore sends each coset to a distinct one of the s values $1, \theta, \dots, \theta^{s-1}$.

Lagrange polynomials. Introduce a formal variable W for interpolation on the s coset labels θ^{i-1} . Their Lagrange basis is

$$\mathcal{L}_i^*(W) := \frac{\theta^{i-1}(W^s - 1)}{s(W - \theta^{i-1})}, \quad i \in [s].$$

Each $\mathcal{L}_i^*$ has degree $s-1$ and equals 1 at θ^{i-1} and 0 at the other labels. Substituting $W = Y^\ell$ makes $\mathcal{L}_i^*(Y^\ell)$ equal 1 on $\mathcal{H}_i$ and 0 on every other coset, with $\sum_i \mathcal{L}_i^*(Y^\ell) = 1$.

Within coset $\mathcal{H}_i$, the degree- $(\ell-1)$ Lagrange polynomial for $\rho_{i,k}$ is

$$\mathcal{L}_k^{(i)}(Y) := \frac{\rho_{i,k}(Y^\ell - \theta^{i-1})}{\ell \theta^{i-1}(Y - \rho_{i,k})}, \quad k \in [\ell].$$

It equals 1 at $\rho_{i,k}$ and 0 at the other points of $\mathcal{H}_i$. Multiplying by the coset selector gives the Lagrange basis on the full domain:

$$\mathcal{L}_{i,k}(Y) := \mathcal{L}_i^*(Y^\ell) \cdot \mathcal{L}_k^{(i)}(Y).$$

This product has degree $s\ell - 1$, equals 1 at $\rho_{i,k}$ and vanishes at every other domain point.

Sumcheck on each coset. Let $\mathbb{R} = \mathbb{F}[\mathbf{X}]$. Since $\mathcal{L}_k^{(i)}(0) = 1/\ell$, evaluating the interpolation of any $g \in \mathbb{R}[Y]$ of degree at most $\ell - 1$ at 0 gives

$$\sum_{k \in [\ell]} g(\rho_{i,k}) = \ell \cdot g(0), \quad i \in [s]. \quad (20)$$

For $\Phi \in \mathbb{R}_{\leq 2s\ell-2}[Y]$, let c be the remainder upon division by $Y^{s\ell} - 1$. For each i , let c_i be the polynomial of degree at most $\ell - 1$ that agrees with c on $\mathcal{H}_i$. Then $c = \sum_i \mathcal{L}_i^*(Y^\ell) c_i(Y)$, and the sum of Φ on coset i is $\ell c_i(0)$. All coset sums vanish precisely when each c_i is divisible by Y . This gives the subspace

$$\mathcal{V}_s := \left\{ \begin{array}{l} p(\mathbf{X}, Y) \cdot (Y^{s\ell} - 1) \\ + \sum_{i \in [s]} \mathcal{L}_i^*(Y^\ell) \cdot r_i(\mathbf{X}, Y) \cdot Y \end{array} : \begin{array}{l} p \in \mathbb{R}_{\leq s\ell-2}[Y], \\ r_i \in \mathbb{R}_{\leq \ell-2}[Y] \end{array} \right\} \subseteq \mathbb{R}_{\leq 2s\ell-2}[Y], \quad (21)$$

whose elements have unique representations of this form (Lemma 9). For $s = 1$ and $\sigma = \omega$, we recover $\mathcal{V}_s = \mathcal{V}$ and the sumcheck of Section 2.

5.2 The scheme

Let $f \in \mathbb{F}_{\leq d}[\mathbf{X}]$ and let $(z_i, v_i)_{i \in [s]}$ be the batch. For each i , let $q_{i,k}$ be the PST quotients for f at z_i (Lemma 2). We interpolate the $s\ell$ quotients and linear factors over the full domain:

$$\begin{aligned} q(\mathbf{X}, Y) &:= \sum_{i,k} \mathcal{L}_{i,k}(Y) \cdot q_{i,k}(\mathbf{X}), & u(\mathbf{X}, Y) &:= \sum_{i,k} \mathcal{L}_{i,k}(Y) \cdot X_k, \\ \bar{u}(Y) &:= \sum_{i,k} \mathcal{L}_{i,k}(Y) \cdot z_{i,k}, & h &:= u - \bar{u}, \end{aligned} \quad (22)$$

so that $q(\mathbf{X}, \rho_{i,k}) = q_{i,k}(\mathbf{X})$ and $h(\mathbf{X}, \rho_{i,k}) = X_k - z_{i,k}$. The s evaluation claims correspond to the s coset sums $\sum_{k \in [\ell]} q(\mathbf{X}, \rho_{i,k}) \cdot h(\mathbf{X}, \rho_{i,k}) = f(\mathbf{X}) - v_i$, and by Lemma 9 (with $t_i = f - v_i$ and $\sum_i \mathcal{L}_i^*(Y^\ell) = 1$) they hold if and only if

$$\Phi := q \cdot h - \frac{1}{\ell} \left(f - \sum_{i \in [s]} v_i \cdot \mathcal{L}_i^*(Y^\ell) \right) \in \mathcal{V}_s. \quad (23)$$

Closed forms. For each claim $f(z_i) = v_i$, let $F_k^{(i)} := f(z_{i,[k]}, \mathbf{X}_{[k+1:]})$ for $k \in [0, \ell]$ and $q_{i,k} := (F_{k-1}^{(i)} - F_k^{(i)}) / (X_k - z_{i,k})$ for $k \in [\ell]$. Let p, r_i be the polynomials in Lemma 9(2) for $a = q$, $b = h$ and $t_i = f - v_i$. The closed forms of Lemma 3 extend by applying the same telescoping argument within each coset. Define

$$U_{i,k}(\mathbf{X}, Y) := \mathcal{L}_{i,k}(Y) \cdot (u(\mathbf{X}, Y) - X_k), \quad Z_{i,k}(Y) := \mathcal{L}_{i,k}(Y) \cdot (\bar{u}(Y) - z_{i,k}), \quad (24)$$

and $M_0^{(i)} := \mathcal{L}_1^{(i)} - \frac{1}{\ell}$, $M_k^{(i)} := \mathcal{L}_{k+1}^{(i)} - \mathcal{L}_k^{(i)}$ for $k \in [\ell - 1]$, $M_\ell^{(i)} := \frac{1}{\ell} - \mathcal{L}_\ell^{(i)}$. The proof of Lemma 3 shows that $U_{i,k}$ and $Z_{i,k}$ are divisible by $Y^{s\ell} - 1$ with quotients of degree at most $s\ell - 2$. Each $M_k^{(i)}$ is divisible by Y , and

$$\Phi = \sum_{i,k} q_{i,k}(\mathbf{X}) \cdot (U_{i,k}(\mathbf{X}, Y) - Z_{i,k}(Y)) + \sum_{i \in [s]} \mathcal{L}_i^*(Y^\ell) \sum_{k=0}^{\ell} F_k^{(i)}(\mathbf{X}) \cdot M_k^{(i)}(Y), \quad (25)$$

where the first sum equals $p \cdot (Y^{s\ell} - 1)$ and the i -th inner sum equals $r_i \cdot Y$. As before, we write $Z_{i,k}(Y) = (Y^{s\ell} - 1) \sum_{m=0}^{s\ell-2} w_{k,m}^{(i)} Y^m$. The coefficients are computed from the opening points with $O(s^2 \ell^2)$ field operations.

Tagging the commitment. The batch prover key contains Y -dependent encodings. Without a separate commitment tag, a prover could use them to commit to $c(\mathbf{X}, Y) = \sum_{i \in [s]} f_i(\mathbf{X}) \mathcal{L}_i^*(Y^\ell)$ and satisfy each coset identity using the PST quotients of a different polynomial f_i . We prevent this by sampling an additional nonzero trapdoor γ and setting $C = [(f(\boldsymbol{\tau}) + \eta' \delta) / \gamma]_1$. Only the commitment-key elements $[\boldsymbol{\tau}^i / \gamma]_1$ and the masking element $[\delta / \gamma]_1$ carry $1/\gamma$. Pairing C with $[\gamma / \ell]_2$ lets the extraction argument isolate these terms and recover a single polynomial independent of Y . The masking hint remains $D = \frac{\eta'}{\ell} \cdot [1]_1$.

The verifier encodes the claimed values as $V_v = \sum_{i \in [s]} v_i \cdot [\mathcal{L}_i^*(\mu^\ell) / \ell]_2$. Publishing $[\mathcal{L}_i^*(\mu^\ell) / (\ell \gamma)]_1$ would again permit Y -dependent tagged commitments. The verifier therefore pairs V_v with $[1]_1$ separately, giving four pairings overall. With fresh $\eta \xleftarrow{\$} \mathbb{F}$, the prover sends

$$Q = [q(\boldsymbol{\tau}, \mu) + \eta \delta]_1, \quad P = [\Phi(\boldsymbol{\tau}, \mu) / \delta + \eta h(\boldsymbol{\tau}, \mu) - \eta' / \ell]_1,$$

and the verifier checks

$$e(Q, [h(\boldsymbol{\tau}, \mu)]_2) = e(P, [\delta]_2) + e(C, [\gamma / \ell]_2) - e([1]_1, V_v).$$

Take $\eta = \eta' = 0$ without hiding. The full algorithms and keys are in Fig. 4.

5.3 Security and costs

Theorem 3. *Let $\ell, d, s \geq 1$ with $s\ell \mid p - 1$. The batch variant of Couplet in Fig. 4 is a complete and perfectly zero-knowledge polynomial commitment with batch openings for $\mathbb{F}_{\leq d}[\mathbf{X}]$ (Definition 3).*

It is extractable in the AGM under the $(d\ell + 2s\ell, s\ell + 2)$ -dlog assumption. For every algebraic PPT adversary $\mathcal{A}$, there are an efficient extractor $\mathcal{E}$ and an expected-polynomial-time discrete-log adversary $\mathcal{B}$ such that the extractability error is at most

$$\text{Adv}_{\mathcal{B}}^{(d\ell + 2s\ell, s\ell + 2)\text{-dlog}} + \frac{d\ell + 3s\ell + 4}{p}.$$

The extractor recovers f and a scalar η' from the representation of C alone, before receiving an opening. Except with the same error bound, every accepted opening also satisfies $C = [(f(\boldsymbol{\tau}) + \eta' \delta) / \gamma]_1$.

Proofs consist of two $\mathbb{G}_1$ elements. Verification takes $s\ell + s$ scalar multiplications in $\mathbb{G}_2$ and four pairings.

The proof is in Section B.5.

Prover costs. For $N = (d + 1)^\ell$, the prover key has $O(Ns\ell)$ group elements, and an opening uses $O(Ns\ell)$ MSM terms and $O(s^2N\ell)$ field operations. For multilinear polynomials with $n = 2^\ell$, preprocessing reduces each opening to $O(s\sqrt{n}\log n)$ MSM terms and $O(sn + s^2\ell\sqrt{n})$ field operations. Preprocessing uses $O(sn\log^2 n)$ MSM terms and stores $O(s\sqrt{n}\log n)$ auxiliary group elements. Detailed key and MSM counts are in Section B.5 and Remark 6.

References

1. Arnon, G., Chiesa, A., Fenzi, G., Yorgev, E.: WHIR: Reed–Solomon proximity testing with super-fast verification. In: Fehr, S., Fouque, P.A. (eds.) *Advances in Cryptology – EUROCRYPT 2025*. Lecture Notes in Computer Science, vol. 15604, pp. 214–243. Springer (2025). https://doi.org/10.1007/978-3-031-91134-7_8, <https://eprint.iacr.org/2024/1586>
2. Arun, A., Ganesh, C., Lokam, S., Mopuri, T., Sridhar, S.: Dew: Transparent constant-sized zkSNARKs. *Cryptology ePrint Archive*, Paper 2022/419 (2022), <https://eprint.iacr.org/2022/419>
3. Arun, A., Setty, S., Thaler, J.: Jolt: SNARKs for virtual machines via lookups. In: *Advances in Cryptology – EUROCRYPT 2024*. LNCS, vol. 14656, pp. 3–33. Springer (2024). https://doi.org/10.1007/978-3-031-58751-1_1
4. Aztec Labs: Barretenberg: Aztec’s cryptographic proving backend (UltraHonk). <https://github.com/AztecProtocol/barretenberg> (2026), accessed September 2026
5. Bauer, B., Fuchsbauer, G., Loss, J.: A classification of computational assumptions in the algebraic group model. In: *Advances in Cryptology - CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part II*. pp. 121–151. Springer-Verlag, Berlin, Heidelberg (2020). https://doi.org/10.1007/978-3-030-56880-1_5
6. Belohorec, J., Hubáček, P., Kalsta, A., Mašková, K.: Chopin: Optimal pairing-based multilinear polynomial commitments from bivariate KZG. In: Heninger, N., Rosulek, M. (eds.) *Advances in Cryptology – CRYPTO 2026*. Lecture Notes in Computer Science, vol. 16808, pp. 428–459. Springer (2026). https://doi.org/10.1007/978-3-032-35424-2_15, <https://eprint.iacr.org/2026/480>
7. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) *Advances in Cryptology – EUROCRYPT 2019*. pp. 103–128. Springer International Publishing, Cham (2019)
8. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.S. (eds.) *Advances in Cryptology – EUROCRYPT 2016*. pp. 327–357. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
9. Bootle, J., Chiesa, A., Hu, Y., Orrù, M.: Gemini: Elastic SNARKs for diverse environments. In: Dunkelman, O., Dziembowski, S. (eds.) *Advances in Cryptology – EUROCRYPT 2022*. pp. 427–457. Springer International Publishing, Cham (2022)

10. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: *Advances in Cryptology – EUROCRYPT 2020* (2020)
11. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: *2018 IEEE Symposium on Security and Privacy (SP)*. pp. 315–334 (2018). <https://doi.org/10.1109/SP.2018.00020>
12. Campanelli, M., Nitulescu, A., Ràfols, C., Zacharakis, A., Zapico, A.: Linear-map vector commitments and their practical applications. *Cryptology ePrint Archive*, Paper 2022/705 (2022), <https://eprint.iacr.org/2022/705>
13. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In: Hazay, C., Stam, M. (eds.) *Advances in Cryptology – EUROCRYPT 2023*. pp. 499–530. Springer Nature Switzerland, Cham (2023)
14. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: Pre-processing zkSNARKs with universal and updatable SRS. In: *Advances in Cryptology - EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I. pp. 738–768. Springer-Verlag, Berlin, Heidelberg (2020). https://doi.org/10.1007/978-3-030-45721-1_26
15. Daza, V., Ràfols, C., Zacharakis, A.: Updateable inner product argument with logarithmic verifier and applications. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) *Public-Key Cryptography – PKC 2020*. pp. 527–557. Springer International Publishing, Cham (2020)
16. Diamond, B.E., Posen, J.: Succinct arguments over towers of binary fields. *Cryptology ePrint Archive*, Paper 2023/1784 (2023), <https://eprint.iacr.org/2023/1784>
17. Eagen, L., Gabizon, A.: MERCURY: A multilinear polynomial commitment scheme with constant proof size and no prover FFTs. *Cryptology ePrint Archive*, Paper 2025/385 (2025), <https://eprint.iacr.org/2025/385>
18. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) *Advances in Cryptology – CRYPTO 2018*. pp. 33–62. Springer International Publishing, Cham (2018)
19. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over Lattice bases for oecumenical noninteractive arguments of knowledge. *Cryptology ePrint Archive*, Paper 2019/953 (2019), <https://eprint.iacr.org/2019/953>
20. Ganesh, C., Patranabis, S., Singh, N.: Samaritan: Linear-time prover SNARK from new multilinear polynomial commitments. In: Hanaoka, G., Yang, B.Y. (eds.) *Advances in Cryptology – ASIACRYPT 2025*. Lecture Notes in Computer Science, vol. 16249, pp. 456–489. Springer (2026). https://doi.org/10.1007/978-981-95-5116-3_15, <https://eprint.iacr.org/2025/419>
21. Golovnev, A., Lee, J., Setty, S., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In: *Advances in Cryptology – CRYPTO 2023* (2023)
22. Groth, J.: On the size of pairing-based non-interactive arguments. In: *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Vienna, Austria, May 8-12, 2016, Proceedings, Part II. pp. 305–326 (2016)
23. Hafezi, H., Shirzad, A., Bünz, B., Bonneau, J.: Transparent dictionaries from polynomial commitments. In: *35th USENIX Security Symposium (USENIX Security 26)*. pp. 4307–4324. USENIX Association (2026), <https://www.usenix.org/conference/usenixsecurity26/presentation/hafezi>

24. Hemo, T., Jue, K., Rabinovich, E., Roh, G., Rothblum, R.D.: Jagged polynomial commitments (or: How to stack multilinear). In: *Advances in Cryptology – EUROCRYPT 2026*. LNCS, vol. 16547, pp. 121–147. Springer (2026). https://doi.org/10.1007/978-3-032-25336-1_5, cryptology ePrint Archive, Paper 2025/917
25. Kadianakis, G., Zapico, A., Hafezi, H., Bünz, B.: KZH-fold: Accountable voting from sublinear accumulation. *Cryptology ePrint Archive*, Paper 2025/144 (2025), <https://eprint.iacr.org/2025/144>
26. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-Size Commitments to Polynomials and Their Applications. In: *ASIACRYPT’10* (2010)
27. Kohrita, T., Towa, P.: Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments. *Journal of Cryptology* **37**(4), 38 (Oct 2024). <https://doi.org/10.1007/s00145-024-09519-0>
28. Lee, J.: Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In: *Theory of Cryptography – TCC 2021* (2021), *cryptology ePrint Archive*, Paper 2020/1274
29. Lipmaa, H.: Orca and Dolphin: Efficient bivariate and multilinear polynomial commitment schemes under standard assumptions. *Cryptology ePrint Archive*, Paper 2026/539 (2026), <https://eprint.iacr.org/2026/539>
30. Liu, J., Zhang, L.F.: Matproofs: Maintainable matrix commitment with efficient aggregation. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. pp. 2041–2054. CCS ’22, Association for Computing Machinery, New York, NY, USA (2022). <https://doi.org/10.1145/3548606.3560591>
31. Lund, C., Fortnow, L., Karloff, H., Nisan, N.: Algebraic methods for interactive proof systems. *J. ACM* **39**(4), 859–868 (oct 1992). <https://doi.org/10.1145/146585.146605>
32. Luo, Z., Jia, Y., Gracia, A.V.O., Kate, A.: Cauchyproofs: Batch-updatable vector commitment with easy aggregation and application to stateless blockchains. *Cryptology ePrint Archive*, Paper 2025/038 (2025). <https://doi.org/10.1109/SP61157.2025.00247>, <https://eprint.iacr.org/2025/038>
33. Nguyen, N.K., O’Rourke, G., Zhang, J.: Hachi: Efficient lattice-based multilinear polynomial commitments over extension fields. *Cryptology ePrint Archive*, Paper 2026/156 (2026), <https://eprint.iacr.org/2026/156>
34. Nguyen, N.K., Seiler, G.: Greyhound: Fast polynomial commitments from lattices. In: *Advances in Cryptology – CRYPTO 2024* (2024)
35. Papamanthou, C., Shi, E., Tamassia, R.: Signatures of Correct Computation. In: *TCC’13* (2013)
36. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: *2013 IEEE Symposium on Security and Privacy, SP 2013*, Berkeley, CA, USA, May 19–22, 2013. pp. 238–252 (2013)
37. Schwartz, J.T.: Probabilistic algorithms for verification of polynomial identities. In: *International Symposium on Symbolic and Algebraic Manipulation*. pp. 200–215. Springer (1979)
38. Setty, S.: Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In: *Advances in Cryptology – CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020*, Santa Barbara, CA, USA, August 17–21, 2020, *Proceedings, Part III*. pp. 704–737. Springer-Verlag, Berlin, Heidelberg (2020). https://doi.org/10.1007/978-3-030-56877-1_25

39. Setty, S.: HyperKZG: A polynomial commitment scheme for multilinear polynomials derived from Gemini. Implementation in the Nova library (2023), <https://github.com/microsoft/Nova>
40. Srinivasan, S., Chepurnoy, A., Papamanthou, C., Tomescu, A., Zhang, Y.: Hyperproofs: Aggregating and maintaining proofs in vector commitments. In: 31st USENIX Security Symposium (USENIX Security 22). USENIX Association, Boston, MA (Aug 2022), <https://www.usenix.org/conference/usenixsecurity22/presentation/srinivasan>
41. Sun, H., Li, J., Zhang, H.: zkLLM: Zero knowledge proofs for large language models. In: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security. pp. 4405–4419. CCS '24, ACM (2024). <https://doi.org/10.1145/3658644.3670334>
42. Wahby, R.S., Tzialla, I., abhi shelat, Thaler, J., Walfish, M.: Doubly-efficient zk-SNARKs without trusted setup. In: IEEE Symposium on Security and Privacy (SP) 2018 (2018)
43. Wang, W., Papamanthou, C., Srinivasan, S., Papadopoulos, D.: Dynamic zk-SNARKs. Cryptology ePrint Archive, Paper 2024/1566 (2024), <https://eprint.iacr.org/2024/1566>
44. Wang, W., Ulichney, A., Papamanthou, C.: BalanceProofs: Maintainable vector commitments with fast aggregation. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 4409–4426. USENIX Association, Anaheim, CA (Aug 2023), <https://www.usenix.org/conference/usenixsecurity23/presentation/wang-weijie>
45. Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: Succinct zero-knowledge proofs with optimal prover computation. In: Advances in Cryptology – CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III. pp. 733–764. Springer-Verlag, Berlin, Heidelberg (2019). https://doi.org/10.1007/978-3-030-26954-8_24
46. Yong, X., Wu, J., Wang, J.: Xproofs: New aggregatable and maintainable matrix commitment with optimal proof size. Cryptology ePrint Archive, Paper 2024/663 (2024), <https://eprint.iacr.org/2024/663>
47. Zeilberger, H., Chen, B., Fisch, B.: BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In: Advances in Cryptology – CRYPTO 2024 (2024)
48. Zhang, Y., Genkin, D., Katz, J., Papadopoulos, D., Papamanthou, C.: vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 863–880 (2017)
49. Zhang, Y.: Vela and Carina: Fast pairing-based multilinear polynomial commitments from reciprocal polynomials. Cryptology ePrint Archive, Paper 2026/1438 (2026), <https://eprint.iacr.org/2026/1438>
50. Zippel, R.: Probabilistic algorithms for sparse polynomials. In: International Symposium on Symbolic and Algebraic Manipulation. pp. 216–226. Springer (1979)

A Algebraic reduction tools

We reduce polynomial relations among SRS elements to discrete logarithms, following the affine-embedding approach of [8,15,18,19,5,43]. All polynomial families below are public, independent of the sampled trapdoors, and explicitly represented with polynomially bounded size and degree. For a PPT adversary $\mathcal{A}$,

the discrete-log reductions run in expected polynomial time because of root finding. A bounded-time root-finding implementation with restarts yields PPT reductions, with negligible additional failure probability added to the displayed bounds.

A.1 Polynomial relations and discrete logarithms

Lemma 7 (Affine Schwartz–Zippel). *Let $f \in \mathbb{F}[X_1, \dots, X_\ell]$ be nonzero of total degree at most D , and fix $\mathbf{s} \in \mathbb{F}^\ell$. For uniform $\mathbf{r} \in \mathbb{F}^\ell$, the polynomial $g(Y) := f(\mathbf{r}Y + \mathbf{s})$ is identically zero with probability at most D/p .*

Proof. Let D' be the total degree of f , and let $f_{D'}$ be its nonzero homogeneous part of degree D' . The coefficient of $Y^{D'}$ in g is $f_{D'}(\mathbf{r})$. Thus Schwartz–Zippel [37, 50] gives $\Pr[g \equiv 0] \leq \Pr[f_{D'}(\mathbf{r}) = 0] \leq D'/p \leq D/p$. This also covers nonzero constant f , for which the probability is zero. $\square$

The following game formalizes a nontrivial relation among evaluated polynomials, either in a source group or after pairing.

Assumption A.1 (($\mathbf{f}, \mathbf{h}$)-find-rep) *For every PPT adversary $\mathcal{A}$, the probability of winning the game in Fig. 3 is negligible. We denote this probability by $\text{Adv}_{\mathcal{A}}^{\text{find-rep}}$, leaving the polynomial families implicit when they are clear from context.*

($\mathbf{f}, \mathbf{h}$)-find-rep $_{\mathcal{A}, \mathcal{G}}$:

- 1: $(G_1, G_2) \leftarrow \text{GroupGen}(\mathcal{G})$.
- 2: $\tau \xleftarrow{\$} \mathbb{F}^\ell$.
- 3: $\mathbf{U} \leftarrow ([f_1(\tau)]_1, \dots, [f_u(\tau)]_1)$.
- 4: $\mathbf{W} \leftarrow ([h_1(\tau)]_2, \dots, [h_v(\tau)]_2)$.
- 5: $(\mu, \nu, \zeta) \leftarrow \mathcal{A}(\mathbf{U}, \mathbf{W})$.
- 6: Return 1 if and only if at least one condition holds:
 - (a) $\mu \in \mathbb{F}^u$, $\mu^\top \mathbf{f} \neq 0$, and $\mu^\top \mathbf{U} = 0$.
 - (b) $\nu \in \mathbb{F}^v$, $\nu^\top \mathbf{h} \neq 0$, and $\nu^\top \mathbf{W} = 0$.
 - (c) $\zeta \in \mathbb{F}^{u \times v}$, $\mathbf{f}^\top \zeta \mathbf{h} \neq 0$, and $\sum_{i,j} \zeta_{i,j} e(U_i, W_j) = 0$.

Fig. 3. The **find-rep** game for public polynomial families $\mathbf{f} = (f_1, \dots, f_u)$ and $\mathbf{h} = (h_1, \dots, h_v)$. Include the constant polynomial 1 in both families to supply the public generators.

Theorem 4 (((q_1, q_2) -dlog implies ($\mathbf{f}, \mathbf{h}$)-find-rep). *Let $q_1, q_2 \geq 1$ satisfy $q_1 \geq \max_i \deg f_i$ and $q_2 \geq \max_j \deg h_j$, where $\deg$ denotes total degree, and put $D := q_1 + q_2$. If a PPT adversary $\mathcal{A}$ wins Fig. 3 with probability ϵ , there is an expected-polynomial-time discrete-log adversary $\mathcal{B}$ with success probability at least $\epsilon - D/p$. It runs $\mathcal{A}$ once, computes the simulated SRS, and finds the roots of a univariate polynomial of degree at most D .*

Proof. Given the (q_1, q_2) -**dlog** challenge for α , sample independent uniform $\mathbf{r}, \mathbf{s} \in \mathbb{F}^\ell$ and embed $\boldsymbol{\tau} := \mathbf{r}\alpha + \mathbf{s}$. Each $f_i(\mathbf{r}Y + \mathbf{s})$ has degree at most q_1 in Y , so its evaluation at α in $\mathbb{G}_1$ is computable from the challenge powers. The same holds for the h_j in $\mathbb{G}_2$ with degree bound q_2 . This simulates the game exactly: $\boldsymbol{\tau}$ is uniform and independent of the generators.

Run $\mathcal{A}$ on these elements. On a winning output, choose one of conditions (a)–(c) that holds and form

$$F := \boldsymbol{\mu}^\top \mathbf{f}, \quad F := \boldsymbol{\nu}^\top \mathbf{h}, \quad \text{or} \quad F := \mathbf{f}^\top \boldsymbol{\zeta} \mathbf{h}$$

for case (a), (b), or (c), respectively. In each case $F \neq 0$, $\deg F \leq D$, and $F(\boldsymbol{\tau}) = 0$. Hence $g(Y) := F(\mathbf{r}Y + \mathbf{s})$ satisfies $g(\alpha) = 0$.

Since F is chosen adaptively, we first condition on $\alpha, \boldsymbol{\tau}$, the generators, and $\mathcal{A}$'s random coins. The view of $\mathcal{A}$, and hence F , is then fixed, while $\mathbf{r}$ remains uniform: for each $\mathbf{r}$ there is exactly one $\mathbf{s} = \boldsymbol{\tau} - \mathbf{r}\alpha$. Under this conditioning,

$$g(Y) = F(\mathbf{r}(Y - \alpha) + \boldsymbol{\tau}).$$

Translating the variable does not change whether a polynomial is identically zero. Applying Lemma 7 with fixed offset $\boldsymbol{\tau}$ therefore bounds the probability that a winning output gives $g \equiv 0$ by D/p .

When g is nonzero, find all its roots in $\mathbb{F}$ and select the unique root β satisfying $\beta G_1 = \alpha G_1$, using the challenge element αG_1 . The correct root is present because $g(\alpha) = 0$. Polynomial factorization over a finite field and these root checks run in expected polynomial time. The success probability is at least $\epsilon - D/p$. $\square$

A.2 Laurent-polynomial relations

Our reference strings also contain trapdoors in denominators. Clearing them corresponds to rescaling the generators [18, Sec. 7].

Corollary 1 (find-rep for Laurent polynomials). *Let m be a monomial involving ℓ_m variables. Let $\mathbf{f}, \mathbf{h}$ be public Laurent-polynomial families, including 1 in both, such that every mf_i and mh_j is a polynomial. In the game of Fig. 3, sample the coordinates of $\boldsymbol{\tau}$ occurring in m uniformly from $\mathbb{F}^*$, and require the winning relations to be nonzero Laurent polynomials. If a PPT adversary wins with probability ϵ , then, for any*

$$q_1 \geq \max\{1, \max_i \deg(mf_i)\}, \quad q_2 \geq \max\{1, \max_j \deg(mh_j)\},$$

there is an expected-polynomial-time (q_1, q_2) -dlog adversary with success probability at least $\epsilon - (D + \ell_m)/p$, where $D := q_1 + q_2$.

Proof. Use the affine embedding of Theorem 4 for the polynomial families $m\mathbf{f}$ and $m\mathbf{h}$. Give $\mathcal{A}$ the elements $U_i = [(mf_i)(\boldsymbol{\tau})]_1$ and $W_j = [(mh_j)(\boldsymbol{\tau})]_2$, computed from the challenge powers. The event $m(\boldsymbol{\tau}) = 0$ has probability at most ℓ_m/p . Conditioned on its complement, $\boldsymbol{\tau}$ has the required distribution and

$$U_i = f_i(\boldsymbol{\tau})\hat{G}_1, \quad W_j = h_j(\boldsymbol{\tau})\hat{G}_2, \quad \hat{G}_b := m(\boldsymbol{\tau})G_b.$$

The rescaled generators $\hat{G}_1, \hat{G}_2$ are uniform and independent of each other and of τ . They are supplied by the constant entries of the Laurent families. Thus $\mathcal{A}$ sees the prescribed Laurent game and wins with unconditional probability at least $\epsilon - \ell_m/p$.

A winning relation gives a nonzero polynomial vanishing at τ : it is $m\mu^\top \mathbf{f}$ in case (a), $m\nu^\top \mathbf{h}$ in case (b), or $m^2 \mathbf{f}^\top \zeta \mathbf{h}$ in case (c). Each has degree at most D . The conditional argument of Theorem 4 still applies: conditioned on α, τ and the original generators, the rescaled view is fixed and $\mathbf{r}$ is uniform. An identically zero affine substitution contributes at most another D/p to the reduction loss, proving the claim. $\square$

A.3 From pairing equations to polynomial identities

The next lemma applies the relation game to the single-opening and batch verification equations. It allows the coefficients and the algebraic outputs to depend on the adversary's view of the SRS.

Lemma 8 (From pairing equations to polynomial identities). *Let $\mathbf{f}, \mathbf{h}$ and m be as in Corollary 1, and let $(\mathbf{U}, \mathbf{W})$ be the group elements of the game. Let $\mathcal{A}$ be an algebraic adversary that, on input $(\mathbf{U}, \mathbf{W})$, outputs elements $V_1, \dots, V_t \in \mathbb{G}_1$ with representations $V_i = \sum_{j'} \nu_{i,j'} \cdot U_{j'}$ together with coefficients $c_{i,j} \in \mathbb{F}$, and set $\phi_i := \sum_{j'} \nu_{i,j'} f_{j'}$. Then the probability that $\sum_{i,j} c_{i,j} \cdot e(V_i, W_j) = 0$ while $\sum_{i,j} c_{i,j} \cdot \phi_i \cdot h_j$ is a nonzero Laurent polynomial is at most $\text{Adv}_{\mathcal{A}'}^{\text{find-rep}}$, where $\mathcal{A}'$ runs $\mathcal{A}$ and outputs ζ with $\zeta_{j',j} := \sum_i c_{i,j} \nu_{i,j'}$.*

Proof. By bilinearity, $\sum_{i,j} c_{i,j} \cdot e(V_i, W_j) = \sum_{j',j} \zeta_{j',j} \cdot e(U_{j'}, W_j)$, and $\mathbf{f}^\top \zeta \mathbf{h} = \sum_{i,j} c_{i,j} \phi_i h_j$. Hence the event is precisely case (c) of Fig. 3 for the output ζ . $\square$

B Deferred material and proofs

The extraction arguments use the reduction tools of Section A.

B.1 Sumcheck and closed-form identities

Proof (of Lemma 1). (1) Since $Y^\ell - 1$ is monic, Euclidean division with respect to Y gives unique polynomials $p(Y), c(Y) \in \mathbb{R}[Y]$ such that

$$\Phi(Y) = p(Y) \cdot (Y^\ell - 1) + c(Y)$$

where $c(Y)$ has degree at most $\ell - 1$, and $p(Y)$ has degree at most $\ell - 2$ because Φ has degree at most $2\ell - 2$. As Φ and c agree on every ω^k ($k \in [\ell]$) and c has degree at most $\ell - 1$,

$$\sum_{k \in [\ell]} \Phi(\omega^k) = \sum_{k \in [\ell]} c(\omega^k) = \ell \cdot c(0)$$

by the Lagrange identities in Section 2. Hence the sum vanishes if and only if $c(0) = 0$, i.e., if and only if $c(Y) = r(Y) \cdot Y$ for a polynomial $r(Y)$ of degree at most $\ell - 2$, which is unique since Y is monic. This gives both the characterization and the uniqueness of the representation.

(2) Apply (1) to $\Phi := a(Y) \cdot b(Y) - \frac{1}{\ell} \cdot t$, which has degree at most $2\ell - 2$ and satisfies $\sum_{k \in [\ell]} \Phi(\omega^k) = \sum_{k \in [\ell]} a(\omega^k) \cdot b(\omega^k) - t$.

□

Proof (of Lemma 3). (1) U_k vanishes at $Y = \omega^{k'}$ for $k' \neq k$ because of the factor $\mathcal{L}_k(Y)$, and at $Y = \omega^k$ because $u(\mathbf{X}, \omega^k) = X_k$; hence the ℓ distinct linear factors of $Y^\ell - 1$ divide U_k . As $\deg_Y U_k \leq 2\ell - 2$, the quotient has degree at most $\ell - 2$. The same argument applies to Z_k (with $u(\mathbf{z}, \omega^k) = z_k$), and $M_k(0) = 0$ follows from $\mathcal{L}_k(0) = 1/\ell$ for every k .

(2) Using $\mathcal{L}_k(Y) \cdot h(\mathbf{X}, Y) = U_k - Z_k + \mathcal{L}_k(Y) \cdot (X_k - z_k)$ and $q_k \cdot (X_k - z_k) = F_{k-1} - F_k$ gives

$$\sum_{k \in [\ell]} q_k \cdot (U_k - Z_k) = q \cdot h - \sum_{k \in [\ell]} \mathcal{L}_k(Y) \cdot (F_{k-1} - F_k).$$

Summation by parts gives $\sum_{k=0}^{\ell} F_k M_k = \sum_{k \in [\ell]} (F_{k-1} - F_k) \cdot (\mathcal{L}_k(Y) - \frac{1}{\ell}) = \sum_{k \in [\ell]} \mathcal{L}_k(Y) \cdot (F_{k-1} - F_k) - \frac{1}{\ell} (f - v)$. Adding these two identities shows that the right-hand sides of Eq. (11) sum to $q \cdot h - \frac{1}{\ell} (f - v) = p \cdot (Y^\ell - 1) + r \cdot Y$. By (1), the first sum lies in $(Y^\ell - 1) \cdot \mathbb{R}_{\leq \ell-2}[Y]$ and the second in $Y \cdot \mathbb{R}_{\leq \ell-2}[Y]$, so the claim follows from the uniqueness in Lemma 1(1).

(3) For the X -degrees, $q_k \cdot X_{k'}$ has degree at most $d + 1$ in $X_{k'}$ (Lemma 2), while each F_k lies in $\mathbb{F}_{\leq d}[\mathbf{X}]$. □

B.2 Security of single openings

We prove Theorem 1. The SRS classification below also identifies the admissible key changes used by the multilinear prover.

Proof (of Theorem 1(1)). The closed forms of Lemma 3 give the prover's stated elements Q and P . Substituting them and the commitment into the verification equation gives Eq. (13), which holds whenever $v = f(\mathbf{z})$. □

The SRS as Laurent polynomials. Let $\boldsymbol{\tau}' = (\tau_1, \dots, \tau_\ell, \mu, \delta)$ be the vector of all $\ell + 2$ trapdoors, and write $\mathbf{X}' = (\mathbf{X}, Y, \Delta)$ for the corresponding formal variables. Every element of the SRS of Fig. 1 is of the form $[g(\boldsymbol{\tau}')]_1$ or $[g(\boldsymbol{\tau}')]_2$ for a Laurent polynomial g in $\mathbf{X}'$ whose denominator divides Δ . Corollary 1 gives the corresponding relation bound for these Laurent polynomials.

Shape of the SRS. Every $\mathbb{G}_1$ element of the SRS in Fig. 1 has the form $[\phi(\tau, \mu) \cdot \theta]_1$ for one of the following three *types*:

- (B) $\theta = \delta, \quad \phi = 1;$
- (E) $\theta = 1, \quad \phi \in \mathbb{R}_{\leq \ell-1}[Y]$ of degree $\leq d$ in every X_k
 (the elements $[\tau^i]_1$ of ck , $[1]_1$, $[\tau^i \mathcal{L}_k(\mu)]_1$, $[\mathcal{L}_k(\mu)]_1$, $[\tau_k \mathcal{L}_k(\mu)]_1$);
- (D) $\theta = 1/\delta, \phi \in \mathcal{V}$
 (the elements $[\tau^i U_k/\delta]_1$, $[\tau^i \mu^m(\mu^\ell - 1)/\delta]_1$, $[\tau^i M_k/\delta]_1$),

(26)

where membership in $\mathcal{V}$ for type (D) follows from Lemma 3(1). The $\mathbb{G}_2$ elements are $[\delta]_2$, $[\mathcal{L}_k(\mu)]_2$, $[u(\tau, \mu)]_2$ and the constant $[1/\ell]_2$. Consequently, any $\mathbb{G}_1$ element output by an algebraic adversary, being a linear combination of SRS elements, is of the form

$$V = \left[c_B \cdot \delta + c_E(\mathbf{X}, Y) + \frac{c_D(\mathbf{X}, Y)}{\delta} \right]_1 \Big|_{(\mathbf{X}, Y) = (\tau, \mu)} \quad (27)$$

with $c_B \in \mathbb{F}$, $c_E \in \mathbb{R}_{\leq \ell-1}[Y]$ of degree $\leq d$ in every X_k , $c_D \in \mathcal{V}$,

and the three components are determined by the representation of V . The algebraic extraction argument below uses only Eq. (26); the reduction loss also depends on the degrees of the SRS polynomials. We say that an SRS is *admissible* if all its $\mathbb{G}_1$ elements are of one of the three types and its $\mathbb{G}_2$ elements are the ones above. The proof therefore also applies to the key bases in Section 4.

Proof (of Theorem 1(2)). Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be an algebraic adversary, where $\mathcal{A}_0$ outputs C and $\mathcal{A}_1$ outputs $(z, v, \pi = (Q, P))$, together with representations of C, Q, P over the $\mathbb{G}_1$ elements of the SRS. By Eq. (27), write

$$C \leftrightarrow (c_B, c_E, c_D), \quad Q \leftrightarrow (q_B, q_E, q_D), \quad P \leftrightarrow (p_B, p_E, p_D).$$

The extractor $\mathcal{E}$ outputs $f := c_E(\mathbf{X}, 0) \in \mathbb{F}_{\leq d}[\mathbf{X}]$, the constant coefficient in Y of the type-(E) component of C (for an honest commitment, $c_E = f$). It uses only the representation that $\mathcal{A}_0$ outputs with C , as Definition 1 requires. The SRS of Fig. 1 has the distribution of (U, W) in the game of Corollary 1, with $m = \Delta$ ($\ell_m = 1$): **Setup** samples δ from $\mathbb{F}^*$ and the other trapdoors from $\mathbb{F}$.

We bound the probability that **Verify** accepts with $f(z) \neq v$ by $\text{Adv}_{\mathcal{A}'}^{\text{find-rep}}$, where $\mathcal{A}'$ is the adversary of Lemma 8. The theorem then follows from Corollary 1 with $q_1 = d\ell + 2\ell - 1$ and $q_2 = \ell + 1$. These bound the total degrees of $m \cdot g$ for the $\mathbb{G}_1$ and $\mathbb{G}_2$ SRS elements $[g]$, respectively; in particular, $\deg(\mathbf{X}^i U_k) \leq d\ell + 2\ell - 1$ and $\deg(u\Delta) \leq \ell + 1$. Thus $D = d\ell + 3\ell$ and the reduction loss is $(D + \ell_m)/p = (d\ell + 3\ell + 1)/p$.

The verification equation pairs $Q, P, C, [1]_1$ with the SRS elements $[\mathcal{L}_k(\mu)]_2$, $[u(\tau, \mu)]_2$, $[\delta]_2$ and $[1/\ell]_2$ in $\mathbb{G}_2$, using coefficients determined by z and v . By Lemma 8, unless $\mathcal{A}'$ wins the **find-rep** game, acceptance implies the following Laurent-polynomial identity in $\mathbf{X}' = (\mathbf{X}, Y, \Delta)$, where $h = h(\mathbf{X}, Y)$ is as in Eq. (7):

$$\left(q_B \Delta + q_E + \frac{q_D}{\Delta} \right) \cdot h = \left(p_B \Delta + p_E + \frac{p_D}{\Delta} \right) \cdot \Delta + \frac{1}{\ell} \left(c_B \Delta + c_E + \frac{c_D}{\Delta} - v \right). \quad (28)$$

Comparing the coefficients of the powers of Δ on both sides gives the following constraints:

Δ^b	constraint	comment
Δ^{-1}	$q_D \cdot h = c_D / \ell$	type-(D) parts of C, Q do not enter the last row
Δ	$q_B \cdot h = p_E + c_B / \ell$	honest masks: $q_B = \eta$, $p_E = \eta h - \eta' / \ell$, $c_B = \eta'$
Δ^2	$p_B = 0$	
Δ^0	$q_E \cdot h = p_D + \frac{1}{\ell}(c_E - v)$	the sumcheck identity

The last row determines the evaluation claim. Since $p_D \in \mathcal{V}$, its sum over the roots of unity vanishes (Lemma 1). Summing this row over $Y = \omega^1, \dots, \omega^\ell$ and using $h(\mathbf{X}, \omega^k) = X_k - z_k$ gives

$$\sum_{k \in [\ell]} q_E(\mathbf{X}, \omega^k) \cdot (X_k - z_k) = \frac{1}{\ell} \sum_{k \in [\ell]} c_E(\mathbf{X}, \omega^k) - v = c_E(\mathbf{X}, 0) - v,$$

where the last step uses the degree bound $\ell - 1$ on c_E in Y (Section 2, Lagrange polynomials). Substituting $\mathbf{X} := \mathbf{z}$ gives $c_E(\mathbf{z}, 0) = v$, so the extracted polynomial satisfies the evaluation claim.

Hence, whenever **Verify** accepts and $f(\mathbf{z}) \neq v$, the identity Eq. (28) fails and $\mathcal{A}'$ wins the **find-rep** game. The claimed extractability bound follows from Corollary 1 and the degree estimates above. $\square$

Proof (of Theorem 1(3)). We describe the simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$ of Definition 1.

- $\mathcal{S}_0(1^\lambda, \ell, d)$ runs **Setup** of Fig. 1 and outputs **srs** together with simulator state $\mathbf{td} = (G_1, \tau, \mu, \delta)$.
- $\mathcal{S}_1(\mathbf{td})$ picks $c \xleftarrow{\$} \mathbb{F}$ and outputs the commitment $C := c \cdot G_1$ (a uniformly random element of $\mathbb{G}_1$) and $\mathbf{st} := \perp$.
- $\mathcal{S}_2(\mathbf{td}, \mathbf{st}, C, \mathbf{z}, v)$ picks $\xi \xleftarrow{\$} \mathbb{F}$, sets $Q^* := \xi \cdot G_1$, computes $h := h(\tau, \mu) = \sum_{k \in [\ell]} \mathcal{L}_k(\mu)(\tau_k - z_k) \in \mathbb{F}$ from the trapdoor, and outputs $((Q^*, P^*), \perp)$, where

$$P^* := \frac{h}{\delta} \cdot Q^* - \frac{1}{\ell\delta} \cdot (C - v \cdot G_1).$$

The simulated proof satisfies $e(P^*, [\delta]_2) = [\xi h - (c - v)/\ell]_T$. It therefore verifies, since $e(Q^*, [h(\tau, \mu)]_2) - e(C - v[1]_1, [1/\ell]_2) = [\xi h]_T - [(c - v)/\ell]_T$.

We compare the two experiments of Definition 1 for an arbitrary polynomial $f \in \mathbb{F}_{\leq d}[\mathbf{X}]$ and adversary $\mathcal{A}$. The **srs** is distributed identically in both, since $\mathcal{S}_0$ runs **Setup**. In the real experiment, $C = [f(\tau) + \eta'\delta]_1$ is uniform in $\mathbb{G}_1$ because η' is uniform and $\delta \neq 0$; in the ideal experiment, C is uniform by construction.

Consider the i -th query $\mathbf{z}_i$, chosen adaptively by $\mathcal{A}$, and let $v_i = f(\mathbf{z}_i)$. Write $q^{(i)}$ for its interpolated quotient polynomial. In the real experiment, $Q_i = [q^{(i)}(\tau, \mu) + \eta_i\delta]_1$ is uniform and independent of C , the previous answers and $\mathbf{z}_i$, because η_i is fresh and uniform. In the ideal experiment, Q_i^* is uniform and independent by construction.

In both experiments, the second proof element is the *unique* $P \in \mathbb{G}_1$ satisfying $e(P, [\delta]_2) = e(Q_i, [h(\tau, \mu)]_2) - e(C - v_i[1]_1, [1/\ell]_2)$. This equation holds by completeness in the real experiment (Eq. (13)) and by the computation above in the ideal experiment. Uniqueness follows from non-degeneracy of e and $\delta \neq 0$: if $e(P - P', [\delta]_2) = 0$, then $P = P'$. Hence the joint distribution of $(\text{srs}, C, (z_i, v_i, Q_i, P_i)_i)$, and therefore $\mathcal{A}$'s output, is identical in both experiments. $\square$

The proof and verifier costs in Theorem 1 follow directly from Fig. 1: the proof contains Q, P , and the verifier computes H_z and $v[1]_1$ before checking three pairings.

B.3 Correctness and costs of the multilinear prover

Proof (of Lemma 6). Multilinear interpolation gives $f(\tau) = \sum_i f(i) \text{eq}(i; \tau)$; adding the same hiding mask gives the same group element C . For Q , substitute the prefix expansion of q_k (levels $k \leq \kappa$) or its suffix expansion (levels $k > \kappa$) from Lemma 5 into $q(\tau, \mu) = \sum_k q_k(\tau) \mathcal{L}_k(\mu)$ and use Eqs. (18) and (19). For P , do the same in Eq. (11), using Eq. (12) for Z_k and $F_k(j) = T_k(j)$; the mask terms are identical. For security, the four families in Eq. (17) have types (E), (D), (D), (D), respectively, as defined in Eq. (26). The unmasked commitment-key elements have type (E), with $\phi = \text{eq}(i; \mathbf{X}) \in \mathbb{F}_{\leq 1}[\mathbf{X}]$ independent of Y , while the masking element $[\delta]_1$ has type (B). The $\mathbb{G}_2$ elements are unchanged, and the degrees of the defining polynomials after clearing denominators do not exceed the bounds for Fig. 1 with $d = 1$. Hence the SRS is admissible and Remark 2 applies. $\square$

Proof (of Theorem 2). (1) The four families of Eq. (17) have $n - 1$, $n - 1$, $(n - 1)(\ell - 1)$ and $2n - 1$ elements; adding $[\delta]_1$ and the 2ℓ elements $[\mathcal{L}_k(\mu)]_1, [\tau_k \mathcal{L}_k(\mu)]_1$ gives $(\ell + 3)n + \ell - 1$.

(2) The auxiliary information consists of D , the $(\ell + 1)(2^\kappa - 1)$ elements $A_{k,i}, B_{k,i}, B_{k,i,m}$, and the $2^{\kappa+1} - 1$ elements $R_{k,i}$. Apart from the already computed hint D , each auxiliary element at level k is computed by one MSM($2^{\ell-k}$). The total MSM size is therefore

$$(\ell + 1) \sum_{k=1}^{\kappa} 2^{k-1} 2^{\ell-k} + \sum_{k=0}^{\kappa} 2^k 2^{\ell-k} = \frac{\kappa(\ell + 1)}{2} n + (\kappa + 1)n.$$

(3) The recurrence in Section 4.1 computes the tables with $n - 1$ multiplications. Computing the prefix weights $e_{k,i}$ for $k \leq \kappa$, each level from the previous one, costs fewer than $2^{\kappa+1}$ multiplications. The products $e_{k-1,i} w_{k,m}$ and $q_k(j) w_{k,m}$ cost $(\ell - 1) \sum_{k=1}^{\kappa} 2^{k-1} \leq (\ell - 1) 2^\kappa$ and $(\ell - 1) \sum_{k=\kappa+1}^{\ell} 2^{\ell-k} \leq (\ell - 1) 2^{\ell-\kappa}$ multiplications, respectively; computing the $w_{k,m}$ costs $O(\ell^2)$.

Grouping the terms for Q and P by their bases gives total MSM size at most

$$\begin{aligned}
& (\ell + 1) \left(\sum_{k=1}^{\kappa} 2^{k-1} + \sum_{k=\kappa+1}^{\ell} 2^{\ell-k} \right) \\
& + \left(\sum_{k=0}^{\kappa} 2^k + \sum_{k=\kappa+1}^{\ell} 2^{\ell-k} \right) + 2\ell + 2 \\
& \leq (\ell + 3)2^{\kappa} + (\ell + 2)2^{\ell-\kappa}.
\end{aligned}$$

The three groups of terms account, respectively, for A, B, B_m and the corresponding S elements; for R and S^M ; and for the mask terms, including $[\delta]_1$ and D .

(4) follows from Theorem 1. $\square$

B.4 Maintainable vector commitments

A maintainable VC [40,44] keeps prover state that supports openings at any position and can be updated when the vector changes.

Definition 2 (Maintainable vector commitments). *A vector commitment for $\mathbf{v} \in \mathbb{F}^n$ consists of five efficient algorithms:*

- $\text{Setup}(1^\lambda, n) \rightarrow \text{pp}$ generates public parameters.
- $\text{Init}(\text{pp}, \mathbf{v}) \rightarrow (C, \text{st})$ computes the commitment C and initializes the maintainer's state st .
- $\text{Update}(\text{pp}, C, \text{st}, i, \epsilon) \rightarrow (C', \text{st}')$ updates the commitment and state for the change $v_i \leftarrow v_i + \epsilon$, where $i \in [n]$ and $\epsilon \in \mathbb{F}$.
- $\text{Query}(\text{pp}, C, \text{st}, i) \rightarrow (v, \pi)$ returns the current entry $v = v_i$ at $i \in [n]$ and its opening proof, leaving C and st unchanged.
- $\text{Verify}(\text{pp}, C, i, v, \pi) \rightarrow \{0, 1\}$ checks the opening.

Correctness. After honest setup and initialization of any vector, each query returns the current entry and an accepted proof with probability one, throughout any polynomial-length sequence of adaptively chosen updates and queries.

Position binding. For every polynomially bounded $n = n(\lambda)$ and every PPT adversary given $\text{pp} \leftarrow \text{Setup}(1^\lambda, n)$, the probability of producing a commitment C and two accepted openings (i, v, π) and (i, v', π') , with $i \in [n]$ and $v \neq v'$, is negligible in λ . In the AGM, this requirement is restricted to algebraic adversaries.

Maintainability. The scheme is efficiently maintainable if, after initialization, updates and queries each take sublinear work in n , suppressing dependence on λ . Costs are worst-case over all reachable states unless stated otherwise, assuming random access to the keys and state and in-place updates. Initialization and storage costs are reported separately.

We treat $\mathbf{v}$ as the hypercube table T_0 of its MLE, with position $i \in [n]$ represented by the binary encoding of $i - 1$. Initialization precomputes all prefix levels ($\kappa = \ell$).

Corollary 2. *Let $\ell \geq 1$, $\ell \mid p-1$ and $n = 2^\ell$. There is a vector commitment satisfying Definition 2 in the AGM under the $(3\ell-1, \ell+1)$ -dlog assumption, with a one-element commitment and two-element individual proofs in $\mathbb{G}_1$, and the following costs.*

1. *The public keys and auxiliary state each contain $O(n \log n)$ group elements; the state also stores the n vector entries. Preprocessing costs $O(n \log^2 n)$ MSM terms.*
2. *An update changes the commitment and auxiliary state with at most $\ell(\ell+3)+2$ scalar multiplications and as many group additions in $\mathbb{G}_1$.*
3. *A query uses at most $\ell^2+O(\ell)$ group additions in $\mathbb{G}_1$, plus two scalar multiplications in the hiding variant. Verification uses three pairings, $O(\ell)$ additions in $\mathbb{G}_2$ and one scalar multiplication in $\mathbb{G}_1$.*

These bounds hold after any polynomial-length sequence of updates.

This variant also replaces the S^Z key family and its cached encodings with the fixed family defined below. Each Z_k is then expressed using the query coordinates as coefficients. At Boolean points, these coefficients and the prefix weights are bits, so the prover combines cached elements using group additions, apart from masking. A single-entry update affects only one prefix per level in each auxiliary family.

The same keys and state support openings at arbitrary field points with $O(\sqrt{n} \log n)$ MSM terms and $O(n)$ field operations, using prefix expansions up to level $\lfloor \ell/2 \rfloor$ and suffix expansions thereafter.

These results concern individual openings and maintenance of the prover’s shared state; aggregation, updates to separately stored client proofs, and use of the tagged batch setup (Section 5) require separate analysis. Updates retain the commitment mask and D . The zero-knowledge guarantee applies to repeated openings of a fixed vector and does not cover the update history.

Table 4. Individual openings for a vector of length n . Proof sizes count group elements. Update, query and verification costs suppress dependence on the security parameter. Hyperproofs and BalanceProofs costs follow [44, Table 1 and Section 4.3]. The Couplet row uses full preprocessing ($\kappa = \ell$) and the correction basis of Corollary 2.

Scheme	Proof	Update	Query	Verify
Hyperproofs [40]	$\log n$	$O(\log n)$	$O(\log n)$	$O(\log n)$
BalanceProofs [44]	1	$O(\sqrt{n} \log n)$	$O(\sqrt{n})$	$O(1)$
Two-layer variant	3	$O(n^{1/4} \log n)$	$O(n^{1/4})$	$O(1)$
Couplet	2	$O(\log^2 n)$	$O(\log^2 n)$	$O(\log n)$

The other schemes in Table 4 use $O(n)$ public parameters; Couplet uses $O(n \log n)$ group elements in both its public keys and auxiliary state. At $n = 2^{16}$, the auxiliary group elements alone occupy about 60 MiB with 48-byte encodings.

Keys and initialization. Use ck and vk from Fig. 2. For $k, m \in [\ell]$, define

$$A_{k,m}(Y) := \mathcal{L}_k(Y)\mathcal{L}_m(Y) - [k = m]\mathcal{L}_k(Y), \quad Z_k(Y) = \sum_{m=1}^{\ell} z_m A_{k,m}(Y).$$

Replace the S^Z family in pk by

$$S_{k,j,m}^A := [\text{eq}(\mathbf{j}; \boldsymbol{\tau}_{[k+1:]})A_{k,m}(\mu)/\delta]_1, \quad \mathbf{j} \in \{0,1\}^{\ell-k}, \quad k, m \in [\ell],$$

and keep the S^Q, S^U, S^M and masking elements. Initialize C, D as in Fig. 2, and precompute $A_{k,i}, B_{k,i}, R_{k,i}$ for all levels. In place of $B_{k,i,m}$, store

$$B_{k,i,m}^A := [\hat{q}_{k,i}(\boldsymbol{\tau})A_{k,m}(\mu)/\delta]_1 = \sum_{\mathbf{j} \in \{0,1\}^{\ell-k}} (T_0(\mathbf{i}, 1, \mathbf{j}) - T_0(\mathbf{i}, 0, \mathbf{j})) S_{k,j,m}^A.$$

The state st consists of T_0, D and these four auxiliary families; the maintainer also retains C . No table depending on a future query is precomputed.

Update. To increment the entry at position $\mathbf{a} \in \{0,1\}^{\ell}$ by $\epsilon \in \mathbb{F}$, set

$$T_0(\mathbf{a}) \leftarrow T_0(\mathbf{a}) + \epsilon, \quad C \leftarrow C + \epsilon[\text{eq}(\mathbf{a}; \boldsymbol{\tau})]_1.$$

For each $k \in [\ell]$, write $\mathbf{i} = \mathbf{a}_{[:k-1]}$, $\mathbf{j} = \mathbf{a}_{[k+1:]}$ and $\epsilon_k = (2a_k - 1)\epsilon$, and update

$$\begin{aligned} A_{k,i} &\leftarrow A_{k,i} + \epsilon_k S_{k,j}^Q, & B_{k,i} &\leftarrow B_{k,i} + \epsilon_k S_{k,j}^U, \\ B_{k,i,m}^A &\leftarrow B_{k,i,m}^A + \epsilon_k S_{k,j,m}^A & \text{for every } m \in [\ell]. \end{aligned}$$

For each $k \in [0, \ell]$, also set

$$R_{k,\mathbf{a}_{[:k]}} \leftarrow R_{k,\mathbf{a}_{[:k]}} + \epsilon S_{k,\mathbf{a}_{[k+1:]}}^M.$$

All other entries and D remain fixed. To replace the stored value v_{old} with v_{new} , set $\epsilon = v_{\text{new}} - v_{\text{old}}$.

Boolean query and verification. Given $\mathbf{z} \in \{0,1\}^{\ell}$, read $v = T_0(\mathbf{z})$ and write $\mathbf{i}_k := \mathbf{z}_{[:k]}$ for $k \in [0, \ell]$. Sample $\eta \xleftarrow{\$} \mathbb{F}$ in the hiding variant, and output (v, Q, P) with

$$\begin{aligned} Q &= \sum_{k=1}^{\ell} A_{k,\mathbf{i}_{k-1}} + \eta[\delta]_1, \\ P &= \sum_{k=1}^{\ell} \left(B_{k,\mathbf{i}_{k-1}} - \sum_{m: z_m=1} B_{k,\mathbf{i}_{k-1},m}^A \right) + \sum_{k=0}^{\ell} R_{k,\mathbf{i}_k} + \eta H_1(\mathbf{z}) - D, \quad (29) \\ H_1(\mathbf{z}) &= \sum_{k=1}^{\ell} [\tau_k \mathcal{L}_k(\mu)]_1 - \sum_{k: z_k=1} [\mathcal{L}_k(\mu)]_1. \end{aligned}$$

Take $\eta = 0$ and $D = 0$ without hiding. The proof is (Q, P) ; the claimed value v is separate. Verification is that of Fig. 1. In particular, H_z is formed using only additions in $\mathbb{G}_2$ when $\mathbf{z}$ is Boolean.

Proof (of Corollary 2). **State invariant and correctness.** At initialization, every stored group element has the value specified above or in Eq. (19). If entry $\mathbf{a}$ changes by ϵ , then f changes by $\epsilon \text{eq}(\mathbf{a}; \mathbf{X})$. Consequently, $\hat{q}_{k,i}$ changes only for $i = \mathbf{a}_{[k-1]}$, by

$$(2a_k - 1)\epsilon \text{eq}(\mathbf{a}_{[k+1:]}; \mathbf{X}_{[k+1:]}) ,$$

and $\hat{F}_{k,i}$ changes only for $i = \mathbf{a}_{[k]}$, by $\epsilon \text{eq}(\mathbf{a}_{[k+1:]}; \mathbf{X}_{[k+1:]})$. The update algorithm adds exactly the encodings of these changes. By induction, the state equals freshly precomputed state for the current vector after every update, with the original commitment mask.

At a Boolean query, only the prefix matching $\mathbf{z}$ has nonzero interpolation weight. Substituting $Z_k = \sum_m z_m \Lambda_{k,m}$ into Lemma 3 therefore gives exactly Eq. (29). The proof elements agree with those of Fig. 1, so completeness follows.

Position binding. Each $\Lambda_{k,m}$ vanishes on $\{\omega^k : k \in [\ell]\}$ and has degree at most $2\ell - 2$. Thus $\text{eq}(\mathbf{j}; \mathbf{X}_{[k+1:]})\Lambda_{k,m}(Y)$ belongs to $\mathcal{V}$, and the replacement key elements have type (D) in Eq. (26), within the existing degree bounds. The remaining key elements are unchanged. By Remark 2, the extractor of Theorem 1 still extracts a multilinear polynomial from the commitment representation before any opening is given. Two accepted openings of that commitment at the same Boolean position must both equal the extracted polynomial's value, except with negligible probability under the stated assumption. This is position binding. The argument applies to commitments and openings chosen after updates as well. The masking elements also preserve zero knowledge for repeated openings of a fixed vector.

Costs. The quotient-related key families have $n - 1$, $n - 1$ and $\ell(n - 1)$ entries, and the S^M family has $2n - 1$. Including masking entries, pk has $(\ell + 4)n + \ell - 2$ listed $\mathbb{G}_1$ elements. Together with ck , the count is $(\ell + 5)n + \ell - 1$; vk has $\ell + 2$ elements in $\mathbb{G}_2$. The auxiliary families and D contain

$$(\ell + 2)(n - 1) + (2n - 1) + 1 = (\ell + 4)n - \ell - 2$$

group elements. At level k , computing the A entries, the B entries, or the B^A entries for any fixed m requires 2^{k-1} MSMs of size $2^{\ell-k}$, totaling $n/2$ terms. The R family uses n terms per level, including level zero. Hence preprocessing uses

$$\frac{\ell(\ell + 2)}{2}n + (\ell + 1)n = \left(\frac{\ell(\ell + 4)}{2} + 1\right)n$$

MSM terms, besides computing C, D . The required field differences take $O(n\ell)$ operations.

An update touches $\ell + 2$ quotient-related entries at each of ℓ levels, $\ell + 1$ entries of R , and C . This requires at most $\ell(\ell + 3) + 2$ scalar multiplications and the same number of group additions, with no rebuilding step. A Boolean query sums at most ℓ^2 entries from B^A and $O(\ell)$ other elements, using only two scalar multiplications for masking. The verifier adds at most ℓ key elements to form H_z , computes $v[1]_1$, and checks three pairings. These bounds do not depend on the number of preceding updates. $\square$

Openings at arbitrary field points. Retain the same keys and full maintained state, but choose $\kappa = \lfloor \ell/2 \rfloor$ at query time. For $k \leq \kappa$, use the prefix expansions of Lemma 5; the coefficients of $B_{k,i,m}^A$ in P are $-e_{k-1,i}z_m$. For $k > \kappa$, compute the usual folded tables and use the suffix keys, with coefficients $-q_k(j)z_m$ for $S_{k,j,m}^A$. The A, B, R terms use the same prefix/suffix split as Fig. 2. The identity for Z_k holds at arbitrary field points, so this produces the same proof. Replacing $\ell - 1$ correction terms by ℓ changes only the constants in the geometric sums of Section B.3: the total MSM size is $O(\ell(2^\kappa + 2^{\ell-\kappa}))$ and field work is $O(n)$. No second cache or additional key family is needed.

The corollary assumes the unpadding domain. The padding method of Remark 1 also applies here: inactive quotient levels are zero and the final partial evaluations are all v . Their M_k terms can be combined into the last active level. When a supported order $t = O(\ell)$ is available, the asymptotic maintenance, query and storage bounds remain the same; the reduction uses the larger padded degrees.

B.5 Batch openings

This subsection gives the formal interface and the sumcheck lemma used in Section 5. It also specifies the full algorithms, analyzes their costs, and proves Theorem 3.

Definition 3 ((Multivariate) PCs with batch openings). A multivariate polynomial commitment (PC) scheme with batch openings is a tuple of algorithms $\text{PC} = (\text{Setup}, \text{Commit}, \text{Eval}, \text{Verify})$ as in Definition 1, with the following differences:

- $\text{PC.Setup}(1^\lambda, \ell, d, s) \rightarrow \text{srs}$ takes an additional parameter s , the maximum number of evaluation points in a batch.
- $\text{PC.Eval}(\text{pk}, f, ((z_i, v_i))_{i \in [s]}; \text{aux}) \rightarrow \pi$ outputs a proof that $v_i = f(z_i)$ for all $i \in [s]$.
- $\text{PC.Verify}(\text{vk}, C, ((z_i, v_i))_{i \in [s]}, \pi) \rightarrow 0/1$ outputs a single bit.
- Completeness, extractability and zero knowledge are adapted to batches. Extractability requires the extracted f to satisfy $f(z_i) = v_i$ for all i . In the zero-knowledge experiments, each query consists of s points and is answered with the s values and one proof.

A batch is an ordered tuple; repeated evaluation pairs are allowed. For a nonempty batch of fewer than s pairs, repeat one pair until all s slots are filled. The prover and verifier use the same slot order.

Lemma 9 (Univariate sumcheck over cosets).

1. A polynomial $\Phi \in \mathbb{R}_{\leq 2s\ell-2}[Y]$ lies in $\mathcal{V}_s$ if and only if $\sum_{k \in [\ell]} \Phi(\rho_{i,k}) = 0$ for every $i \in [s]$, and the representation Eq. (21) of an element of $\mathcal{V}_s$ is unique.

2. For $a(Y), b(Y) \in \mathbb{R}[Y]$ of degree at most $s\ell - 1$ and $t_1, \dots, t_s \in \mathbb{R}$, the identities $\sum_{k \in [\ell]} a(\rho_{i,k}) \cdot b(\rho_{i,k}) = t_i$ hold for every $i \in [s]$ if and only if there exist unique $p \in \mathbb{R}_{\leq s\ell-2}[Y]$ and $r_i \in \mathbb{R}_{\leq \ell-2}[Y]$ such that

$$a(Y) \cdot b(Y) = p(Y) \cdot (Y^{s\ell} - 1) + \sum_{i \in [s]} \mathcal{L}_i^*(Y^\ell) \cdot \left(r_i(Y) \cdot Y + \frac{1}{\ell} \cdot t_i \right). \quad (30)$$

Proof (of Lemma 9). (1) Euclidean division by $Y^{s\ell} - 1$ gives unique $p, c \in \mathbb{R}[Y]$ with $\Phi = p \cdot (Y^{s\ell} - 1) + c$, $\deg_Y c \leq s\ell - 1$ and $\deg_Y p \leq s\ell - 2$. The polynomials c and Φ agree on all $s\ell$ points. Define $c_i(Y) := \sum_{k \in [\ell]} c(\rho_{i,k}) \cdot \mathcal{L}_k^{(i)}(Y)$, the unique polynomial of degree at most $\ell - 1$ that agrees with c on coset i . The polynomial $c' := \sum_{i \in [s]} \mathcal{L}_i^*(Y^\ell) \cdot c_i(Y)$ has degree at most $\ell(s - 1) + \ell - 1 = s\ell - 1$ and agrees with c at every $\rho_{i',k'}$: only the term $i = i'$ survives, and $c_{i'}(\rho_{i',k'}) = c(\rho_{i',k'})$. Thus $c = c'$, since $\mathbb{R}$ is an integral domain. Any representation $c = \sum_i \mathcal{L}_i^*(Y^\ell) \tilde{c}_i$ with $\deg_Y \tilde{c}_i \leq \ell - 1$ has $\tilde{c}_i = c_i$, as seen by evaluating on coset i . By Eq. (20), $\sum_k \Phi(\rho_{i,k}) = \sum_k c_i(\rho_{i,k}) = \ell \cdot c_i(0)$. Hence all coset sums vanish if and only if $c_i(0) = 0$ for every i , or equivalently, $c_i = r_i \cdot Y$ for a unique r_i of degree at most $\ell - 2$. This is the representation in Eq. (21).

(2) Apply (1) to $\Phi := a \cdot b - \frac{1}{\ell} \sum_i \mathcal{L}_i^*(Y^\ell) t_i$, whose sum over coset i equals $\sum_k a(\rho_{i,k}) b(\rho_{i,k}) - t_i$, because $\mathcal{L}_{i'}^*(\rho_{i,k}^\ell) = \mathcal{L}_{i'}^*(\theta^{i-1})$ is 1 for $i' = i$ and 0 otherwise. $\square$

General prover costs. Let $N = (d + 1)^\ell$. Including the hiding terms, the prover key in Fig. 4 lists $N(4s\ell + s - 1) + 2s\ell + 1$ elements of $\mathbb{G}_1$, approximately $4Ns\ell$, and the verification key lists $s\ell + s + 3$ elements of $\mathbb{G}_2$. Computing the commitment and mask hint uses one $\text{MSM}(N)$ and two scalar multiplications in $\mathbb{G}_1$. An opening uses $O(s^2 N \ell)$ field operations and at most $N(s\ell + 4s) + 2s\ell + 2$ MSM terms in $\mathbb{G}_1$. For the latter bound, combine the contributions of all claims to each shared correction base $[\tau^i \mu^m (\mu^{s\ell} - 1) / \delta]_1$ before the MSM.

Remark 6 (Multilinear polynomials). Let $d = 1$ and $n = 2^\ell$. The two-sided expansions of Lemma 5 apply to each $q_{i,k}$ and $F_k^{(i)}$. For the prover key, use the level structure of Section 4.2, replacing $\mathcal{L}_k$, U_k , $\mu^m(\mu^\ell - 1)$ and M_k by $\mathcal{L}_{i,k}$, $U_{i,k}$, $\mu^m(\mu^{s\ell} - 1)$ and $\mathcal{L}_i^*(\mu^\ell) M_k^{(i)}$, respectively. This gives about $(s\ell + 4s)n$ elements of $\mathbb{G}_1$.

Fix the level parameter κ and precompute the corresponding prefix encodings as in Section 4.2. The families weighted by $\mathcal{L}_{i,k}$, $U_{i,k}$ and $\mathcal{L}_i^*(\mu^\ell) M_k^{(i)}$ have separate encodings for each slot i . The correction encodings $[\hat{q}_{k,i}(\tau) \mu^m (\mu^{s\ell} - 1) / \delta]_1$, for $m \in [0, s\ell - 2]$, are shared across all slots. For each shared base, sum its scalar contributions over i before the MSM; for a cached correction encoding, the scalar is $-\sum_{i \in [s]} \text{eq}(\mathbf{i}; \mathbf{z}_{i, [k-1]}) w_{k,m}^{(i)}$.

Preprocessing produces about $(s\ell + 4s)2^\kappa$ auxiliary group elements and has total MSM size $\frac{\kappa(s\ell + 4s - 1)}{2} n + sn$, excluding the mask hint. These encodings are independent of the evaluation points and can be reused across batches for the

Setup($1^\lambda, \ell, d, s$) $\rightarrow$ (ck, pk, vk):

- Pick $\tau \xleftarrow{\$} \mathbb{F}^\ell$, $\mu \xleftarrow{\$} \mathbb{F}$, $\delta, \gamma \xleftarrow{\$} \mathbb{F}^*$, and let $U_{i,k}, M_k^{(i)}$ be as defined in Section 5.2.
- Output, with i ranging over $[0, d]^\ell$, i over $[s]$, k over $[\ell]$, k' over $[0, \ell]$ and m over $[0, s\ell - 2]$,

$$\begin{aligned} \text{ck} &= \left(\left(\left[\frac{\tau^i}{\gamma} \right]_1 \right)_i, \left[\frac{\delta}{\gamma} \right]_1 \right), \\ \text{pk} &= \left(\begin{array}{c} [\delta]_1, ([\mathcal{L}_{i,k}(\mu)]_1, [\tau_k \mathcal{L}_{i,k}(\mu)]_1)_{i,k}, \\ ([\tau^i \mathcal{L}_{i,k}(\mu)]_1)_{i,i,k}, \left(\left[\frac{\tau^i U_{i,k}(\tau, \mu)}{\delta} \right]_1 \right)_{i,i,k}, \\ \left(\left[\frac{\tau^i \mu^m (\mu^{s\ell} - 1)}{\delta} \right]_1 \right)_{i,m}, \left(\left[\frac{\tau^i \mathcal{L}_i^*(\mu^\ell) M_{k'}^{(i)}(\mu)}{\delta} \right]_1 \right)_{i,i,k'} \end{array} \right), \\ \text{vk} &= \left([\delta]_2, \left[\frac{\gamma}{\ell} \right]_2, ([\mathcal{L}_{i,k}(\mu)]_2)_{i,k}, H_\tau := [u(\tau, \mu)]_2, \left(\left[\frac{\mathcal{L}_i^*(\mu^\ell)}{\ell} \right]_2 \right)_i \right). \end{aligned}$$

Commit(ck, f) $\rightarrow$ (C , aux): write $f = \sum_i f_i \mathbf{X}^i$, pick $\eta' \xleftarrow{\$} \mathbb{F}$, and output $C = \sum_i f_i \cdot \left[\frac{\tau^i}{\gamma} \right]_1 + \eta' \cdot \left[\frac{\delta}{\gamma} \right]_1$ and $\text{aux} = D := \frac{\eta'}{\ell} \cdot [1]_1$.

Eval(pk, $f, ((z_i, v_i))_{i \in [s]}$; aux) $\rightarrow \pi$:

- Compute $F_k^{(i)} = \sum_i F_{k,i}^{(i)} \mathbf{X}^i$, $q_{i,k} = \sum_i q_{i,k,i} \mathbf{X}^i$ and the coefficients $w_{k,m}^{(i)}$.
- Parse D from aux and pick $\eta \xleftarrow{\$} \mathbb{F}$.
- Output $\pi = (Q, P)$ with

$$\begin{aligned} Q &= \sum_{i,k} \sum_i q_{i,k,i} \cdot [\tau^i \mathcal{L}_{i,k}(\mu)]_1 + \eta \cdot [\delta]_1, \\ P &= \sum_{i,k} \sum_i q_{i,k,i} \cdot \left(\left[\frac{\tau^i U_{i,k}(\tau, \mu)}{\delta} \right]_1 - \sum_{m=0}^{s\ell-2} w_{k,m}^{(i)} \cdot \left[\frac{\tau^i \mu^m (\mu^{s\ell} - 1)}{\delta} \right]_1 \right) \\ &\quad + \sum_{i \in [s]} \sum_{k=0}^\ell \sum_i F_{k,i}^{(i)} \cdot \left[\frac{\tau^i \mathcal{L}_i^*(\mu^\ell) M_k^{(i)}(\mu)}{\delta} \right]_1 \\ &\quad + \eta \cdot \sum_{i,k} ([\tau_k \mathcal{L}_{i,k}(\mu)]_1 - z_{i,k} \cdot [\mathcal{L}_{i,k}(\mu)]_1) - D, \end{aligned}$$

i.e., $Q = [q(\tau, \mu) + \eta \delta]_1$ and $P = [\Phi(\tau, \mu)/\delta + \eta h(\tau, \mu) - \eta'/\ell]_1$.

Verify(vk, $C, ((z_i, v_i))_{i \in [s]}$, π) $\rightarrow$ 0/1:

- Compute $H_z = \sum_{i,k} z_{i,k} \cdot [\mathcal{L}_{i,k}(\mu)]_2$ and $V_v = \sum_{i \in [s]} v_i \cdot \left[\frac{\mathcal{L}_i^*(\mu^\ell)}{\ell} \right]_2$.
- Accept if and only if

$$e(Q, H_\tau - H_z) = e(P, [\delta]_2) + e(C, \left[\frac{\gamma}{\ell} \right]_2) - e([1]_1, V_v).$$

Fig. 4. The batch variant of Couplet, supporting openings at up to s points. Orange marks the terms needed only for zero knowledge.

same commitment. The total MSM size per batch opening is about $(s\ell + 4s)(2^\kappa + 2^{\ell-\kappa})$ in $\mathbb{G}_1$.

For even ℓ , choosing $\kappa = \ell/2$ gives about $(s\ell + 4s)\sqrt{n}$ auxiliary elements and $O(s\sqrt{n} \log n)$ opening MSM terms. Each batch opening uses $O(sn + s^2\ell\sqrt{n})$ field operations, including computation of the coefficients $w_{k,m}^{(i)}$.

Proof (of Theorem 3). Completeness. Let $v_i = f(\mathbf{z}_i)$ for all i . By Eqs. (23) and (25), the prover's elements are $Q = [q(\boldsymbol{\tau}, \mu) + \eta\delta]_1$ and $P = [\Phi(\boldsymbol{\tau}, \mu)/\delta + \eta h(\boldsymbol{\tau}, \mu) - \eta'/\ell]_1$ with $\Phi = q \cdot h - \frac{1}{\ell}(f - \sum_i v_i \mathcal{L}_i^*(Y^\ell))$, and the commitment is $C = [(f(\boldsymbol{\tau}) + \eta'\delta)/\gamma]_1$. Evaluating at the trapdoors,

$$(q + \eta\delta) \cdot h = \left(\frac{\Phi}{\delta} + \eta h - \frac{\eta'}{\ell}\right) \cdot \delta + \frac{f + \eta'\delta}{\gamma} \cdot \frac{\gamma}{\ell} - \frac{1}{\ell} \sum_{i \in [s]} v_i \mathcal{L}_i^*(\mu^\ell),$$

since the right-hand side equals $\Phi + \eta\delta h + \frac{f}{\ell} - \frac{1}{\ell} \sum_i v_i \mathcal{L}_i^*(\mu^\ell) = q \cdot h + \eta\delta h$. This is the verification equation of Fig. 4 in the exponent.

Extractability. We follow the proof of Theorem 1(2), now with the $\ell + 3$ trapdoors $(\boldsymbol{\tau}, \mu, \delta, \gamma)$ and the formal variables $(\mathbf{X}, Y, \Delta, \Gamma)$. Every $\mathbb{G}_1$ SRS element in Fig. 4 has one of five types. The commitment key supplies type (A), with $\theta = 1/\gamma$ and $\phi \in \mathbb{F}_{\leq d}[\mathbf{X}]$ (the elements $[\boldsymbol{\tau}^i/\gamma]_1$ of ck), and type (B'), with $\theta = \delta/\gamma$ and $\phi = 1$ (the masking element of ck). Type (B) has $\theta = \delta$ and $\phi = 1$. Type (E) has $\theta = 1$ and $\phi \in \mathbb{R}_{\leq s\ell-1}[Y]$ of degree at most d in every X_k ; it contains $[1]_1$, $[\mathcal{L}_{i,k}(\mu)]_1$, $[\tau_k \mathcal{L}_{i,k}(\mu)]_1$ and $[\boldsymbol{\tau}^i \mathcal{L}_{i,k}(\mu)]_1$. Type (D) has $\theta = 1/\delta$ and $\phi \in \mathcal{V}_s$: membership of $\mathbf{X}^i U_{i,k}(\mathbf{X}, Y)$, $\mathbf{X}^i Y^m (Y^{s\ell} - 1)$ and $\mathbf{X}^i \mathcal{L}_i^*(Y^\ell) M_k^{(i)}(Y)$ in $\mathcal{V}_s$ follows from the batch analogue of Lemma 3(1). The $\mathbb{G}_2$ elements are $[\delta]_2$, $[\gamma/\ell]_2$, $[\mathcal{L}_{i,k}(\mu)]_2$, $[u(\boldsymbol{\tau}, \mu)]_2$ and $[\mathcal{L}_i^*(\mu^\ell)/\ell]_2$.

Write the representations as $C \leftrightarrow (c_A, c_{B'}, c_B, c_E, c_D)$, $Q \leftrightarrow (q_A, \dots)$ and $P \leftrightarrow (p_A, \dots)$, with $c_A \in \mathbb{F}_{\leq d}[\mathbf{X}]$, $c_{B'}, c_B \in \mathbb{F}$, $c_E \in \mathbb{R}_{\leq s\ell-1}[Y]$ and $c_D \in \mathcal{V}_s$, and likewise for Q and P . The extractor outputs $f := c_A \in \mathbb{F}_{\leq d}[\mathbf{X}]$ and $\eta' := c_{B'}$ using only the representation of C . The verification equation pairs Q, P, C and the SRS element $[1]_1$ with $\mathbb{G}_2$ SRS elements. By Lemma 8, acceptance implies the following identity unless the **find-rep** game is won:

$$\begin{aligned} \left(\frac{q_A}{\Gamma} + \frac{q_{B'} \Delta}{\Gamma} + q_B \Delta + q_E + \frac{q_D}{\Delta}\right) \cdot h &= \left(\frac{p_A}{\Gamma} + \frac{p_{B'} \Delta}{\Gamma} + p_B \Delta + p_E + \frac{p_D}{\Delta}\right) \cdot \Delta \\ &+ \left(\frac{c_A}{\Gamma} + \frac{c_{B'} \Delta}{\Gamma} + c_B \Delta + c_E + \frac{c_D}{\Delta}\right) \cdot \frac{\Gamma}{\ell} - \frac{1}{\ell} \sum_{i \in [s]} v_i \mathcal{L}_i^*(Y^\ell), \end{aligned}$$

Here $h = h(\mathbf{X}, Y)$ is as in Eq. (22). We compare coefficients of the monomials $\Gamma^a \Delta^b$. The monomials Γ , $\Gamma \Delta$ and $\Gamma \Delta^{-1}$ occur only in $C \cdot \Gamma/\ell$, so $c_E = c_B = c_D = 0$ and hence $C = [(c_A + c_{B'}\delta)/\gamma]_1$. The coefficient of $\Gamma^0 \Delta^0$ gives the sumcheck identity:

$$q_E \cdot h = p_D + \frac{1}{\ell} \left(c_A - \sum_{i \in [s]} v_i \mathcal{L}_i^*(Y^\ell) \right) = p_D + \sum_{i \in [s]} \mathcal{L}_i^*(Y^\ell) \cdot \frac{c_A - v_i}{\ell} \quad \text{with } p_D \in \mathcal{V}_s,$$

using $\sum_i \mathcal{L}_i^*(Y^\ell) = 1$. By Lemma 9(1), the sum of p_D over every coset vanishes; summing the identity over coset i , where $h(\mathbf{X}, \rho_{i,k}) = X_k - z_{i,k}$ and $\mathcal{L}_{i'}^*(\rho_{i,k}^\ell) = [i' = i]$, gives for every $i \in [s]$

$$\sum_{k \in [\ell]} q_E(\mathbf{X}, \rho_{i,k}) \cdot (X_k - z_{i,k}) = c_A(\mathbf{X}) - v_i,$$

and substituting $\mathbf{X} := \mathbf{z}_i$ gives $c_A(\mathbf{z}_i) = v_i$. The tag γ ensures that c_A is free of Y and therefore the same on every coset.

Reduction bound. The total degrees of $\Gamma \Delta \cdot g$ over the $\mathbb{G}_1$ and $\mathbb{G}_2$ SRS elements are at most $d\ell + 2s\ell$ and $s\ell + 2$, respectively: $\deg(\mathbf{X}^i U_{i,k} \Gamma) \leq d\ell + 2s\ell$ and $\deg(u\Gamma \Delta) \leq s\ell + 2$, and the remaining families satisfy the same bounds. As Setup samples δ, γ from $\mathbb{F}^*$, the SRS is distributed exactly as in the game of Corollary 1 with $m = \Delta\Gamma$, so Corollary 1 with $D = d\ell + 3s\ell + 2$ and $\ell_m = 2$ gives the stated bound $(D + \ell_m)/p = (d\ell + 3s\ell + 4)/p$.

Zero knowledge. Use the simulator for Theorem 1(3) in Section B.2, with $\mathcal{S}_0$ running the batch setup of Fig. 4 and retaining $\text{td} = (G_1, \boldsymbol{\tau}, \mu, \delta, \gamma)$. The algorithm $\mathcal{S}_1$ again outputs a uniform commitment. For a query $(\mathbf{z}_i, v_i)_{i \in [s]}$, $\mathcal{S}_2$ samples a uniform $Q^* \in \mathbb{G}_1$, computes $h := \sum_{i,k} \mathcal{L}_{i,k}(\mu)(\tau_k - z_{i,k})$, and outputs $((Q^*, P^*), \perp)$ with

$$P^* := \frac{h}{\delta} \cdot Q^* - \frac{\gamma}{\ell\delta} \cdot C + \frac{1}{\ell\delta} \left(\sum_{i \in [s]} v_i \mathcal{L}_i^*(\mu^\ell) \right) \cdot G_1.$$

The real commitment $[(f(\boldsymbol{\tau}) + \eta'\delta)/\gamma]_1$ is uniform because $\delta/\gamma \neq 0$, and each real Q is independently uniform because its mask is fresh. The displayed P^* is the unique element satisfying the batch verification equation for C, Q^* and the public claim. The same conditional-distribution argument as for single openings therefore gives perfect zero knowledge for adaptive repeated batch queries.

Proof and verifier costs. The proof has the two elements Q, P . The verifier computes H_z and V_v with $s\ell + s$ scalar multiplications in $\mathbb{G}_2$ and checks four pairings, as listed in Fig. 4. $\square$

C AI tool usage

GPT 6 Astra and Claude Fable 5.1 were used to help implement our scheme (Section 1.3), polish the writing and verify the technical details in this paper.