

MERCURY: A multilinear Polynomial Commitment Scheme with constant proof size and linear field work

Liam Eagen

Alpen Labs

Ariel Gabizon

Aztec Labs

April 28, 2025

Abstract

We construct a pairing-based polynomial commitment scheme for multilinear polynomials of size n where constructing an opening proof requires $O(n)$ field operations, and $2n + O(\sqrt{n})$ scalar multiplications. Moreover, the opening proof consists of a constant number of field elements. This is a significant improvement over previous works which would require either

1. $O(n \log n)$ field operations; or
2. $O(\log n)$ size opening proof.

The main technical component is a new method of verifiably folding a witness via univariate polynomial division. As opposed to previous methods, the proof size and prover time remain constant *regardless of the folding factor*.

1 Introduction

Polynomial Commitment Schemes (PCSs)[KZG10] allow a party to commit to a polynomial and later prove an evaluation of the polynomial is correct. That is, for a commitment cm and values a, b ; a prover \mathbf{P} can produce a proof that $\text{cm} = \text{com}(f(X))$ and $f(a) = b$. PCSs form an essential part of most modern Succinct Non-interactive Arguments of Knowledge (SNARKs). They allow a protocol designer to focus on designing a so-called polynomial Interactive Oracle Proof which can then be compiled, via a PCS, to a SNARK (see [BFS19, GWC19, CHM⁺19] for descriptions of such compilers). In fact,

many of the most important properties of a SNARK, like proof size, verifier complexity, and cryptographic assumptions, follow primarily from the PCS. The earliest polynomial commitment schemes [KZG10] supported univariate polynomials and were used to construct SNARKs like Plonk [GWC19] and Marlin [CHM⁺19] with $O(n \log n)$ prover complexity and constant proof size. A different class of SNARKs [Set19, CBBZ22] arising from the sumcheck protocol [LFKN92] have linear prover time, but require *multilinear* Polynomial Commitment Schemes (ml-PCS’s).

1.1 Our results

Existing transformations from a univariate PCS to ml-PCS are either linear time but require a logarithmic opening proof size, like **gemini** [BCHO22] and **zeromorph** [KT23], or have constant size opening proofs but incur an additional $O(n \log n)$ prover cost to perform univariate polynomial multiplication via FFT’s. We propose a new protocol that goes beyond this tradeoff: **mercuri** has constant proof size and only $O(n)$ prover operations (in addition to the $O(\lambda n / (\log(\lambda n)))$ operations for multi-scalar multiplications arising in KZG commitments). It is also concretely more efficient than existing schemes with similar verifier complexity¹ in terms of the required scalar multiplications, as can be seen in table 1.

Table 1: Comparison of pairing-based ml-PCS. \mathbb{G} denotes a scalar multiplication. All verifiers below additionally require two pairings. Proof size is measured in elements of \mathbb{F} , and uses the fact that a \mathbb{G} -element is encoded by two \mathbb{F} -elements.

Scheme	Proof size	Prover Work	Verifier Work
univariate-based e.g.[PH23]	$O(1)$	$O(n \log n) \mathbb{F}, O(n) \mathbb{G}$	$O(\log n) \mathbb{F}, O(1) \mathbb{G}$
gemini [BCHO22]	$O(\log n)$	$O(n) \mathbb{F}, 3n \mathbb{G}$	$O(\log n) \mathbb{F}, O(\log n) \mathbb{G}$
zeromorph [KT23]	$O(\log n)$	$O(n) \mathbb{F}, 2.5n \mathbb{G}$	$O(\log n) \mathbb{F}, O(\log n) \mathbb{G}$
mercuri (this work)	$O(1)$	$O(n) \mathbb{F}, 2n + O(\sqrt{n}) \mathbb{G}$	$O(\log n) \mathbb{F}, O(1) \mathbb{G}$

Concurrent work Samaritan[GPS25] presents a similar construction.

2 Overview of technique

In this overview, we use some of the notation defined in Sections 3.1 and 3.2.

Our technique is best thought of as an improvement of the **gemini** ml-PCS [BCHO22]. Let’s start by recalling how **gemini** works. **gemini** commits to a multilinear function as

¹The ml-PCS from [KZHB25] requires only $O(\sqrt{n})$ scalar multiplications. However, it requires $O(\log n)$ proof length and $O(\log n)$ verifier pairings whereas all schemes in table 1 require only two. Moreover, committing to polynomials still requires $O(n)$ scalar multiplications and in most cases (with the exception of preprocessed polynomials) we need to do both in a SNARK proof.

a *univariate* KZG commitment [KZG10]. Specifically, fix a vector $f \in \mathbb{F}^n$ describing the function's values on the boolean cube \mathbf{B}_s where $s = \log n$. That is, we think of f as representing the multilinear

$$M(X_0, \dots, X_{s-1}) = \sum_{i < n} \mathbf{eq}(i, X_0, \dots, X_{s-1}) f_i.$$

(Here, as explained in Section 3.2, we interpret i as its binary decomposition (i_0, \dots, i_{s-1}) when used as input to \mathbf{eq} .) Let $\mathbf{srs} = \{[x^i]\}_{i < n}$ be a KZG structured reference string. **gemini** outputs $\mathbf{cm} = [f(x)] = \sum_{i < n} f_i [x^i]$ as a commitment to M .

Now suppose prover **P** wants to convince verifier **V** that $M(z) = v$, for some $z = (z_0, \dots, z_{s-1}) \in \mathbb{F}^s$. In **gemini**, **P** sends commitments $\mathbf{cm}_1, \dots, \mathbf{cm}_s$ to the s incremental restrictions leading to evaluation at z . Namely, to $M_1 = M(z_0, X_1, \dots, X_{s-1})$, $M_2 = M(z_0, z_1, X_2, \dots, X_{s-1})$, \dots , $M_s = M(z_0, \dots, z_{s-1})$. Assuming **P** sent commitments to the correct functions, all that is needed is to check that \mathbf{cm}_s is the commitment to the constant v . Of course, the interesting part is proving the commitments *are* to the correct functions!

For this purpose, **gemini** exploits a connection between M and its corresponding univariate $f(X)$: Write $f(X) = f_0(X^2) + X f_1(X^2)$, for $f_0(X), f_1(X)$ of degree $< n/2$. Let $f_{z_0}(X)$ be the univariate corresponding to M_1 defined above. Then, we have

$$f_{z_0}(X) = (1 - z_0)f_0(X) + z_0 f_1(X).$$

Additionally, we can evaluate f_0 and f_1 via f using the equations

$$f_0(X^2) = \frac{f(X) + f(-X)}{2}, f_1(X^2) = \frac{f(X) - f(-X)}{2X}$$

Thus, we can perform consistency checks between each pair $\mathbf{cm}_{i-1}, \mathbf{cm}_i$, via univariate KZG openings at a random challenge, inductively showing \mathbf{cm}_i is indeed the commitment to the next desired restriction. Of course, we get $O(s) = O(\log n)$ proof length due to this sequence of restriction commitments.

Here is a first idea on how to reduce proof length. Protocols based on univariate polynomials allow us to do multilinear evaluation in $O(n \log n)$ prover time with constant proof size (e.g. Section 5 of [PH23]). Choose a parameter t and set $b = 2^t$. We can run *only the first t rounds* of **gemini**, reaching a restricted multilinear on $n - t$ variables. If $n' = n/b \leq n/\log n$, we can afford to run a univariate protocol with $O(n' \log n') = O(n)$ prover time to evaluate $M_t(z_t, \dots, z_{s-1})$. This still doesn't take us to overall constant proof size - as we need to use a super-constant t to reach such n' . (For us $t = \log n/2$ will be optimal, although $t \geq \log \log n$ suffices here.)

This raises the question - can we "skip" the intermediate **gemini** rounds and send *only* the commitment \mathbf{cm}_t , and directly prove it is consistent with the original \mathbf{cm} ? Extrapolating the **gemini** strategy in the natural way, we get the answer - yes, but not with constant proof size: We can decompose f into b polynomials of degree $< n/b$: $f(X) = \sum_{0 \leq i < b} X^i f_i(X^b)$. As in the $b = 2$ case, one can show the univariate $f'(X)$

corresponding to M_t is a linear combination of the $\{f_i(X)\}$. Moreover, evaluating the f_i using f (for the consistency check) can be done. However, it requires b evaluations of f . Specifically, $f_i(r^b)$ is a linear combination of $\{f(r), f(r\omega), \dots, f(r\omega^{b-1})\}$ where ω is a primitive b 'th root of unity.

Our central innovation is a different way to prove cm_t is correct *with* constant proof size. Let's switch notation and denote the opening point as $u = (u_1, u_2)$ where $u_1 \in \mathbb{F}^t, u_2 \in \mathbb{F}^{s-t}$. The (univariate corresponding to the) correct restricted polynomial is

$$h(X) = \sum_{0 \leq i < b} \mathbf{eq}(i, u_1) f_i(X).$$

Let $g(X) := f(X) \bmod X^b - \alpha$. Calculation shows

$$g(X) = \sum_{0 \leq i < b} X^i f_i(\alpha).$$

The multilinear \hat{g} corresponding to $g(X)$ is

$$\hat{g}(X_0, \dots, X_{t-1}) = \sum_{0 \leq i < b} \mathbf{eq}(i, X_0, \dots, X_{t-1}) f_i(\alpha).$$

In particular, we have

$$\hat{g}(u_1) = \sum_{0 \leq i < b} \mathbf{eq}(i, u_1) f_i(\alpha) = h(\alpha).$$

In words, the evaluation of g at u_1 as a *multilinear* corresponds to the evaluation of h at α as a univariate! We can use standard univariate KZG to open cm_t at α . And, crucially, we can afford to evaluate $\hat{g}(u_1)$ using the aforementioned univariate protocols as it is of size b rather than n . In summary, we can show a committed polynomial corresponds to the correct restriction. And now, again, we can afford to open h as a multilinear at u_2 using univariate protocols as it has size n/b rather than n .

Remark 2.1. *MERCURY* bares resemblance to a bivariate sumcheck protocol [LFKN92]; however the correspondence is not perfect. We explore this in Appendix A.

3 Preliminaries

3.1 Terminology and conventions

Fields and Groups We assume our field \mathbb{F} is of prime order. We denote by $\mathbb{F}_{<d}[X]$ the set of univariate polynomials over \mathbb{F} of degree smaller than d . We assume all algorithms described receive as an implicit parameter the security parameter λ .

Whenever we use the term *efficient*, we mean an algorithm running in time $\text{poly}(\lambda)$. Furthermore, we assume an *object generator* \mathcal{O} that is run with input λ before all protocols, and returns all fields and groups used. Specifically, in our protocol $\mathcal{O}(\lambda) = (\mathbb{F}, \mathbb{G}, \mathbb{G}_2, \mathbb{G}_t, e, \mathcal{G}, \mathcal{G}_2, \mathcal{G}_t)$ where

- \mathbb{F} is a prime field of super-polynomial size $r = \lambda^{\omega(1)}$.
- $\mathbb{G}, \mathbb{G}_2, \mathbb{G}_t$ are all groups of size r , and e is an efficiently computable non-degenerate pairing $e : \mathbb{G} \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$.
- $\mathcal{g}, \mathcal{g}_2$ are uniformly chosen generators such that $e(\mathcal{g}, \mathcal{g}_2) = \mathcal{g}_t$.

We usually let the λ parameter be implicit, i.e. write \mathbb{F} instead of $\mathbb{F}(\lambda)$. We write \mathbb{G} and \mathbb{G}_2 additively. We use the notations $[x] := x \cdot \mathcal{g}$ and $[x]_2 := x \cdot \mathcal{g}_2$.

Vectors and polynomials We work with integer parameter n that we'll assume throughout the paper is of the form $n = 2^{2t}$ for integer $t > 0$. We'll denote its square root by $b := 2^t = \sqrt{n}$. We index vectors starting from zero. For example, for $g \in \mathbb{F}^b$ we have $g = (g_0, \dots, g_{b-1})$. We associate vectors with univariate polynomials in the following natural way: Given $g \in \mathbb{F}^b$ we denote $g(X) := \sum_{0 \leq i < b} g_i X^i$.

We make the convention that integer ranges in sums begin at zero if not specified otherwise. Thus, we write $g(X) = \sum_{i < b} g_i X^i$.

We assume vectors of size n are indexed by two indices ranging over $\{0, \dots, b-1\}$. It will be convenient to think, non-standardly, of the first index as the least significant digit. Thus, for $f \in \mathbb{F}^n$, we have $f = (f_{0,0}, \dots, f_{b-1,0}, \dots, f_{0,b-1}, \dots, f_{b-1,b-1})$. For $0 \leq i < b$, we denote by f_i the vector $(f_{i,0}, \dots, f_{i,b-1})$.

In particular, for $f \in \mathbb{F}^n$ we have under these notations that

$$f(X) := \sum_{i < b} X^i f_i(X^b) = \sum_{i < b} \sum_{j < b} f_{i,j} X^{i+j \cdot b}$$

For integer $m > 0$, we denote by \mathbf{B}_m the binary cube $\{0, 1\}^m \subset \mathbb{F}^m$ of dimension m .

3.2 Multilinear polynomials

Let $n = 2^{2t}$, and $s = 2t$. We define the well-known **eq** multilinear polynomial in $2s$ variables.

$$\mathbf{eq}(x, y) := \prod_{i=0}^{s-1} (x_i y_i + (1 - x_i)(1 - y_i))$$

We have for $x, y \in \mathbf{B}_s$, $\mathbf{eq}(x, y) = 1$ when $x = y$ and $\mathbf{eq}(x, y) = 0$ otherwise.

We use the convention that an integer $0 \leq i < n$ can be used as an input to **eq** by interpreting i as its binary representation. Namely, for $0 \leq i < n$, $u \in \mathbb{F}^s$, $\mathbf{eq}(i, u) := \mathbf{eq}(i_0, \dots, i_{s-1}, u)$ where $i = \sum_{j < s} i_j 2^j$.

For $f \in \mathbb{F}^n$, we define \hat{f} to be the multilinear polynomial obtaining f 's values on the boolean cube. Namely,

$$\hat{f}(X_0, \dots, X_{s-1}) := \sum_{i < n} \mathbf{eq}(i, X_0, \dots, X_{s-1}) \cdot f_i.$$

Decomposing \mathbf{eq} We'll overload \mathbf{eq} to also denote the analogous equality function for $x, y \in \mathbf{B}_t$. With this overloading, given $w_1, w_2, u_1, u_2 \in \mathbf{B}_t$ we have the convenient decomposition

$$\mathbf{eq}((w_1, w_2), (u_1, u_2)) = \mathbf{eq}(w_1, u_1) \mathbf{eq}(w_2, u_2).$$

3.3 The algebraic group model

We introduce some terminology from [GWC19] to capture analysis in the Algebraic Group Model of Fuchsbauer, Kiltz and Loss [FKL18]. In this subsection, we use the notation $\mathbb{G}_1 = \mathbb{G}$. In our protocols, by an *algebraic adversary* \mathcal{A} in an SRS-based protocol we mean a $\text{poly}(\lambda)$ -time algorithm which satisfies the following.

- For $i \in \{1, 2\}$, whenever \mathcal{A} outputs an element $A \in \mathbb{G}_i$, it also outputs a vector v over \mathbb{F} such that $A = \langle v, \mathbf{srs}_i \rangle$.

First we say our \mathbf{srs} has degree Q if all elements of \mathbf{srs}_i are of the form $[f(x)]_i$ for $f \in \mathbb{F}_{<Q+1}[X]$ and uniform $x \in \mathbb{F}$. In the following discussion let us assume we are executing a protocol with a degree Q SRS, and denote by $f_{i,j}$ the corresponding polynomial for the j 'th element of \mathbf{srs}_i .

Denote by a, b the vectors of \mathbb{F} -elements whose encodings in $\mathbb{G}_1, \mathbb{G}_2$ an algebraic adversary \mathcal{A} outputs during a protocol execution; e.g., the j 'th \mathbb{G}_1 element output by \mathcal{A} is $[a_j]$.

By a “real pairing check” we mean a check of the form

$$(a \cdot T_1) \cdot (T_2 \cdot b) = 0$$

for some matrices T_1, T_2 over \mathbb{F} . Note that such a check can indeed be done efficiently given the encoded elements and the pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$.

Given such a “real pairing check”, and the adversary \mathcal{A} and protocol execution during which the elements were output, define the corresponding “ideal check” as follows. Since \mathcal{A} is algebraic when he outputs $[a_j]_i$ he also outputs a vector v such that, from linearity, $a_j = \sum v_\ell f_{i,\ell}(x) = R_{i,j}(x)$ for $R_{i,j}(X) := \sum v_\ell f_{i,\ell}(X)$. Denote, for $i \in \{1, 2\}$ the vector of polynomials $R_i = (R_{i,j})_j$. The corresponding ideal check, checks as a polynomial identity whether

$$(R_1 \cdot T_1) \cdot (T_2 \cdot R_2) \equiv 0$$

The following lemma is inspired by [FKL18]'s analysis of [Gro16], and tells us that for soundness analysis against algebraic adversaries it suffices to look at ideal checks. Before stating the lemma we define the Q -DLOG assumption similarly to [FKL18].

Definition 3.1. Fix integer Q . The Q -DLOG assumption for $(\mathbb{G}_1, \mathbb{G}_2)$ states that given

$$[1], [x], \dots, [x^Q], [1]_2, [x]_2, \dots, [x^Q]_2$$

for uniformly chosen $x \in \mathbb{F}$, the probability of an efficient \mathcal{A} outputting x is $\text{negl}(\lambda)$.

Lemma 3.2. *Assume the Q -DLOG for $(\mathbb{G}_1, \mathbb{G}_2)$. Given an algebraic adversary \mathcal{A} participating in a protocol with a degree Q SRS, the probability of any real pairing check passing is larger by at most an additive $\text{negl}(\lambda)$ factor than the probability the corresponding ideal check holds.*

See [GWC19] for the proof.

3.4 Polynomial commitment schemes for multilinear polynomials

We give a formal definition of an ml-PCS secure in the algebraic group model.

Definition 3.3. *Let $n = 2^s$. A multilinear polynomial commitment scheme (ml-PCS) consists of*

- $\text{gen}(n)$ - a randomized algorithm that outputs an SRS srs .
- $\text{com}(f, \text{srs})$ - that given a polynomial $f \in \mathbb{F}^n$ returns a commitment cm to f .
- A public coin protocol $\text{open}(\text{cm}, n, u, v)$ between parties \mathbf{P} and \mathbf{V} . \mathbf{P} is given $f \in \mathbb{F}^n$. \mathbf{P} and \mathbf{V} are both given integer n , cm - the purported commitment to f , $u \in \mathbb{F}^s$ and $v \in \mathbb{F}$ - the purported value $\hat{f}(u)$.

such that

- **Completeness:** Suppose that $\text{cm} = \text{com}(f, \text{srs})$. Then if open is run correctly with values $n, \text{cm}, u, v = \hat{f}(u)$, \mathbf{V} outputs *accept* with probability one.
- **Knowledge soundness in the algebraic group model:** There exists an efficient \mathcal{E} such that for any efficient algebraic adversary \mathcal{A} the probability of \mathcal{A} winning the following game is $\text{negl}(\lambda)$ over the randomness of \mathcal{A} and gen .
 1. Given srs , \mathcal{A} outputs n, cm .
 2. \mathcal{E} , given access to the messages of \mathcal{A} during the previous step, outputs $f \in \mathbb{F}^n$.
 3. \mathcal{A} outputs $u \in \mathbb{F}^s$ and $v \in \mathbb{F}$.
 4. \mathcal{A} takes the part of \mathbf{P} in the protocol open with inputs n, cm, u, v .
 5. \mathcal{A} wins if
 - \mathbf{V} outputs *accept* at the end of the protocol.
 - $\hat{f}(u) \neq v$.

4 Components

In this section we go over known components (with some new optimizations), that will be used in our main protocol in Section 6. The treatment will be semi-formal, and assume basic familiarity with the KZG polynomial commitment scheme [KZG10]. The formal treatment will be part of the description and knowledge soundness proof of the main protocol in Section 6.

4.1 Inner products in $O(b \log b)$ time.

Fix polynomials $g_1(X) = \sum_{i=0}^{d_1} a_i X^i, g_2 = \sum_{i=0}^{d_2} b_i X^i$ in $\mathbb{F}[X]$. We define $\langle g_1, g_2 \rangle$ to be $\sum_{i=0}^d a_i b_i$ where $d := \min\{d_1, d_2\}$. We present a convenient way to verify inner products $\langle g_1, g_2 \rangle$ similar to [BCC⁺16, MBKM19]. The basic observation is that $\langle g_1, g_2 \rangle$ is the constant coefficient of the rational function $R(X) := g_1(X)g_2(1/X)$. Thus, $\langle g_1, g_2 \rangle = v$ is equivalent to the existence of polynomials $S_1(X), S_2(X)$ such that

$$g_1(X)g_2(1/X) = 1/X \cdot S_1(1/X) + v + X \cdot S_2(X).$$

We can thus send commitments to S_1, S_2 as proof of the correctness of v . As an optimization, we observe that we can “symmetrize” R and look instead at the rational function

$$R'(X) := g_1(X)g_2(1/X) + g_1(1/X)g_2(X).$$

The advantage of R' is that the negative and positive coefficients are equal. Thus, $\langle g_1, g_2 \rangle = v$ is equivalent to the existence of $S(X) \in \mathbb{F}[X]$ such that

$$g_1(X)g_2(1/X) + g_1(1/X)g_2(X) = 2v + X \cdot S(X) + (1/X)S(1/X).$$

Claim 4.1. *Suppose $g_1(X), g_2(X) \in \mathbb{F}_{<b}[X]$. Let $S(X)$ be as defined above. Then S can be computed in $O(b \log b)$ \mathbb{F} -operations.*

Proof. When $g_1(X), g_2(X) \in \mathbb{F}_{<b}[X]$ we multiply the equation above by X^{b-1} to get

$$X^{b-1}(g_1(X)g_2(1/X) + g_1(1/X)g_2(X)) = X^{b-1}(2v + X \cdot S(X) + (1/X)S(1/X)).$$

We can use an $O(b \log b)$ time FFT to evaluate the LHS on $2b$ points. We then do an inverse FFT to get the coefficients c_0, \dots, c_{2b-2} of the LHS. Now, we can output $S = (c_b, \dots, c_{2b-2})$. \square

Batching inner product checks Suppose we now have two inner product claims $\langle g_1, g_2 \rangle = v_1$ and $\langle h_1, h_2 \rangle = v_2$. The following claim gives us a way to randomly batch them so that one polynomial $S(X)$ suffices to prove both.

Claim 4.2. *Fix polynomials $g_1, g_2, h_1, h_2 \in \mathbb{F}[X]$ and $v_1, v_2 \in \mathbb{F}$. Suppose $\langle g_1, g_2 \rangle \neq v_1$ or $\langle h_1, h_2 \rangle \neq v_2$. Then, e.w.p $1/|\mathbb{F}|$ over $\gamma \in \mathbb{F}$ there does not exist $S(X) \in \mathbb{F}[X]$ such that*

$$\begin{aligned} g_1(X)g_2(1/X) + g_1(1/X)g_2(X) + \gamma(h_1(X)h_2(1/X) + h_1(1/X)h_2(X)) \\ = 2(v_1 + \gamma v_2) + X \cdot S(X) + (1/X)S(1/X) \end{aligned}$$

Proof. Denote $(v'_1, v'_2) = (\langle g_1, g_2 \rangle, \langle h_1, h_2 \rangle)$. The constant coefficient of the LHS is $2(v'_1 + \gamma v'_2)$. To satisfy the equation in the claim for some S , we need

$$v'_1 + \gamma v'_2 = v_1 + \gamma v_2,$$

which can hold for at most one γ when $(v_1, v_2) \neq (v'_1, v'_2)$. \square

4.2 Multilinear evaluations as inner products of univariate polynomials.

For $u \in \mathbb{F}^t$ define the polynomial $P_u(X) := \sum_{i < b} \mathbf{eq}(i, u) X^i$. Note that for $g(X) \in \mathbb{F}_{<b}[X]$, we have

$$\langle P_u, g \rangle = \sum_{i < b} \mathbf{eq}(i, u) g_i = \hat{g}(u).$$

As leveraged in [BGH19], we have the product formula

$$P_u(X) = \prod_{i=0}^{t-1} (u_i X^{2^i} + 1 - u_i),$$

implying $P_u(X)$ can be evaluated in $O(t)$ \mathbb{F} -operations. Hence, a verifier \mathbf{V} operating in $O(\log n)$ time can evaluate $P_u(X)$ itself.

Using the polynomials P_u , multilinear evaluations can be proven in a batched manner based on Claim 4.2: Suppose we want to show given committed univariates $g(X), h(X) \in \mathbb{F}[X]$, $u_1, u_2 \in \mathbb{F}^t$, $v_1, v_2 \in \mathbb{F}$ that $\hat{g}(u_1) = v_1$ and $\hat{h}(u_2) = v_2$.

1. \mathbf{V} sends random $\gamma \in \mathbb{F}$.
2. \mathbf{P} sends commitment to S such that

$$\begin{aligned} g(X)P_{u_1}(1/X) + g(1/X)P_{u_1}(X) + \gamma(h(X)P_{u_2}(1/X) + h(1/X)P_{u_2}(X)) \\ = 2(v_1 + \gamma v_2) + X \cdot S(X) + (1/X)S(1/X). \end{aligned}$$

3. \mathbf{V} chooses a random $\mathfrak{z} \in \mathbb{F}$.
4. \mathbf{P} sends and proves correctness of the values of g, h and S on $\mathfrak{z}, 1/\mathfrak{z}$.
5. \mathbf{V} evaluates P_{u_1}, P_{u_2} at $\mathfrak{z}, 1/\mathfrak{z}$.
6. \mathbf{V} checks the equation in step 2 holds at \mathfrak{z} .

4.3 Degree checks

The idea presented here is from [Tha23]. Suppose \mathbf{P} wants to prove to \mathbf{V} that \mathbf{cm} is a commitment to a polynomial $g(X) \in \mathbb{F}_{<b}[X]$. Let $D(X) := X^{b-1}g(1/X)$. The idea is that $D(X)$ is a polynomial if and only if $g(X)$ has degree $< b$. Thus, assuming our structured reference string doesn't contain negative powers, \mathbf{P} can commit to D if and only if $g(X) \in \mathbb{F}_{<b}[X]$.

This motivates the following protocol.

1. \mathbf{P} sends a commitment \mathbf{d} to $D(X)$.
2. \mathbf{V} chooses random $\mathfrak{z} \in \mathbb{F}$.
3. \mathbf{P} sends $D_{\mathfrak{z}} := D(\mathfrak{z}), \bar{g}_{\mathfrak{z}} := g(1/\mathfrak{z})$, and uses KZG to prove their correctness.
4. \mathbf{V} can now check D 's correctness on \mathfrak{z} , using the equation

$$D_{\mathfrak{z}} \stackrel{?}{=} \mathfrak{z}^{b-1} \bar{g}_{\mathfrak{z}}.$$

5 Univariate division

Our protocol crucially relies on the following simple claim about division by a polynomial of the form $X^b - \alpha$.

Claim 5.1. *Fix integer $b > 0$ and let $n = b^2$. Fix $\alpha \in \mathbb{F}$, and $f(X) \in \mathbb{F}_{<n}[X]$. Let $f_0(X), \dots, f_{b-1}(X) \in \mathbb{F}_{<b}[X]$ be such that $f(X) = \sum_{i<b} X^i f_i(X^b)$. Let $g(X) \in \mathbb{F}_{<b}[X], q(X) \in \mathbb{F}[X]$ be such that*

$$f(X) = (X^b - \alpha) \cdot q(X) + g(X).$$

Then,

1. $g(X) = \sum_{i<b} X^i f_i(\alpha)$.
2. The coefficients of $q(X)$ can be computed in $O(n)$ \mathbb{F} -operations.

Proof. To see the first item, note that reduction mod $X^b - \alpha$ corresponds to substituting α into X^b inside each $f_i(X^b)$ in the expression $\sum_{i<b} X^i f_i(X^b)$. We proceed to the computation of $q(X)$. We compute for each $0 \leq i < b$, the coefficients of the quotient $q_i(X) \in \mathbb{F}[X]$ such that

$$f_i(X) = q_i(X)(X - \alpha) + f_i(\alpha).$$

Using Horner's method for division by the linear polynomial $X - \alpha$ this requires only n multiplications and additions in \mathbb{F} . Now, we have that

$$f(X) = \sum_{i<b} X^i f_i(X^b) = \sum_{i<b} X^i \left(q_i(X^b)(X^b - \alpha) + f_i(\alpha) \right) = q(X)(X^b - \alpha) + g(X),$$

for $q(X) := \sum_{i<b} X^i q_i(X^b)$. Thus, the coefficients of $q(X)$ are simply the interleaving of the coefficients of the $\{q_i(X)\}$. \square

6 Main Construction

MERCURY is the tuple $(\text{gen}, \text{com}, \text{open})$ described next.

gen(n): Choose random $x \in \mathbb{F}$ and output $\{[1], [x], \dots, [x^{n-1}], [1]_2, [x]_2\}$

com(n, f, srs): Output $\sum_{i<b} \sum_{j<b} f_{i,j} \cdot [x^{i+j \cdot b}]$.

open($n, \text{cm}, u, v; f$):

1. Committing to restricted function:

- (a) Let $u = (u_1, u_2)$ for $u_1, u_2 \in \mathbb{F}^t$. \mathbf{P} computes the polynomial $h(X) := \sum_{i < b} \mathbf{eq}(i, u_1) f_i(X)$.
The coefficient of X^j in $h(X)$ is $\sum_{i < b} \mathbf{eq}(i, u_1) f_{i,j} = \hat{f}(u_1, j)$ when interpreting j as its binary representation on the RHS. Hence, we can think of h as a commitment to the restricted multilinear $\hat{f}(u_1, X_0, \dots, X_{t-1})$.
- (b) \mathbf{P} computes and sends $\mathbf{h} := [h(x)]$.
2. Committing to “folded” polynomial g :

- (a) \mathbf{V} sends random $\alpha \in \mathbb{F}$.
- (b) \mathbf{P} computes polynomials $g(X) \in \mathbb{F}_{<b}[X]$ and $q(X) \in \mathbb{F}[X]$ such that

$$f(X) = (X^b - \alpha) \cdot q(X) + g(X).$$

- (c) \mathbf{P} computes and sends $\mathbf{q} := [q(x)]$ and $\mathbf{g} := [g(x)]$.
3. Sending proofs of correctness for h and the degree of g :

- (a) \mathbf{V} sends a random batching challenge $\gamma \in \mathbb{F}$.
- (b) \mathbf{P} computes and sends $\mathbf{s} = [S(x)]$ where $S(X) \in \mathbb{F}[X]$ is such that

$$\begin{aligned} g(X)P_{u_1}(1/X) + g(1/X)P_{u_1}(X) + \gamma \cdot (h(X)P_{u_2}(1/X) + h(1/X)P_{u_2}(X)) \\ = 2(h(\alpha) + \gamma \cdot v) + X \cdot S(X) + (1/X)S(1/X). \end{aligned}$$

- (c) \mathbf{P} computes and sends $\mathbf{d} := [D(x)]$ where

$$D(X) := X^{b-1}g(1/X).$$

4. KZG evaluations:

- (a) \mathbf{V} sends a random evaluation challenge $\mathfrak{z} \in \mathbb{F}$.
- (b) \mathbf{P} sends the values $g_{\mathfrak{z}} := g(\mathfrak{z}), \bar{g}_{\mathfrak{z}} := g(1/\mathfrak{z}), h_{\mathfrak{z}} := h(\mathfrak{z}), \bar{h}_{\mathfrak{z}} := h(1/\mathfrak{z}), s_{\mathfrak{z}} := S(\mathfrak{z}), \bar{s}_{\mathfrak{z}} := S(1/\mathfrak{z})$.
- (c) \mathbf{V} computes the expected values for $D(\mathfrak{z})$ and $h(\alpha)$ assuming the equations in steps 3b, 3c are satisfied. That is, $D_{\mathfrak{z}} := \mathfrak{z}^{b-1}\bar{g}_{\mathfrak{z}}$, and

$$h_{\alpha} := (g_{\mathfrak{z}}P_{u_1}(1/\mathfrak{z}) + \bar{g}_{\mathfrak{z}}P_{u_1}(\mathfrak{z}) + \gamma(h_{\mathfrak{z}}P_{u_2}(1/\mathfrak{z}) + \bar{h}_{\mathfrak{z}}P_{u_2}(\mathfrak{z}) - 2v) - \mathfrak{z}s_{\mathfrak{z}} - (1/\mathfrak{z})\bar{s}_{\mathfrak{z}}) / 2.$$

- (d) \mathbf{P} computes and sends the KZG opening proof $\pi_{\mathfrak{z}}$ to check the equation of step 2b at \mathfrak{z} . That is $\pi_{\mathfrak{z}} := [H(x)]$ for

$$H(X) := \frac{f(X) - (\mathfrak{z}^b - \alpha)q(X) - g_{\mathfrak{z}}}{X - \mathfrak{z}}.$$

- (e) \mathbf{P} computes and sends a batched KZG opening proof π' for the values sent in step 4b and computed by \mathbf{V} in step 4c, as described in Section 4 of [BDFG20].
- (f) \mathbf{V} checks the proof π_3 via pairings as in [KZG10]:

$$e(\mathbf{cm} - [\mathfrak{z}^b - \alpha] \cdot \mathbf{q} - [g_3], [1]_2) = e(\pi_3, [x - \mathfrak{z}]_2).$$

- (g) \mathbf{V} checks the opening proof π' as described in [BDFG20].
- (h) If one of the checks in steps 4f,4g fails, \mathbf{V} outputs *reject*. Otherwise \mathbf{V} outputs *accept*.

Runtime of \mathbf{P} : Computing $q(X)$ in step 2b requires $O(n)$ operations by Claim 5.1. Computing \mathbf{q} and π_3 requires two MSMs of size n . All other steps are on polynomials of size $O(b) = O(\sqrt{n})$. Thus, other commitments clearly require $O(\sqrt{n})$ scalar multiplications. It is easy to see other steps require $o(n)$ \mathbb{F} -operations. The least trivial of these is perhaps the computation of $S(X)$ shown to require $O(b \log b) = o(n)$ operations in Claim 4.1.

Proving knowledge soundness: Let \mathcal{A} be an efficient algebraic adversary participating in the Knowledge Soundness game from Definition 3.3. We show its probability of winning the game is $\text{negl}(\lambda)$. We define the extractor \mathcal{E} to simply output the vector $f \in \mathbb{F}^n$ \mathcal{A} outputs (as it's algebraic) with $\text{com}(f) = \mathbf{cm}$ together with \mathbf{cm} .

As \mathcal{A} is algebraic, when sending the commitments $\mathbf{h}, \mathbf{q}, \mathbf{g}, \mathbf{s}, \mathbf{d}, \pi_3, \pi'$ during protocol execution it also sends polynomials $h(X), q(X), g(X), S(X), D(X), H(X), Q(X) \in \mathbb{F}_{<n}[X]$ such that the former are their corresponding commitments. Let E be the event that \mathbf{V} outputs *accept*. Let A be the event that \mathcal{A} wins the knowledge soundness game. Note that by definition $A \subset E$, and our goal is to show $\text{prob}(A) = \text{negl}(\lambda)$. We will define a constant number of events such that their union contains A and each has probability $\text{negl}(\lambda)$. This implies the knowledge soundness of the protocol.

E implies all pairing checks have passed. Let $E_0 \subset E$ be the event that one of the corresponding ideal pairing checks as defined in Section 3.3 didn't pass. According to Lemma 3.2, $\text{prob}(E_0) = \text{negl}(\lambda)$.

Given that E_0 didn't occur, we have from the knowledge soundness proof of batched KZG in Section 3 of [BDFG20] that the evaluations sent by \mathbf{P} and computed by \mathbf{V} are all correct. That is,

1. $g_3 = g(\mathfrak{z}), \bar{g}_3 = g(1/\mathfrak{z}), h_3 = h(\mathfrak{z}), \bar{h}_3 = h(1/\mathfrak{z}), s_3 = S(\mathfrak{z}), \bar{s}_3 = S(1/\mathfrak{z}),$
2. $D(\mathfrak{z}) = \mathfrak{z}^{b-1}g(1/\mathfrak{z}),$
3.
$$2(h(\alpha) + \gamma v) = g(\mathfrak{z})P_{u_1}(1/\mathfrak{z}) + g(1/\mathfrak{z})P_{u_1}(\mathfrak{z}) + \gamma(h(\mathfrak{z})P_{u_2}(1/\mathfrak{z}) + h(1/\mathfrak{z})P_{u_2}(\mathfrak{z})) - \mathfrak{z}S(\mathfrak{z}) - (1/\mathfrak{z})S(1/\mathfrak{z}).$$
4. $g(\mathfrak{z}) = f(\mathfrak{z}) - (\mathfrak{z}^b - \alpha)q(\mathfrak{z}).$

Note that items 2-4 can be viewed as rational equations evaluated at \mathfrak{z} . Let E_1 be the event that E_0 didn't occur, and one of the equations in steps 2-4 doesn't hold as a rational identity. Multiplying denominators, we have from the Schwarz-Zippel Lemma that E_1 occurs with probability at most $2n/|\mathbb{F}| = \text{negl}(\lambda)$ over $\mathfrak{z} \in \mathbb{F}$. Assuming E_0 and E_1 didn't occur we have that

1. $D(X) = X^{b-1}g(1/X),$
2.
$$2(h(\alpha) + \gamma v) = g(X)P_{u_1}(1/X) + g(1/X)P_{u_1}(X) + \gamma(h(X)P_{u_2}(1/X) + h(1/X)P_{u_2}(X)) - XS(X) - (1/X)S(1/X).$$
3. $g(X) = f(X) - (X^b - \alpha)q(X).$

Let E_2 be the event that E_0 and E_1 don't occur but $\hat{g}(u_1) \neq h(\alpha)$ or $\hat{h}(u_2) \neq v$. According to Claim 4.2, given the equation in item 2, E_2 occurs with probability at most $1/|\mathbb{F}|$ over γ .

Let E_3 be the event that $E_0 \cup E_1 \cup E_2$ don't occur and

1. $h(X) \neq \sum_{i < b} \mathbf{eq}(i, u_1) f_i(X),$
2. $h(\alpha) = \sum_{i < b} \mathbf{eq}(i, u_1) f_i(\alpha).$

Obviously, E_3 has probability $\text{negl}(\lambda)$.

Assume $E_0 \cup E_1 \cup E_2 \cup E_3$ doesn't occur. We show that $\hat{f}(u) = v$, and thus we are outside the event A . In other words, $A \subset E_0 \cup E_1 \cup E_2 \cup E_3$.

Since $D(X) = X^{b-1}g(1/X)$ we know that $\deg(g) < b$. Since $g(X) = f(X) - (X^b - \alpha)q(X)$, from Claim 5.1 we know that $g(X) = \sum_{i < b} f_i(\alpha)X^i$. Hence, we know that

$$\hat{g}(u_1) = \sum_{i < b} \mathbf{eq}(i, u_1) f_i(\alpha) = h(\alpha).$$

Using $\neg E_3$ we know that $h(X) = \sum_{i < b} \mathbf{eq}(i, u_1) f_i(X)$. Hence, writing $h(X) = \sum_{j < b} h_j X^j$ we have $h_j = \sum_{i < b} \mathbf{eq}(i, u_1) f_{i,j}$. Thus, we have

$$\begin{aligned} \hat{f}(u) &= \sum_{i < b} \sum_{j < b} \mathbf{eq}(i, u_1) \mathbf{eq}(j, u_2) f_{i,j} = \sum_{j < b} \mathbf{eq}(j, u_2) \sum_{i < b} \mathbf{eq}(i, u_1) f_{i,j} \\ &= \sum_{j < b} \mathbf{eq}(j, u_2) h_j = \hat{h}(u_2) = v. \end{aligned}$$

Acknowledgements

We thank Benedikt Bünz, Tohru Kohrita, Marek Sefranek and Justin Thaler for comments and corrections.

References

- [BCC⁺16] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. pages 327–357, 2016.
- [BCG⁺17] E. Ben-Sasson, A. Chiesa, A. Gabizon, M. Riabzev, and N. Spooner. Interactive oracle proofs with constant rate and query complexity. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 40:1–40:15, 2017.
- [BCHO22] J. Bootle, A. Chiesa, Y. Hu, and M. Orrù. Gemini: Elastic snarks for diverse environments. *IACR Cryptol. ePrint Arch.*, page 420, 2022.
- [BDFG20] D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Efficient polynomial commitment schemes for multiple points and polynomials. *IACR Cryptol. ePrint Arch.*, page 81, 2020.
- [BFS19] B. Bünz, B. Fisch, and A. Szepieniec. Transparent snarks from DARK compilers. *IACR Cryptol. ePrint Arch.*, page 1229, 2019.
- [BGH19] S. Bowe, J. Grigg, and D. Hopwood. Halo: Recursive proof composition without a trusted setup. *IACR Cryptol. ePrint Arch.*, page 1021, 2019.
- [CBBZ22] B. Chen, B. Bünz, D. Boneh, and Z. Zhang. Hyperplonk: Plonk with linear-time prover and high-degree custom gates. *IACR Cryptol. ePrint Arch.*, page 1355, 2022.
- [CHM⁺19] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zksnarks with universal and updatable SRS. *IACR Cryptology ePrint Archive*, 2019:1047, 2019.
- [DT24] Q. Dao and J. Thaler. More optimizations to sum-check proving. *Cryptology ePrint Archive*, Paper 2024/1210, 2024.
- [FKL18] G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 33–62, 2018.
- [GPS25] C. Ganesh, S. Patranabis, and N. Singh. Samaritan: Linear-time prover SNARK from new multilinear polynomial commitments. *Cryptology ePrint Archive*, Paper 2025/419, 2025.
- [Gro16] J. Groth. 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*, pages 305–326, 2016.

- [Gru24] Angus Gruen. Some improvements for the PIOP for ZeroCheck. *Cryptology ePrint Archive*, Paper 2024/108, 2024.
- [GWC19] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptology ePrint Archive*, 2019:953, 2019.
- [KT23] T. Kohrita and P. Towa. Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments. *IACR Cryptol. ePrint Arch.*, page 917, 2023.
- [KZG10] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. pages 177–194, 2010.
- [KZHB25] G. Kadianakis, A. Zapico, H. Hafezi, and B. Bünz. Kzh-fold: Accountable voting from sublinear accumulation. *IACR Cryptol. ePrint Arch.*, page 144, 2025.
- [LFKN92] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.
- [MBKM19] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. *IACR Cryptology ePrint Archive*, 2019:99, 2019.
- [PH23] S. Papini and U. Haböck. Improving logarithmic derivative lookups using GKR. *IACR Cryptol. ePrint Arch.*, page 1284, 2023.
- [Set19] S. T. V. Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. *IACR Cryptol. ePrint Arch.*, page 550, 2019.
- [Tha23] S. Thakur. A flexible snark via the monomial basis. *IACR Cryptol. ePrint Arch.*, page 788, 2023.
- [ZBK⁺22] A. Zapico, V. Buterin, D. Khovratovich, M. Maller, A. Nitulescu, and M. Simkin. Caulk: Lookup arguments in sublinear time. *IACR Cryptol. ePrint Arch.*, page 621, 2022.

A A sumcheck perspective on **MERCURY**

We sketch how one could attempt to construct a similar protocol by modifying the sumcheck protocol [LFKN92].

A multilinear evaluation can be written as a sum over the function’s values on \mathbf{B}_s multiplied by the **eq** function:

$$M(u) = \sum_{x \in \mathbf{B}_s} \mathbf{eq}(x, u) M(x).$$

The classic sumcheck protocol, like **gemi**, works by $\log n$ reductions of the domain size by a factor of two; each round fixing one more variable of the summed function. Let $b = \sqrt{n}$. We look instead at a modified sumcheck protocol on a bivariate function, where each variable ranges over a domain of size b . To maintain constant proof size and linear prover time, we need to resolve several issues.

1. The first univariate $s_1(X)$ has the form $s_1(X) = \sum_{i < b} P_{u_1}(X)P_{u_2}(i)F(X, i)$ (see Section 4.2 for definition of P_u). The standard way to compute s_1 is by first computing $P_{u_1}(X)P_{u_2}(i)F(X, i)$ for each $0 \leq i < b$, and summing the results. This requires a b -size *FFT* for each of the b coordinates giving superlinear time $O(n \log b)$ in total. However, as P_{u_1} depends only on X we can simply compute² the polynomial $\sum_{i < b} P_{u_2}(i)F(X, i)$, not requiring these FFTs, and multiply it in the end by $P_{u_1}(X)$, computing $s_1(X)$ in $O(n)$ time in total. In other words, when we are doing sumcheck over a product with a tensor vector like **eq**, computing s_1 can be done in linear time even when the domain is super-constant.
2. We can't send s_1 's coefficients as usually done in sumcheck while maintaining constant proof length. This can be resolved by sending a KZG commitment to s_1 instead of its coefficients.³
3. The sumcheck verifier **V** needs to check the sum of s_1 's values on the first variable's domain is v . **V** can't do this check directly from the commitment. This can be resolved by a univariate sumcheck subprotocol. A technical note is that **mercur** uses the polynomial h which actually corresponds to s_1/P_{u_1} . That is why an inner product subprotocol is needed (to take the inner product with P_{u_1}) rather than merely a univariate sumcheck.
4. Finally, $F(X, Y)$ as a bivariate needs to be opened at the sumcheck challenge (r_1, r_2) . Thus, we need a bivariate PCS. It turns out we have implicitly used a variant of the bivariate PCS from [GPS25] based on [ZBK⁺22]. Specifically, $g(\mathfrak{z})$ in our main protocol in Section 6 is the evaluation $F(\mathfrak{z}, \alpha)$; where $F(X, Y)$ is the bivariate polynomial such that our $f(X)$ is $F(X, X^b)$. Here, in fact, there is a missing technical component to make the sumcheck route give a similar result. We need a bivariate PCS where one can both commit and open the polynomial in linear time given its values, rather than its coefficients as the case in [ZBK⁺22, GPS25].

Is **mercur** a sumcheck? The main reason **mercur** is *not* precisely an instance of the sumcheck protocol is that it “mixes” the monomial and Lagrange base. In a sumcheck, we check s_1 's *values* over the domain sum to the expected final value, and then evaluate $s_1(X)$ at a random α to connect it to the next univariate s_2 . We are doing the first check over the monomial coefficients of $s_1(X)$ rather than its values (via

²This optimization for computing $s_1(X)$ originates from [Gru24]. See also [DT24] for a more in-depth study of optimizing sumchecks involving the **eq** function.

³A bivariate sumcheck over large variable domains was handled similarly in [BCG⁺17].

the inner product argument between h and P_{u_1}); but still evaluate $h(X)$ to connect it to $g(X)$ which roughly corresponds to $s_2(X)$ in a sumcheck.