On the Equivalence of Polynomial Commitments for an Identical Polynomial under Different Bases

Dengji Ma[a], Jingyu Ke[b], Sinka Gao[a], Guoqiang Li[b,*]

[a] *Delphinus Lab, Sydney, Australia*
[b] *Shanghai Jiao Tong University, China*

Abstract

We propose a Pairing-based Polynomial Consistency Protocol (PPCP) that verifies the equivalence of polynomial commitments generated under different basis representations, such as the coefficient and Lagrange bases. By leveraging pairing relations, PPCP proves that two commitments correspond to an identical underlying polynomial vector without revealing the polynomial itself. This enables efficient proof aggregation and recursive composition across heterogeneous SNARK systems that adopt distinct polynomial encodings.

Keywords:
Zero-knowledge proofs, Polynomial commitment schemes, Proof aggregation, Cross-system interoperability

1. Introduction

Zero-knowledge proof (ZKP) [1] systems are cryptographic protocols that allow a prover to convince a verifier of a statement's validity without revealing any information beyond its truth. Polynomial commitments [2, 3] play a central role in such systems by enabling the prover to commit to a polynomial and later prove evaluations at specific points efficiently and securely. These commitments can be represented under different bases, most notably the Coefficient and Lagrange forms, which are mathematically equivalent and can be transformed into each other.

Different ZKP frameworks adopt different representations. Plonkish prover systems such as *Plonk* [4] and *Halo2* [5] use the Lagrange basis, converting it to the coefficient basis through an inverse Fast Fourier Transform (IFFT) to generate KZG proofs. Multilinear prover systems such as *Gemini* [6], *Zeromorph* [7] and *GKR* [8], on the other hand, employ the sumcheck protocol to operate directly in the coefficient domain, eliminating FFT overhead and reducing proof generation complexity from $O(n \log n)$ to $O(n)$. Despite their efficiency, these systems use distinct polynomial encodings, which makes interoperability and proof aggregation across them non-trivial.

While Plonkish provers can achieve high performance through GPU parallelism, this often leads to considerable computational and deployment costs. Multilinear provers offer a more cost-effective alternative with comparable proving speed but lower hardware

*Corresponding author.
Email address: li.g@sjtu.edu.cn (Guoqiang Li)

requirements. In large zkVM architectures, computationally heavy components—such as cryptographic hash circuits—are often separated from the main proving pipeline. Proofs for these auxiliary modules can be generated using a multilinear prover and then aggregated with the main Plonkish proof. To preserve soundness during this aggregation, it is necessary to verify that the polynomial vectors produced by both provers correspond to the same underlying polynomial, even though they are expressed under different bases.

To address this challenge, we propose a lightweight *Pairing-based Polynomial Consistency Protocol (PPCP)* that verifies the equivalence of two commitments generated under distinct bases without revealing the polynomial itself. By leveraging pairing relations, PPCP enables efficient consistency checking between heterogeneous proof systems and can be seamlessly integrated with existing KZG pairing verification.

Two variants of PPCP are presented. The first guarantees consistency without relying on external assumptions but requires three pairing checks. The second achieves the same goal with only two pairing checks by introducing minimal external assumptions. Both variants are compatible with current KZG verification, with the latter achieving full integration without adding any extra verification cost.

This work contributes a practical solution to cross-system proof aggregation by ensuring algebraic consistency between commitments in heterogeneous ZKP frameworks. PPCP thus provides an essential foundation for scalable, interoperable, and recursively composable proof systems.

2. Technical Background

2.1. Plonkish Prover System

A Plonkish prover system, as exemplified by *Plonk* and *Halo2*, represents an arithmetic circuit $C(x, w) = 0$ over a finite field $\mathbb{F}_q$ using a set of low-degree polynomials defined on a multiplicative subgroup $H = \{\omega^0, \omega^1, \ldots, \omega^{n-1}\} \subset \mathbb{F}_q$, where ω is a primitive n-th root of unity. The prover encodes wire values into evaluation polynomials $a(X), b(X), c(X) \in \mathbb{F}_q[X]$ such that, for each $X \in H$, the circuit's gate constraint polynomial

$$P_{\text{gate}}(X) = q_L(X)a(X) + q_R(X)b(X) + q_M(X)a(X)b(X) + q_O(X)c(X) + q_C(X)$$

vanishes over the evaluation domain. Here, the selector polynomials q_L, q_R, q_M, q_O, q_C capture the circuit's local wiring and arithmetic relations.

To ensure the correct wiring between different gates, Plonkish systems introduce a grand-product polynomial $z(X)$ that enforces permutation consistency among wire values. The permutation relation is captured by a polynomial $P_{\text{perm}}(X)$, which algebraically links (a, b, c) to the corresponding permutation mapping. All constraints are unified in a single quotient polynomial

$$t(X) = \frac{P_{\text{gate}}(X) + \alpha\, P_{\text{perm}}(X)}{Z_H(X)},$$

where $Z_H(X) = X^n - 1$ is the vanishing polynomial over H, and α is a random challenge binding the gate and permutation constraints.

The prover commits to all polynomials using a polynomial commitment scheme, typically the KZG PCS, and provides opening proofs at a challenge point $\zeta \notin H$. Because Plonkish systems represent polynomials in the *Lagrange basis* over H, commitment and quotient construction require conversion to the *coefficient basis* via an inverse Fast Fourier Transform (IFFT). This FFT-based structure enables efficient composition and batching but incurs a prover complexity of $O(n \log n)$, where n is the circuit size. The resulting proof achieves succinct verification with a constant number of pairings under standard cryptographic assumptions.

2.2. Multilinear Prover System

In contrast, a multilinear prover system, as used in *Gemini* and *Zeromorph*, represents the witness as a multilinear extension polynomial $f_w : \mathbb{F}_q^{\log N} \to \mathbb{F}_q$ defined by

$$f_w(X_1, \ldots, X_{\log N}) = \sum_{i \in \{0,1\}^{\log N}} w_i \prod_{j=1}^{\log N} X_j^{i_j} (1 - X_j)^{1 - i_j},$$

where each w_i corresponds to a component of the witness vector w. This multilinear encoding ensures that $f_w(\mathbf{i}) = w_i$ for any Boolean index $\mathbf{i} \in \{0,1\}^{\log N}$, allowing the circuit constraints to be expressed as evaluations of multilinear functions over the Boolean hypercube.

The system employs the sumcheck protocol [9] to verify relations of the form

$$\sum_{\mathbf{b} \in \{0,1\}^v} g(\mathbf{b}) = s,$$

where g is a low-degree polynomial derived from circuit constraints and s is the claimed result. Through a sequence of verifier challenges, this multi-variable equation is reduced to the evaluation of a single univariate polynomial, which the prover can efficiently open using the underlying commitment scheme.

Because all polynomial operations are conducted directly in the multilinear domain, this framework eliminates the need for FFT-based transformations. As a result, the prover's computational complexity is reduced to $O(n)$, while maintaining succinct verification through inner-product and commitment checks. The multilinear paradigm thus provides a lightweight and parallelizable proving architecture that complements Plonkish systems, making it particularly suitable for subcircuits or plugin components in zkVM frameworks.

3. Protocol Design

Definition 3.1. Let λ be the security parameter. An *object generator* $\mathcal{O}$ is executed once on input λ prior to all protocols. The generator outputs the tuple

$$\mathcal{O}(\lambda) = (\mathbb{F}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, g_T),$$

which specifies the algebraic setting of the system, with the following requirements:

- $\mathbb{F}$ is a finite field of prime order r, where $\log r = \Theta(\lambda)$;
- $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ are cyclic groups of order r;

- $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_{\mathbb{T}}$ is an efficiently computable, non-degenerate bilinear pairing;
- $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are uniformly sampled generators such that $e(g_1, g_2) = g_T$.

For any $x \in \mathbb{F}$, the notations $[x]_1 := x \cdot g_1$ and $[x]_2 := x \cdot g_2$ denote the canonical encodings of x in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

Definition 3.2. A polynomial commitment scheme (PCS) consists of three algorithms $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ defined as follows:

- $\mathsf{Gen}(d)$ is a randomized algorithm that, given a positive integer d, outputs a structured reference string (SRS) denoted by srs.
- $\mathsf{Com}(f, \mathsf{srs})$ is an algorithm that, given a polynomial $f \in \mathbb{F}_{<d}[X]$ and an SRS srs, returns a commitment cm to the polynomial f.
- Open is a public-coin interactive protocol between a prover P and a verifier V. Given polynomials $f_1, \ldots, f_k \in \mathbb{F}_{<d}[X]$, the prover demonstrates correctness of the evaluations of these polynomials at designated points, without revealing the polynomials themselves.

3.1. Pairing-based Polynomial Consistency Protocol (PPCP)

This section presents a protocol that certifies whether two polynomial commitments, expressed under different bases, correspond to the same underlying polynomial vector.

The protocol leverages the bilinearity of pairings to enforce cross-basis consistency while preserving the hiding property of the committed polynomial.

Protocol 3.1 (Pairing-based Polynomial Consistency Protocol)**.** Let n denote the size of the evaluation domain, equivalently the length of the polynomial vector being committed. The protocol consists of the following stages:

Setup: Given n, the algorithm $\mathsf{Gen}(n)$ samples a secret value $x \in \mathbb{F}$ and constructs the following structured reference strings.

- **Coefficient-basis SRS.** The monomial basis is encoded using the secret evaluation point x:

$$\mathsf{srs}_{\mathsf{coef}} = \big([1]_1, [x]_1, \ldots, [x^{n-1}]_1,\ [1]_2, [x]_2\big),$$

  where $c_i := x^i$ for all i.
- **Lagrange-basis SRS.** Using a primitive n-th root of unity w, the Lagrange basis polynomials are defined as

$$\ell_i(X) = \frac{w^i(X^n - 1)}{n(X - w^i)},$$

  which yield the SRS (at the evaluation point x):

$$\mathsf{srs}_{\mathsf{lag}} = \big([\ell_0]_1, [\ell_1]_1, \ldots, [\ell_{n-1}]_1\big).$$

- **Cross-term SRS.** For a secret scalar $r \in \mathbb{F}$ (which can be set to $r = x$), define
$$t_i = r^{-1} c_i \sum_{j \neq i} \frac{\ell_j(x)}{c_j},$$
and include
$$\mathsf{srs}_{\mathsf{cross}} = \big([t_0]_1, [t_1]_1, \ldots, [t_{n-1}]_1\big).$$
- **Auxiliary G_2-element.** An additional group element is defined as
$$M_2 := \left[\sum_{i=0}^{n-1} \frac{\ell_i(x)}{c_i}\right]_2.$$

Commit: The prover $\mathcal{P}$ holds a secret polynomial vector $a = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{F}^n$ and computes the commitments
$$X = \sum_{i=0}^{n-1} a_i [c_i]_1, \qquad Y = \sum_{i=0}^{n-1} a_i [\ell_i]_1, \qquad Z = \sum_{i=0}^{n-1} a_i [t_i]_1.$$
The prover sends (X, Y, Z) to the verifier.

Verify: The verifier checks the pairing equation
$$e(X, M_2) \stackrel{?}{=} e(Y, [1]_2) \cdot e(Z, [x]_2)$$
and accepts if and only if the equality holds.

Theorem 3.1 (Cross-basis Consistency)**.** *Let the SRS be generated as in the **Setup** phase of the PPCP protocol, yielding $\{[c_i]_1, [\ell_i]_1, [t_i]_1\}_{i=0}^{n-1}$ and M_2. Let an algebraic adversary, given this SRS, output group elements $X, Y, Z \in G_1$ that satisfy the verifier's pairing check*
$$e(X, M_2) = e(Y, [1]_2) \cdot e(Z, [r]_2).$$
Then there exists a (unique) vector $a = (a_0, \ldots, a_{n-1}) \in \mathbb{F}^n$ such that
$$X = \sum_{i=0}^{n-1} a_i [c_i]_1, \qquad Y = \sum_{i=0}^{n-1} a_i [\ell_i]_1, \qquad Z = \sum_{i=0}^{n-1} a_i [t_i]_1.$$
In particular, X, Y, Z are consistent commitments to the same polynomial vector under the coefficient, Lagrange, and cross-term bases, respectively.

Proof. Consider the pairing equation
$$e(X, M_2) = e(Y, [1]_2) \cdot e(Z, [r]_2),$$
where X, Y, Z commit to the same polynomial vector. Fix an index j and isolate the j-th components $X_j = a_j [c_j]_1$, $Y_j = a_j [\ell_j]_1$, $Z_j = a_j [t_j]_1$. The corresponding pairing instance is
$$e\left(X_j, \left[\sum_{i=0}^{n-1} \tfrac{\ell_i}{c_i}\right]_2\right) = e(Y_j, [1]_2) \cdot e(Z_j, [r]_2). \tag{1}$$

As an algebraic adversary is assumed to be able to construct arbitrary linear combinations of the available SRS elements, its most general form for the j-th components can be written as

$$X_j = (x_0[c_j]_1 + x_1[\ell_j]_1 + x_2[t_j]_1), \quad Y_j = (y_0[c_j]_1 + y_1[\ell_j]_1 + y_2[t_j]_1), \quad Z_j = (z_0[c_j]_1 + z_1[\ell_j]_1 + z_2[t_j]_1).$$

Using $e(ag_1, bg_2) = e(g_1, g_2)^{ab}$, the exponent equality implied by (1) becomes

$$(x_0 c_j + x_1 \ell_j + x_2 t_j) \cdot \sum_i \tfrac{\ell_i}{c_i} = (y_0 c_j + y_1 \ell_j + y_2 t_j) + (z_0 c_j + z_1 \ell_j + z_2 t_j) r.$$

Expanding the left-hand side gives

$$x_0 \ell_j + x_0 c_j \sum_{i \neq j} \tfrac{\ell_i}{c_i} + x_1 \ell_j \sum_i \tfrac{\ell_i}{c_i} + x_2 t_j \sum_i \tfrac{\ell_i}{c_i},$$

which contains the terms

$$\ell_j \sum_{i \neq j} \tfrac{\ell_i}{c_i}, \qquad t_j \sum_{i \neq j} \tfrac{\ell_i}{c_i},$$

that do not appear on the right-hand side (recall that $c_j \sum_{i \neq j} \frac{\ell_i}{c_i} = r \cdot t_j$).

Expanding the right-hand side gives

$$(y_0 + z_0 r) c_j + (y_1 + z_1 r) \ell_j + (y_2 + z_2 r) t_j,$$

which contains the term c_j that does not appear on the left-hand side.

Since these terms cannot be matched, their coefficients must be zero:

$$x_1 = x_2 = y_0 = z_0 = 0.$$

Now the equality reduces to

$$x_0 \ell_j + x_0 c_j \sum_{i \neq j} \tfrac{\ell_i}{c_i} = y_1 \ell_j + z_1 r \ell_j + y_2 t_j + z_2 r t_j.$$

Substituting $t_j = r^{-1} c_j \sum_{i \neq j} \frac{\ell_i}{c_i}$ gives

$$y_1 \ell_j + z_1 r \ell_j + y_2 r^{-1} c_j \sum_{i \neq j} \tfrac{\ell_i}{c_i} + z_2 c_j \sum_{i \neq j} \tfrac{\ell_i}{c_i}.$$

Since the left-hand side contains no terms of the form $r\ell_j$ or $r^{-1} c_j \sum_{i \neq j} \frac{\ell_i}{c_i}$, we must have $z_1 = y_2 = 0$. The equality then becomes

$$x_0 \ell_j + x_0 c_j \sum_{i \neq j} \tfrac{\ell_i}{c_i} = y_1 \ell_j + z_2 c_j \sum_{i \neq j} \tfrac{\ell_i}{c_i},$$

from which

$$x_0 = y_1 = z_2.$$

Thus the coefficients of X_j, Y_j, Z_j coincide, i.e., $X_j = Y_j = Z_j$, which corresponds exactly to a consistent commitment to the same polynomial coefficient a_j. Hence the pairing equation enforces cross-basis consistency. □

3.2. Circuit impact

Given a polynomial $f \in \mathbb{F}[X]$ satisfies $f(z) = v$ if and only if there exists a polynomial $q \in \mathbb{F}[X]$ such that $f - v = (X - z)q$, i.e., if and only if $X - z$ divides $f - v$ in $\mathbb{F}[X]$. Committing to the polynomial f and q yields the commitment C and proof Π. Denote $W := C - [v]_1 + z \cdot \Pi$. The standard pairing check requires two pairings[KZG]:

$$e(W, -[1]_2) \cdot e(\Pi, [x]_2) = 1$$

Denote C_{coeff},C_{lag} and C_{cross} as coefficient basis, Lagrange basis and cross term commitments, respectively. With a random challenge u, this PPCP protocol can be folded into this check, adding only one additional pairing:

$$e(W + u \cdot C_{lag}, -[1]_2) \cdot e(\Pi - u \cdot C_{cross}, [x]_2) \cdot e(u \cdot C_{coeff}, M_2) = 1$$

3.3. Commitment-Sum Polynomial Consistency Protocol (CSPCP)

This protocol establishes a lighter consistency check obtained by combining commitments through their sum, while relying on the same SRS structure as in the PPCP protocol. As before, let $\{c_i\}, \{\ell_i\}$ be the coefficient and Lagrange basis, respectively. In addition, a new cross-term basis is introduced:

$$s_i := r(c_i + \ell_i),$$

where r is the secret scalar.

Given a polynomial vector $a = (a_0, \ldots, a_{n-1})$, the prover forms the coefficient-basis commitment X, the Lagrange-basis commitment Y, and the cross-term commitment Z as

$$X = \sum_i a_i[c_i]_1, \qquad Y = \sum_i a_i[\ell_i]_1, \qquad Z = \sum_i a_i[s_i]_1.$$

The verification equation of CSPCP differs from PPCP and is given by

$$e(X + Y, [r]_2) = e(Z, [1]_2).$$

We study under what conditions this equation enforces that X, Y, Z are consistent commitments to the same polynomial vector. In contrast to PPCP, CSPCP requires the additional assumption that X is expressed in the coefficient basis and Y in the Lagrange basis. Without this restriction, the equation admits many spurious solutions.

Algebraic structure of the pairing equation. Fix an index j. The corresponding SRS elements are the field-independent group elements $c_j, \ell_j, s_j = r(c_j + \ell_j)$.

In the algebraic adversary model, the most general forms of the j-th coordinates of X, Y, Z are written as

$$X_j = x_0[c_j]_1 + x_1[\ell_j]_1 + x_2[s_j]_1, \qquad Y_j = y_0[c_j]_1 + y_1[\ell_j]_1 + y_2[s_j]_1, \qquad Z_j = z_0[c_j]_1 + z_1[\ell_j]_1 + z_2[s_j]_1$$

where all $x_i, y_i, z_i \in \mathbb{F}$ are scalars.

From

$$e(X + Y, [r]_2) = e(Z, [1]_2),$$

the exponent relation for this fixed coordinate becomes

$$r\big((x_0 + y_0)c_j + (x_1 + y_1)\ell_j + (x_2 + y_2)s_j\big) = z_0 c_j + z_1 \ell_j + z_2 s_j.$$

Substituting $s_j = r(c_j + \ell_j)$ gives

$$\begin{aligned}
0 &= (r(x_0 + y_0) - z_0)c_j + (r(x_1 + y_1) - z_1)\ell_j + (r(x_2 + y_2) - z_2)s_j \\
&= (r(x_0 + y_0) - z_0)c_j + (r(x_1 + y_1) - z_1)\ell_j + (r(x_2 + y_2) - z_2)\, r(c_j + \ell_j) \\
&= \big(r(x_0 + y_0) + r^2(x_2 + y_2) - z_0 - rz_2\big)c_j \\
&\quad + \big(r(x_1 + y_1) + r^2(x_2 + y_2) - z_1 - rz_2\big)\ell_j.
\end{aligned}$$

Since c_j and ℓ_j are linearly independent group elements, both coefficients vanish:

$$\begin{cases} r(x_0 + y_0) + r^2(x_2 + y_2) = z_0 + rz_2, \\ r(x_1 + y_1) + r^2(x_2 + y_2) = z_1 + rz_2. \end{cases} \tag{$\star$}$$

Because r is sampled uniformly at random in the setup, these equalities must hold as polynomial identities in r. Matching coefficients gives

$$x_2 + y_2 = 0, \qquad x_0 + y_0 = z_2, \qquad x_1 + y_1 = z_2, \qquad z_0 = z_1 = 0.$$

Theorem 3.2 (Strong Cross-basis Consistency)**.** *In CSPCP, suppose the commitments are required to use fixed bases: X is expressed in the coefficient basis and Y in the Lagrange basis.*

If a triple (X, Y, Z) satisfies the verifier equation

$$e(X + Y, [r]_2) = e(Z, [1]_2),$$

then the pairing constraint admits a unique compatible assignment of coefficients in which the three commitments share the same scalar. Consequently, each coordinate satisfies

$$X_j = a_j[c_j]_1, \qquad Y_j = a_j[\ell_j]_1, \qquad Z_j = a_j[s_j]_1,$$

for a common value a_j, and therefore the commitments are consistent with the same underlying polynomial vector.

Proof. Under the basis constraints $x_1 = x_2 = 0$ and $y_0 = y_2 = 0$, the relations

$$x_2 + y_2 = 0, \qquad x_0 + y_0 = z_2, \qquad x_1 + y_1 = z_2, \qquad z_0 = z_1 = 0$$

reduce to

$$x_0 = z_2, \qquad y_1 = z_2.$$

Thus each coordinate satisfies

$$X_j = z_2[c_j]_1, \qquad Y_j = z_2[\ell_j]_1, \qquad Z_j = z_2[s_j]_1,$$

which is the unique form of consistent commitments to the same coefficient $a_j = z_2$. Since j was arbitrary, the conclusion holds component-wise, hence for the entire polynomial vector. □

Remarks. Unlike PPCP, CSPCP does not enforce cross-basis consistency unless the bases of X and Y are fixed in advance. This assumption naturally holds in settings such as aggregation circuits, where each commitment is already required to appear in a predetermined basis (e.g., Lagrange for proof-system A, coefficient for proof-system B). Under this restriction, CSPCP provides a lightweight consistency guarantee with minimal pairing cost.

Protocol	SRS size	Prover work	Proof size[3]	Aggregate Verifier cost
Halo2	$2nG_1,\ 2G_2$	nG_1	$1G_1$	$2P$
Protocol 1 (Lag.)	$2nG_1,\ 2G_2$	nG_1	$1G_1$	–
Protocol 1 (Coeff.)	$2nG_1^{(1)},\ 3G_2$	$2nG_1$	$2G_1$	$3P^{(2)}$
Protocol 2 (Lag.)	$2nG_1,\ 2G_2$	nG_1	$1G_1$	–
Protocol 2 (Coeff.)	$2nG_1,\ 2G_2$	$2nG_1$	$2G_1$	$2P$

Table 1: Comparison with the Plonkish system Halo2. G_i denotes scalar multiplication in G_i, and P denotes a pairing. "Lag." and "Coeff." refer to the Lagrange-basis and coefficient-basis variants, respectively. (1) Coefficient-basis protocols require only coefficient and cross-term SRS elements, whereas Lagrange-basis protocols use coefficient and Lagrange SRS elements. (2) The aggregate verifier additionally supports combining commitments drawn from both bases. (3) The reported proof size refers only to the cross-basis equivalence check inside the aggregated verifier.

3.4. Circuit Impact

Denote C_{coeff},C_{lag} and C_{sum} as coefficient basis, Lagrange basis and cross term commitments, respectively. Using a random challenge u, the protocol can be folded into standard KZG pairing check without additional pairing operations are introduced:

$$e(\Pi + u \cdot (C_{\text{coeff}} + C_{\text{lag}}), [x]_2) \cdot e(W + u \cdot C_{\text{sum}}, -[1]_2) = 1.$$

4. Related Work

There exist several approaches to verifying the equivalence between two polynomial commitments, yet most of them are computationally impractical.

1. **Transformation-based verification.** In a multilinear setting, one may first represent the polynomial vector in the Lagrange basis, commit to it, and then transform it into the coefficient basis using the Fast Fourier Transform (FFT) to generate a KZG proof. However, this approach incurs significant computational overhead and is less efficient compared to our proposed mapping scheme.
2. **Component-wise pairing verification.** Another method is to verify each component individually through pairing checks, i.e.,

$$e\Big([a_i c_i]_1, \Big[\tfrac{l_i}{c_i}\Big]_2\Big) \;=\; e([a_i l_i]_1, [1]_2)\,.$$

   Nevertheless, when the vector size grows to $n = 2^{18\ldots22}$, the required number of pairings renders this approach computationally prohibitive.
3. **Circuit-based equivalence checking.** Alternatively, one can encode all components into a zero-knowledge circuit and verify their consistency within the circuit. Yet, for $n = 2^{18\ldots22}$, the circuit size required to perform pairing-based checks remains extremely large, making this solution inefficient in practice.

5. Conclusion

In this paper, we propose a new approach that achieves efficient equivalence verification between polynomial commitments generated under different bases. During the setup phase, the protocol precomputes the cross terms along with both coefficient and

Lagrange forms as part of the structured reference string (SRS). The commitment is then produced in the coefficient form, allowing the verifier to check consistency through an additional pairing operation or to incorporate the verification directly into the existing KZG pairing check.

As summarized in Table 1, the proposed scheme introduces only minimal overhead compared to the Plonkish Halo2 baseline. In particular, when used as a standalone module, The consistency proof incurs only one additional pairing, which is comparable to the prover cost of a 150K-row Plonkish circuit (approximately 3.5% of a 2^{22}-row circuit).Moreover, it aggregates all equivalent commitments into a single one through random challenges. When integrated with existing KZG verification, it incurs only two extra elliptic-curve point additions, effectively preserving the verification cost while extending compatibility across heterogeneous proof systems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof systems, in: STOC, 1985, pp. 291–304.

[2] A. Kate, G. M. Zaverucha, I. Goldberg, Constant-size commitments to polynomials and their applications, in: International conference on the theory and application of cryptology and information security, Springer, 2010, pp. 177–194.

[3] C. Papamanthou, E. Shi, R. Tamassia, Signatures of correct computation, in: Theory of Cryptography Conference, Springer, 2013, pp. 222–242.

[4] A. Gabizon, Z. J. Williamson, O. Ciobotaru, Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge, Cryptology ePrint Archive (2019).

[5] Z. Foundation, E. C. Company, Halo2: The Halo 2 zero-knowledge proving system, https://github.com/zcash/halo2 (2020).

[6] J. Bootle, A. Chiesa, Y. Hu, M. Orru, Gemini: Elastic snarks for diverse environments, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2022, pp. 427–457.

[7] T. Kohrita, P. Towa, Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments: T. kohrita, p. towa, Journal of Cryptology 37 (4) (2024) 38.

[8] S. Goldwasser, Y. T. Kalai, G. N. Rothblum, Delegating computation: interactive proofs for muggles, Journal of the ACM (JACM) 62 (4) (2015) 1–64.

[9] C. Lund, L. Fortnow, H. Karloff, N. Nisan, Algebraic methods for interactive proof systems, Journal of the ACM (JACM) 39 (4) (1992) 859–868.