Hadamard Product Argument from Lagrange-Based Univariate Polynomials

Jie Xie
xiejie1006@sjtu.edu.cn
Shanghai Jiao Tong University

Yuncong Hu
huyuncong@sjtu.edu.cn
Shanghai Jiao Tong University

Yu Yu
yyuu@sjtu.edu.cn
Shanghai Jiao Tong University

Abstract

Hadamard product is a point-wise product for two vectors. This paper presents a new scheme to prove Hadamard-product relation as a sub-protocol for SNARKs based on univariate polynomials. Prover uses linear cryptographic operations to generate the proof containing logarithmic field elements. The verification takes logarithmic cryptographic operations with constant numbers of pairings in bilinear group. The construction of the scheme is based on the Lagrange-based KZG commitments (Kate, Zaverucha, and Goldberg at Asiacrypt 2010) and the folding technique. We construct an inner-product protocol from folding technique on univariate polynomials in Lagrange form, and by carefully choosing the random polynomials suitable for folding technique, we construct a Hadamard-product protocol from the inner-product protocol, giving an alternative to prove linear algebra relations in linear time, and the protocol has a better concrete proof size than previous works.

Keywords: interactive oracle proofs; SNARKs; Hadamard product

1 Introduction

Succinct non-interactive arguments of knowledge (SNARKs) allow efficient verification for the NP statement. In recent years, there has been a significant increase in interest and research focused on SNARKs. Many researchers have developed sophisticated protocols tailored for industrial applications, each operating under different assumptions. These protocols offer distinct features and efficiencies, making them applicable in various contexts. For example, *Plonk*[GWC19] has a constant proof size and a quasilinear prover running time, while an improved version, *HyperPlonk*[CBBZ23], sacrifices the constant proof size for a linear prover running time. Since the NP statement to be proven in real-world usage is much more complex in blockchain applications like smart contracts, the proving time is extremely crucial, as generating proofs in a reasonable time is of utmost importance. Thus, in recent years, there has been a significant surge in research focused on linear-proving-time protocols[BCGGHJ17; XZZPS19; Set20; BCG20; Lee20; Zha+21; BCL22; BMMTV21; RR22; BCHO22; CBBZ23].

There are numerous SNARKs with quasilinear proving time [CHMMVW20; LSZ22; ZSZSWG22; GWC19; RZ21], among which proofs for the Hadamard product relation or the inner product relation are necessary, and FFT(Fast Fourier Transform) is required in the proofs. A prevalent strategy to circumvent

FFT usage is the adoption of multilinear sumchecks[Set20; CBBZ23; BCHO22], which yield a linear-time prover alongside logarithmic-size proofs and verification. However, due to the inherent discrepancy between low-degree extensions for univariate polynomials and multilinear extensions for multivariate polynomials, certain calculations cannot be seamlessly translated into multilinear polynomial computations. For example, *Spartan*[Set20], which evolves from *Marlin*[CHMMVW20], incorporates an additional memory-checking subprotocol to authenticate the evaluation of the multilinear extension for a matrix. Consequently, our objective is to construct a Hadamard-product protocol and an inner-product protocol that align with the efficiency of multilinear-setting protocols, thereby streamlining the construction of linear-prover protocols from their univariate polynomial counterparts.

1.1 Our Results

In this paper, we present a novel Polynomial Interactive Oracle Proof (PIOP) designed for the Hadamard product, leveraging the univariate Lagrange-based polynomial commitment. This innovative approach is contrasted with *Spartan*[Set20], which utilizes multivariate Lagrange-based polynomial commitment. The protocol facilitates the direct application of the well-established batch KZG commitment scheme [KZG10] for univariate polynomials, resulting in a constant number of pairings in the bilinear group during the verification process. Unlike numerous protocols relying on univariate polynomials [MBKM19; CHMMVW20; LSZ22], this method eliminates dependencies on FFT, except for the setup and update processes of the KZG commitment scheme, leading to a linear proving time.

As previous researches[BCRSVW19; CHMMVW20], to check Hadamard product $\mathbf{a} \circ \mathbf{b} = \mathbf{c}$, we use a random vector $\mathbf{r}$ and show $\langle \mathbf{a} \circ \mathbf{b}, \mathbf{r} \rangle = \langle \mathbf{c}, \mathbf{r} \rangle$, where $\langle \cdot, \cdot \rangle$ means inner product. The main challenge in the protocol is to deal with the product in sumcheck. Multivariate protocols [XZZPS19; BCHO22] have a direct generalization from sumcheck for a single polynomial to sumcheck for products of multiple polynomials. However, it is not the case in univariate setting. In multivariate, in particular multilinear setting, all variables are naturally separated so that prover can limit the summing space by limit the number of variables and it is easy for prover to compute and prove a linear (or higher degree when it comes to the case of product) function, while in univariate setting we have only one variable. *Gemini*[BCHO22] solves the problem by "simulating" multivariate sumcheck through folding technique. In particular, *Gemini* folds the target polynomial of sumcheck by *splitting* its even and odd coefficients to two lower degree polynomials and take their random linear combination as the new target polynomial.

Key Observation. In Hadamard product, we need to compute the inner product of two polynomials. Under the restriction of Lagrange-based, we cannot directly compute the product polynomial $\mathbf{a}(x) \cdot \mathbf{b}(x)$ as it has a double degree while the number of evaluation points remains unchanged. We try to apply the idea of *Gemini* to our protocol. Since the splitting is the linear combination of polynomials and the inner product is quadratic, we need to do some modifications. Our observation is that the sum of product has the property

$$\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{x \in \mathbb{H}} \mathbf{a}(x)\mathbf{b}(x^{-1}) = \sum_{x \in \mathbb{H}} \mathbf{a}_1(x^2)\mathbf{b}_1(x^{-2}) + \mathbf{a}_2(x^2)\mathbf{b}_2(x^{-2})$$

for specific groups $\mathbb{H}$ and split polynomial $\mathbf{a}_1, \mathbf{b}_1, \mathbf{a}_2, \mathbf{b}_2$ if we have polynomials $\mathbf{a}(x)\mathbf{b}(x^{-1})$ that agree with $\mathbf{a}, \mathbf{b}$ on $\mathbb{H}$. In particular, $\mathbb{H}$ is a multiplicative subgroup of a finite field. By the observation, the inner product of two polynomials can be turned into the sum of inner product of split polynomials, which allows us to do the similar folding to polynomial $\mathbf{a}, \mathbf{b}$ as *Gemini*.

However, it is still not enough for Hadamard product. The left-hand side of the Hadamard product contains $\mathbf{a} \circ \mathbf{b}$, which is the Hadamard product itself. To avoid the circular argument, we choose a structured vector $\boldsymbol{\rho}$, which we call it *tensor polynomial* instead of a random $\mathbf{r}$. The feature of $\boldsymbol{\rho}$ is that checking $\langle \mathbf{a}, \boldsymbol{\rho} \rangle$ is equivalent

	Prover	Proof	Verifier
[LSZ22][ZSZSWG22]...	$O(N \log N)$	$O(1)$	$O(1)$
[BCG20]	$O(N)$	$O(N^{\epsilon})$	$O(N^{\epsilon})$
[BBBPWM18]	$O(N)$	$2 \log N \mathbb{G} + 5\mathbb{F}$	$O(N)$
[DRZ20]	$O(N)$	$8 \log N \mathbb{G}_1 + 2 \log N \mathbb{F}$	$O(\log N)$
[BMMTV21][KMP20]	$O(N)$	$2 \log N \mathbb{G}_T + 1\mathbb{G}_1 + 2\mathbb{G}_2$	$O(\log N)$
This work	$O(N)$	$(2 \log N + 18)\mathbb{G}_1 + (4 \log N + 14)\mathbb{F}$	$O(\log N)$

Table 1: Comparison to other Hadamard-product protocols on univariate polynomials. The columns represent asymptotic proving time, proof size and verification time when proving the Hadamard product of two vectors with length $N = 2^n$. For the protocol with logarithm proof size, we list the concrete proof size. By [Lee20], $1\mathbb{G}_T = 4\mathbb{G}_1 = 6\mathbb{F}$ in BLS12-381, thus [DRZ20],[BMMTV21] and [KMP20] have a close proof size.

to checking folding scheme. When checking $u = \langle \mathbf{a} \circ \boldsymbol{\rho}, \mathbf{b} \rangle = \langle \mathbf{a} \circ \mathbf{b}, \boldsymbol{\rho} \rangle$, we first compute $\mathbf{k} = \mathbf{a} \circ \boldsymbol{\rho}$ and check inner product $u = \langle \mathbf{k}, \mathbf{b} \rangle$ by folding scheme, which is equivalent to checking $u_1 = \langle \mathbf{k}, \boldsymbol{\rho}' \rangle$ and $u_2 = \langle \mathbf{b}, \boldsymbol{\rho}' \rangle$ for some u_1, u_2 computed by u. $u_2 = \langle \mathbf{b}, \boldsymbol{\rho}' \rangle$ can be checked by folding scheme. But when we check $u_1 = \langle \mathbf{k}, \boldsymbol{\rho}' \rangle = \langle \mathbf{a} \circ \boldsymbol{\rho}, \boldsymbol{\rho}' \rangle$ by similar method, it turns out to be slightly different. The problem is that a rational function must be checked in the process. Luckily, only the constant term in the polynomial is involved in a rational function, and we can deal with the constant term separately to avoid checking a rational function. By the adjustment, $u_1 = \langle \mathbf{k}, \boldsymbol{\rho}' \rangle = \langle \mathbf{a} \circ \boldsymbol{\rho}, \boldsymbol{\rho}' \rangle$ can be checked, thus finishing the proof of Hadamard product.
Commitment scheme. To instantiate polynomial oracles in the protocol, we use KZG commitment [KZG10]. The original KZG commitment is designed for coefficient-based polynomial. To modify it to Lagrange-based, we make some changes to the CRS, in particular, computing Lagrange polynomial instead of x^i terms to construct the CRS. We show that it has exactly the same property as the standard KZG commitment.
Complexity. By applying the modification of the KZG commitment, our Hadamard-product protocol achieves $O(N)$ proving time, $O(\log N)$ proof size, and verification time where N is the length of the vectors in terms of asymptotic complexity. The proof contains $2 \log N + 13$ group elements and $4 \log N + 20$ field elements, and batching k instances of proof only adds $O(k)$ elements to the proof size. The underlying inner-product protocol has proof size of $\log N + 8$ group elements and $4 \log N + 12$ field elements.
Comparison. Due to the fact that most of the univariate Hadamard-product and inner-product protocols[ILV11; LSZ22; SZ22; ZSZSWG22; GGPR13; BCRSVW19; CHMMVW20; GWC19] are designed in the situation where FFT is available, we mainly make comparison to *Bulletproof* and its improved version. Bulletproof[BBBPWM18] gives an inner-product protocol with $O(N)$ proving time, $O(\log N)$ proof size, and $O(N)$ verification time by a similar folding of our protocol. The improvement of the protocol[DRZ20; BMMTV21; KMP20] gives a $O(\log N)$ verification time, which is the closest to our protocol. The detailed comparison of proof size is listed in Table 1, and our protocol achieves the best proof size for around 30% improvement.

1.2 Related Work

In recent years there have been several inner-product protocols and Hadamard-product protocols applied to distinct SNARK protocols. The inner-product protocols and Hadamard-product protocols can be roughly divided into two groups according to proving time.

The quasilinear-proving-time group contains the directly-computed Hadamard product and inner product with zero polynomial[GGPR13; BCRSVW19; CHMMVW20; GWC19; RZ21][1] and Hadamard product and inner product with Laurent polynomials[MBKM19; ILV11; LSZ22; SZ22; ZSZSWG22]. In this group, the protocols always put the entries of vectors in the coefficients of polynomials. The directly-computed inner product and Hadamard product contain all such protocols that are not specifically designed for proof size, as it is easy and direct to compute the polynomial multiplications the quotient polynomial $\mathbf{q}(x)$ in $\mathbf{a}(x)\mathbf{b}(x) - \mathbf{b}(x) = \mathbf{q}(x)\mathbf{z}(x)$ with the help of FFT. The protocols powered by Laurent polynomials use polynomials including x^{-k} in polynomial multiplications to put the results of inner products on a specific coefficient (typically constant) of the product polynomial. The realization of these protocols are slightly different. [ILV11; LSZ22] do not directly use the x^{-k} terms. They use the degree of polynomial from x^0 to x^N and place the inner product on the coefficient of x^{N+1}. To maintain soundness, a special SRS for the KZG commitment needs to be provided that does not contain the term $[x^{N+1}]_1$. [SZ22; ZSZSWG22] uses an auxiliary polynomial that has zero coefficient in term x^{N+1} to ensure that the coefficient of term x^{N+1} is the alleged value, thus it can use a standard SRS for the KZG commitment. [MBKM19] uses bivariate Laurent polynomials $\mathbf{f}(x, y)$ where y is the variate for the random linear combination and places the alleged value on the coefficient of x^0. The SRS of such a protocol also lacks the term to commit the coefficient of x^0.

The linear-proving-time group contains the Hadamard product and inner product from multivariate sumcheck and from folding techniques. The protocol based on multivariate sumcheck includes [XZZPS19; Set20; BCHO22; CBBZ23]. *Libra*[XZZPS19] first gives a dynamic programming method to compute the intermediate results of the sumcheck for products of multilinear polynomials in linear time and [Set20; BCHO22; CBBZ23] apply the technique to inner product and Hadamard product. The protocol from folding technique includes [BCCGP16; BBBPWM18; DRZ20; BMMTV21; KMP20]. The idea is also applying Laurent polynomials, but each time they only compute polynomials with terms x and x^{-1}, halving the length of vector by separating them into two terms. By this technique, they get a linear prover. [BCCGP16; BBBPWM18] has a linear verifier, since the computations of folding are also required for verifier. [DRZ20; BMMTV21; KMP20] improves the scheme by applying polynomial commitments to [BBBPWM18], using a commit-and-prove technique to delegate verifier's computations to prover. The improvement leads to a logarithm verifier.

2 Preliminaries

2.1 Notation

We use λ to denote the security parameter. Let $\log : \mathbb{N} \to \mathbb{Z}$ be the base-2 logarithm function with rounding. Let $\mathbb{F}$ be a (finite) field and $\mathbb{F}^*$ be $\mathbb{F}\backslash\{0\}$. We use $|\mathbb{G}|$ to denote the number of entries in group $\mathbb{G}$. We will use $\mathbb{H}$ to represent a multiplicative subgroup of field $\mathbb{F}$. $\mathbb{H}$ is a cyclic group. For a vector $\mathbf{a}$ over $\mathbb{F}$ of length N and a multiplicative subgroup $\mathbb{H} = \{1, \omega, ..., \omega^{N-1}\}$ of size N, let polynomial $\mathbf{a}(x)$ be the unique polynomial of degree at most $N - 1$ that defined by $\mathbf{a}(\omega^i) = \mathbf{a}_i$ where $\mathbf{a}_i$ is i-th entry of $\mathbf{a}$. We also call the polynomial Lagrange-based. Sometimes we will also use $\mathbf{a}$ as a polynomial. We use $\deg(\mathbf{a}(x))$ to be the degree of polynomial $\mathbf{a}(x)$. Let $\mathbf{a} \circ \mathbf{b}$ be the Hadamard product of the two vectors and $\mathbf{a} \cdot \mathbf{b} = \langle \mathbf{a}, \mathbf{b} \rangle$ be the inner product of the two vectors. Sometimes we will write inner product in sum, which is $\sum_{x \in \mathbb{H}} \mathbf{a}(x)\mathbf{b}(x)$ for $\mathbf{a}, \mathbf{b} \in \mathbb{F}^N$ and $|\mathbb{H}| = N$. We use PPT as an abbreviation of probabilistic-polynomial-time.
We use $\mathbb{H}^2$ to represent the set of square of all elements in a multiplicative subgroup $\mathbb{H} \subset \mathbb{F}$, and use $\mathbb{G}^k$ for a vector containing k elements of $\mathbb{G}$ otherwise, when $\mathbb{G}$ is not claimed as a multiplicative subgroup of field $\mathbb{F}$.

[1]Most of quasilinear-prover SNARKs contains such type of sub-protocol, thus we only list protocols not containing specifically-designed proofs.

This abuse of notation can be a bit disturbing. For a multiplicative subgroup $\mathbb{H}$ with $|\mathbb{H}| = 2^n$, $\mathbb{H}^{2^n} = \{1\}$.

Definition 2.1 (even and odd parts of polynomials)**.** *Let* $\mathbf{f}(x) \in \mathbb{F}[x]$*. Let* $\mathbf{f}_e(x)$ *and* $\mathbf{f}_o(x)$ *be the even and odd parts of* $\mathbf{f}(x)$ *i.e. the unique polynomials such that* $\mathbf{f}(x) = \mathbf{f}_e(x^2) + x\mathbf{f}_o(x^2)$.

Remark 2.2 (compute even and odd parts)**.** Since all polynomials in the protocol are in the Lagrange base on a multiplicative subgroup $\mathbb{H}$, even and odd parts of polynomials cannot be separated directly. Instead, we use following equations to compute the two parts: $\mathbf{f}_e(x^2) = \frac{\mathbf{f}(x)+\mathbf{f}(-x)}{2}$ and $\mathbf{f}_o(x^2) = \frac{\mathbf{f}(x)-\mathbf{f}(-x)}{2x}$. The equations are consistent in the field $\mathbb{F}$. By the equations, we can directly define $\mathbf{f}_e(x)$ and $\mathbf{f}_o(x)$ on $\mathbb{H}^2$, which is also a multiplicative subgroup.

2.2 Polynomial Interactive Oracle Proof

A *polynomial IOP* [CHMMVW20; BFS20; BCHO22] over a field family $\mathcal{F}$ for an indexed relation $\mathcal{R}$ is a set of tuples $\mathsf{IOP} = (\mathsf{k}, \mathsf{n}, \mathsf{d}, \mathcal{I}, \mathcal{P}, \mathcal{V})$ where $\mathsf{k}, \mathsf{n}, \mathsf{d} : \{0,1\}^* \to \mathbb{N}$ are polynomial-time computable functions and $\mathcal{I}, \mathcal{P}, \mathcal{V}$ are three algorithms called *indexer, prover, verifier* respectively. The function k,n,d specifics the number of interaction rounds, the number of polynomial oracles in each round and the degree bounds on these polynomials.[2]
In round 0 (offline phase), the indexer $\mathcal{I}$ receives index $\mathbb{i}$ and a field $\mathbb{F} \in \mathcal{F}$ as input, and outputs $\mathsf{n}(0)$ polynomial oracles of degree at most d. In online phase, given an instance $\mathbb{x}$ and a witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$, prover $\mathcal{P}$ receives $(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w})$ and verifier $\mathcal{V}$ receives $(\mathbb{F}, \mathbb{x})$ and oracle access to outputs of $\mathcal{I}(\mathbb{F}, \mathbb{i})$. Prover and verifier interactive for $\mathsf{k} = \mathsf{k}(|\mathbb{i}|)$ rounds.
In each round $j = 1, ..., \mathsf{k}$, verifier sends a message $\boldsymbol{\rho}_j \in \mathbb{F}^*$ to prover, and then prover responds with $\mathsf{n}(j)$ polynomial oracles with degree bound d. Verifier may query any oracle it has received at any time. A query should be an oracle $\mathbf{P}_i^{(j)}$ verifier has already received and an element $z \in \mathbb{F}$ and its answer is $\mathbf{P}_i^{(j)}(z) \in \mathbb{F}$. After all interactions, verifier accepts or rejects by outputting a bit $\{0, 1\}$. We say that a prover (probably malicious) $\tilde{\mathcal{P}}$ to be **admissible** that for every oracle prover sends, the degree of the polynomial is bounded by d. A honest prover must be admissible. We also allow prover to send non-oracle message to verifier together with polynomial oracles as in a typical interactive proof.
We say that a polynomial IOP has perfect completeness and soundness error ϵ if the following holds.

- **Completeness.** For every field $\mathbb{F} \in \mathcal{F}$ and tuple $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$, the probability that prover $\mathcal{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w})$ convinces verifier $\mathcal{V}^{\mathcal{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x})$ to accept the proof is 1.
- **Soundness.** For every field $\mathbb{F} \in \mathcal{F}$ and tuple $(\mathbb{i}, \mathbb{x}) \notin \mathcal{L}(\mathcal{R})$, the probability for any admissible prover $\tilde{\mathcal{P}}(\mathbb{i}, \mathbb{x})$ to convince verifier $\mathcal{V}^{\mathcal{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x})$ to accept is at most ϵ.

The *proof size* we used in this paper is the sum of numbers of queries, oracles sent by prover and indexer, and non-oracle messages.

The PIOP we construct has a strong property of *knowledge soundness*(against admissible provers). We define the property below.
Knowledge Soundness. We say a PIOP has knowledge error ϵ if there exists a PPT extractor $\mathcal{E}$ for which the following holds. For every field $\mathbb{F} \in \mathcal{F}$, index-instance tuple $(\mathbb{i}, \mathbb{x})$ and admissible prover $\tilde{\mathcal{P}}$, the probability that $\mathcal{E}^{\tilde{\mathcal{P}}}(\mathbb{F}, \mathbb{i}, \mathbb{x}, 1^{\mathsf{l}(\mathbb{i})})$ outputs a valid witness $\mathbb{w}$ for $(\mathbb{i}, \mathbb{x})$ that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ is at least the probability that $\tilde{\mathcal{P}}$ convinces $\mathcal{V}^{\mathcal{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x})$ to accept minus ϵ. The notation $\mathcal{E}^{\tilde{\mathcal{P}}}$ means that the extractor $\mathcal{E}$ has a black-box access

[2]The Hadamard-product protocol itself is not necessarily to have indexer. We put it here as it can be a sub-protocol of a SNARK with indexer.

to a next-message function defined by the algorithm $\tilde{\mathcal{P}}$; in particular, the extractor $\mathcal{E}$ can "rewind" prover $\tilde{\mathcal{P}}$ to any round to get new messages.

The PIOP that we construct has two additional properties.

Public-coin. IOP is *public-coin* if each verifier's message to prover is a uniformly random string of some prescribed length. For public-coin PIOP, all queries can be postponed, w.l.o.g., to a query phase that is after the interactive phase.

Non-adaptive Queries. IOP is *non-adaptive* if all of verifier's query locations are only determined by verifier's inputs and randomness.

2.3 Tensor Polynomials

Definition 2.3. *Tensor polynomial $\boldsymbol{\rho}_{\mathbb{H},(r_1,\dots r_n)}(x)$ is the unique polynomial defined in a multiplicative subgroup $\mathbb{H}$ with $|\mathbb{H}| = 2^n$ and n random elements in the field $\mathbb{F}$ whose degree is less than 2^n and $\boldsymbol{\rho}_{\mathbb{H},(r_1,\dots r_n)}(x) = \prod_{i=1}^{n} \frac{x^{2^{i-1}}+r_i}{2x^{2^{i-1}}} = \frac{x}{2^n}\prod_{i=1}^{n}(x^{2^{i-1}}+r_i)$ for every $x \in \mathbb{H}$. When $\mathbb{H}$ or random elements are clear, we can omit them from $\boldsymbol{\rho}(x)$.*

Since $x\prod_{i=1}^{n}(x^{2^{i-1}}+r_i)$ is a monic polynomial with degree 2^n, $\boldsymbol{\rho}(x)$ can be written as
$\boldsymbol{\rho}(x) = \frac{1}{2^n}(x\prod_{i=1}^{n}(x^{2^{i-1}}+r_i) - x^{2^n}+1)$.

2.4 Lagrange Polynomials

We define Lagrange polynomials in terms of a multiplicative subgroup of a finite field here.

Definition 2.4 (Lagrange polynomial for multiplicative subgroup)**.** *For a multiplicative subgroup $\mathbb{H} = \{1, \omega, ..., \omega^{N-1}\}$ of a finite field $\mathbb{F}$ of size $N = 2^n$, Lagrange polynomial is the unique polynomial having a degree at most $N-1$ that $\mathbf{L}_k(\omega^k) = 1$ and $\mathbf{L}_k(x) = 0$ for every $x \in \mathbb{H}\backslash\{\omega^k\}$.*

Proposition 2.5. *In multiplicative subgroup $\mathbb{H} = \{1, \omega, ..., \omega^{N-1}\}$ with $N = 2^n$, $\mathbf{L}_k(x) = \frac{x^N-1}{N}\frac{\omega^k}{x-\omega^k}$.*

Proof. The core is to compute $\prod_{0\le i\ne j<N}(\omega^i-\omega^j)$ for $i = 0, ..., N-1$. Since $\omega^{\frac{N}{2}} = -1$, $\prod_{0\le i\ne j<N}(\omega^i - \omega^j) = (\omega^i - \omega^{i+\frac{N}{2}})\prod_{0\le i\ne j<\frac{N}{2}}(\omega^i-\omega^j)(\omega^i+\omega^j) = 2\omega^i\prod_{0\le i\ne j<\frac{N}{2}}(\omega^{2i}-\omega^{2j})$. Since ω^2 is the generator for $\mathbb{H}^2$, we can repeat this, and get $\prod_{0\le i\ne j<N}(\omega^i-\omega^j) = \prod_{k=0}^{n-1}2(\omega^i)^{2^k} = \frac{N}{\omega^i}$. Along with $\prod_{j=0}^{N-1}(x-\omega^j) = x^N-1$, we can get $\mathbf{L}_k(x) = \frac{x^N-1}{x-\omega^k}\frac{\omega^k}{N}$. □

3 Tensor-Check Protocol

In this section we introduce a sub-protocol involving *tensor polynomial* which is used in our construction. In Section 4.4 we show how prover evaluates the tensor polynomial on a multiplicative subgroup $\mathbb{H}$.

Definition 3.1. *The **tensor-check** relation $\mathcal{R}_{TC}$ is a set of tuples*

$$(\mathbb{i}, \mathbb{x}, \mathbb{w}) = (\perp, (\mathbb{F}, \mathbb{H}, N, r_1, ..., r_n, u), \mathbf{f})$$

where $\mathbb{H}$ is a multiplicative subgroup of field $\mathbb{F}$ with size $N = 2^n$, $r_1, ..., r_n \in \mathbb{F}$ and $\sum_{x\in\mathbb{H}}\mathbf{f}(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,...,r_n)}(x) = u$ in which $\boldsymbol{\rho}(x)$ is tensor polynomial defined as in Definition 2.3.

Theorem 3.2. *For every positive integer $N = 2^n$ and a finite field $\mathbb{F}$ that $N \mid |\mathbb{F}^*|$, there is a PIOP for $\mathcal{R}_{TC}$ with proving time $O(N)$, proof size $O(\log N)$, verification time $O(\log N)$ in terms of field operations and field elements. The PIOP has a perfect completeness and soundness error $\frac{N-1}{|\mathbb{F}^*|}$.*

We prove Theorem 3.2 by following construction.

Construction 1. *We construct a PIOP with n rounds for indexed $\mathcal{R}_{TC}$. Prover $\mathcal{P}$ takes $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ of the relation as input; verifier $\mathcal{V}$ takes $(\mathbb{i}, \mathbb{x})$ as input.*

- *Let $\mathbf{f}^{(0)}(x) = \mathbf{f}(x)$.*
- *For $i = 1, ..., n$, prover computes $\mathbf{f}^{(i)}(x^2) = \mathbf{f}_e^{(i-1)}(x^2) + r_i\mathbf{f}_o^{(i-1)}(x^2)$ $= \frac{\mathbf{f^{(i-1)}}(x)+\mathbf{f^{(i-1)}}(-x)}{2} + r_i\frac{\mathbf{f^{(i-1)}}(x)-\mathbf{f^{(i-1)}}(-x)}{2x}$ for $x \in \mathbb{H}^{2^{i-1}}$. The degree of $\mathbf{f}^{(i)}(x)$ is less than 2^{n-i}.*
- *Prover sends oracle $\mathbf{f^{(i)}}(x)$ for $i = 0, ..., n$ to verifier.*
- *Verifier sends challenge $\beta \leftarrow \mathbb{F}^*$.*
- *Prover sends $\mathbf{f^{(i-1)}}(\beta)$, $\mathbf{f^{(i-1)}}(-\beta)$ and $\mathbf{f^{(i)}}(\beta^{\mathbf{2}})$ for $i = 1, ..., n$.*
- *Verifier checks $\mathbf{f^{(i)}}(\beta^{\mathbf{2}}) = \frac{\mathbf{f^{(i-1)}}(\beta)+\mathbf{f^{(i-1)}}(-\beta)}{2} + r_i\frac{\mathbf{f^{(i-1)}}(\beta)-\mathbf{f^{(i-1)}}(-\beta)}{2\beta}$ for $i = 1, ..., n$ and $\mathbf{f}^{(n)}(\beta^2) = u$.*

We call the relation "tensor-check" due to the similarity between Construction 1 and **Construction 1** in [BCHO22]. Since our construction has the same computations and verifications as in [BCHO22], the soundness and complexity directly follow the proof in [BCHO22] and we only need to prove the completeness.

Remark 3.3. $\mathbf{f}^{(n)}$ in the protocol should be the constant u in the relation, thus it is unnecessary for verifier to query $\mathbf{f}^{(n)}(x)$, i.e. verifier can use u to replace $\mathbf{f}^{(n)}(\beta^2)$ to save one oracle and prover sends n oracles in Construction 1 to verifier.

Lemma 3.4. *Construction 1 has perfect completeness.*

Proof. Suppose $\sum_{x\in\mathbb{H}} \mathbf{f}(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,...,r_n)}(x) = u$. The verification consists of two parts. The first part is $\mathbf{f^{(i)}}(\beta^{\mathbf{2}}) = \frac{\mathbf{f^{(i-1)}}(\beta)+\mathbf{f^{(i-1)}}(-\beta)}{2} + \frac{\mathbf{f^{(i-1)}}(\beta)-\mathbf{f^{(i-1)}}(-\beta)}{2\beta}$, which is correct if prover is honest, since $\mathbf{f}_e^{(i-1)}(x)$ and $\mathbf{f}_o^{(i-1)}(x)$ are unique and consistent over field $\mathbb{F}$ in terms of $\mathbf{f}^{(i-1)}(x)$.
The second part is $\mathbf{f}^{(n)}(\beta^2) = u$. Since $\mathbf{f}^{(n)}(x)$ is defined on $\mathbb{H}^{2^n} = \{1\}$, it is a constant polynomial, thus it is equivalent to check $\mathbf{f}^{(n)}(1) = u$. We first rewrite $\mathbf{f}^{(i)}(x^2) = \frac{\mathbf{f^{(i-1)}}(x)+\mathbf{f^{(i-1)}}(-x)}{2} + r_i\frac{\mathbf{f^{(i-1)}}(x)-\mathbf{f^{(i-1)}}(-x)}{2x}$ as $\mathbf{f}^{(i)}(x^2) = \frac{x+r_i}{2x}\mathbf{f}^{(i-1)}(x) + \frac{-x+r_i}{2(-x)}\mathbf{f}^{(i-1)}(-x)$. We observe that $\mathbf{f}^{(i)}(x)$ for every $x \in \mathbb{H}^{2^i}$ is determined by 2 unique evaluations of $\mathbf{f}^{(i-1)}(x)$ on $\mathbb{H}^{2^{i-1}}$ for all $i = 1, ..., n$. Therefore, we can construct a full binary tree to describe the evaluations of all $\mathbf{f}^{(i)}$. We put $\mathbf{f}^{(n)}$ at root (level 0), and $\mathbf{f}^{(i)}$ at level $(n-i)$, and $\mathbf{f}^{(i-1)}(x)$ and $\mathbf{f}^{(i-1)}(-x)$ are siblings whose parent is $\mathbf{f}^{(i)}(x^2)$. We additionally put value $\frac{x+r_i}{2x}$ on the edge linking $\mathbf{f}^{(i-1)}(x)$ and $\mathbf{f}^{(i)}(x^2)$. This will automatically place $\frac{-x+r_i}{2(-x)}$ on the edge connecting $\mathbf{f}^{(i-1)}(-x)$ and $\mathbf{f}^{(i)}(x^2)$. Then for every internal node of the tree, the value of the node can be computed by the product of each of its child and the edge between the node and its child respectively and then summing them up. By induction, we can obtain the value of every subtree's root, which equals the sum of the product of each of its leaves and the edge on the path between the root and its leaves, respectively. Observing the path from leave $\mathbf{f}^{(0)}(x)$ to

root $\mathbf{f}^{(n)}$, the values on the edges are $\frac{x+r_1}{2x}, \frac{x^2+r_2}{2x^2}, ..., \frac{x^{2^{i-1}}+r_i}{2x^{2^{i-1}}}, ..., \frac{x^{2^{n-1}}+r_n}{2x^{2^{n-1}}}$. Thus, the root of the tree, $\mathbf{f}^{(n)}$, equals $\sum_{x\in\mathbb{H}}(\mathbf{f}^{(0)}(x)\prod_{i=1}^{n}\frac{x^{2^{i-1}}+r_i}{2x^{2^{i-1}}}) = \sum_{x\in\mathbb{H}}\mathbf{f}(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,...,r_n)}(x) = u$. □

Remark 3.5 (Batching proofs)**.** To batch m proofs for witnesses of $\mathcal{R}_{TC}$ $\mathbf{f}_0, ..., \mathbf{f}_{m-1}$ with the same $\boldsymbol{\rho}$, verifier can simply send a challenge $\xi \leftarrow \mathbb{F}^*$ to prover and run Construction 1 on polynomial $\mathbf{f}(x) = \sum_{i=0}^{m-1}\xi^i\mathbf{f}_i(x)$. This will add $\frac{m-1}{|\mathbb{F}^*|}$ to the soundness error according to the Schwartz-Zippel lemma. Note that only one challenge[3] is needed in the process.

4 Inner-Product Protocol

In this section, we describe several protocols related to the inner products of polynomials on Lagrange base.

4.1 Inverse Inner Product

Definition 4.1. *The relation **inverse inner product** $\mathcal{R}_{IIP}$ is a set of tuple*

$$(\mathbb{i}, \mathbb{x}, \mathbb{w}) = (\perp, (\mathbb{F}, \mathbb{H}, N, u), (\mathbf{f_1}, \mathbf{f_2}))$$

where $\mathbb{H}$ is a multiplicative subgroup of field $\mathbb{F}$ with size $N = 2^n$, $\mathbf{f_1}, \mathbf{f_2} \in \mathbb{F}^N$, $u \in \mathbb{F}$ and $\sum_{x\in\mathbb{H}}\mathbf{f_1}(x)\mathbf{f_2}(x^{-1}) = u$.

We call it 'inverse inner product' as it uses the inversion of the variable in polynomial $\mathbf{f}_2$.

Theorem 4.2. *For every positive integer $N = 2^n$ and a finite field $\mathbb{F}$ that $N \mid |\mathbb{F}^*|$, there is a PIOP for $\mathcal{R}_{IIP}$ with proving time $O(N)$, proof size $O(\log N)$, verification time $O(\log N)$ in terms of field operations and field elements. The PIOP has a perfect completeness and soundness error $\frac{2N}{|\mathbb{F}^*|}$.*

We prove Theorem 4.2 by following construction.

Construction 2. *We construct a PIOP with $n+2$ rounds for indexed $\mathcal{R}_{IIP}$. Prover $\mathcal{P}$ takes $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ of the relation as input; verifier $\mathcal{V}$ takes $(\mathbb{i}, \mathbb{x})$ as input.*

- *Let $\mathbf{f}_i^{(0)}(x) = \mathbf{f}_i(x)$, $u^{(0)} = u$, $i = 1, 2$.*
- *For each round $j = 1, ..., n$, let $\mathbf{f}_{i,e}^{(j-1)}$ and $\mathbf{f}_{i,o}^{(j-1)}$ be even and odd parts of polynomial $\mathbf{f}_i^{(j-1)}$, respectively. Let $\mathbf{P}^{(j)}(r) = \sum_{x\in\mathbb{H}^{2^j}}(\mathbf{f}_{1,e}^{(j-1)}(x) + r\mathbf{f}_{1,o}^{(j-1)}(x))\cdot$ $(\mathbf{f}_{2,e}^{(j-1)}(x^{-1}) + r\mathbf{f}_{2,o}^{(j-1)}(x^{-1}))$. Note that $\mathbf{P}^{(j)}$ is quadratic. Prover computes $\mathbf{f}_{i,e}^{(j-1)}(x)$ and $\mathbf{f}_{i,o}^{(j-1)}(x)$ for $x \in \mathbb{H}^{2^j}$ by $\mathbf{f}_i^{(j-1)}(x)$. Prover computes $\mathbf{P}^{(j)}(0)$, $\mathbf{P}^{(j)}(1), \mathbf{P}^{(j)}(-1)$ and send them to verifier.*
- *For each round $j = 1, ..., n$, after receiving $\mathbf{P}^{(j)}(0), \mathbf{P}^{(j)}(1), \mathbf{P}^{(j)}(-1)$, verifier checks $\mathbf{P}^{(j)}(1) + \mathbf{P}^{(j)}(-1) = u^{(j-1)}$, samples $r_j \leftarrow \mathbb{F}^*$ and sends r_j to prover. Prover and verifier compute $u^{(j)} = \mathbf{P}^{(j)}(r_j)$.*
- *For each round $j = 1, ..., n$, after receiving r_j, prover computes $\mathbf{f}_i^{(j)}(x) = \mathbf{f}_{i,e}^{(j-1)}(x) + r_j\mathbf{f}_{i,o}^{(j-1)}(x)$ for $x \in \mathbb{H}^{2^j}$ and $i = 1, 2$. Prover sends oracle $\mathbf{f}_i^{(j)}(x)$ and enters the next round.*

[3] β will be reused, thus it will only be counted in Hadamard product protocol

- *In round $n+1$, verifier checks relation among 3 constants $\mathbf{f}_1^{(n)} \cdot \mathbf{f}_2^{(n)} = u^{(n)}$ (similar to what is in tensor-check protocol). Verifier samples a challenge $\beta \leftarrow \mathbb{F}^*$ and sends it to prover.*
- *In round $n+2$, after receiving β, prover opens oracles $\mathbf{f}_i^{(j)}(x)$ where $j = 0, ..., n-1$ at $x = \beta, -\beta$ and opens $\mathbf{f}_i^{(j)}(x)$ where $j = 1, ..., n$ at $x = \beta^2$. Prover sends all openings to verifier.*
- *In round $n+2$, after receiving all openings, verifier checks $\mathbf{f}_i^{(j)}(\beta^2) = \frac{\mathbf{f}_i^{(j-1)}(\beta)+\mathbf{f}_i^{(j-1)}(-\beta)}{2} + r_j \frac{\mathbf{f}_i^{(j-1)}(\beta)-\mathbf{f}_i^{(j-1)}(-\beta)}{2\beta}$ where $i = 1, 2$ and $j = 1, ..., n$.*

Remark 4.3. Noticing that the only position we use the oracle $\mathbf{f}_i(x)$ is in round $n+2$ and the challenge is sent in round $n+1$, we can postpone all oracle sending to round n, i.e. computing and sending them together. Furthermore, by the observation that the checking equations of $\mathbf{f}_i(x)$ have the same form as those of Construction 1, we can use the batch proof of Construction 1 to shrink the proof size with $\frac{1}{|\mathbb{F}^*|}$ extra soundness error. Prover sends $n+1$ oracles, $3n$ field elements for $\mathbf{P}(x)$ and 2 field elements for $\mathbf{f}_i^{(n)}$ to verifier and verifier sends n challenges for r, 1 challenge for β and 1 challenge for batch proof.

Before proving properties of Construction 2, we first prove a lemma about $\mathbf{P}(x)$ and $\mathbf{f}(x)$.

Lemma 4.4. *For every finite field $\mathbb{F}$ and a multiplicative subgroup $\mathbb{H}$ of size $N = 2^n$ for a positive integer n, and two polynomials $\mathbf{f}_1(x), \mathbf{f}_2(x)$ whose degrees are less than N on $\mathbb{F}$, $\sum_{x\in\mathbb{H}} \mathbf{f}_1(x)\mathbf{f}_2(x^{-1}) = 2\sum_{x\in\mathbb{H}^2} \mathbf{f}_{1,e}(x)\mathbf{f}_{2,e}(x^{-1}) + \mathbf{f}_{1,o}(x)\mathbf{f}_{2,o}(x^{-1})$.*

Proof. Let $\mathbb{H} = \{1, \omega, ..., \omega^{N-1}\}$. Since $\mathbb{H}$ is a multiplicative subgroup, we have $\omega^{\frac{N}{2}} = -1$. Then $\sum_{x\in\mathbb{H}} \mathbf{f}_1(x)\mathbf{f}_2(x^{-1}) = \sum_{x\in\mathbb{H}} \mathbf{f}_1(\omega^{\frac{N}{2}}x)\mathbf{f}_2((\omega^{\frac{N}{2}}x)^{-1}) = \sum_{x\in\mathbb{H}} \mathbf{f}_1(-x)\mathbf{f}_2(-x^{-1})$. Considering

$$
\begin{aligned}
&\sum_{x\in\mathbb{H}} \mathbf{f}_1(x)\mathbf{f}_2(x^{-1}) + \sum_{x\in\mathbb{H}} \mathbf{f}_1(-x)\mathbf{f}_2(-x^{-1}) \\
=&\sum_{x\in\mathbb{H}} (\mathbf{f}_{1,e}(x^2) + x\mathbf{f}_{1,o}(x^2))(\mathbf{f}_{2,e}(x^{-2}) + x^{-1}\mathbf{f}_{2,o}(x^{-2})) \\
&+ (\mathbf{f}_{1,e}(x^2) - x\mathbf{f}_{1,o}(x^2))(\mathbf{f}_{2,e}(x^{-2}) - x^{-1}\mathbf{f}_{2,o}(x^{-2})) \\
=&2\sum_{x\in\mathbb{H}} \mathbf{f}_{1,e}(x^2)\mathbf{f}_{2,e}(x^{-2}) + \mathbf{f}_{1,o}(x^2)\mathbf{f}_{2,o}(x^{-2}) \\
=&4\sum_{x\in\mathbb{H}^2} \mathbf{f}_{1,e}(x)\mathbf{f}_{2,e}(x^{-1}) + \mathbf{f}_{1,o}(x)\mathbf{f}_{2,o}(x^{-1})
\end{aligned}
$$

, we have $\sum_{x\in\mathbb{H}} \mathbf{f}_1(x)\mathbf{f}_2(x^{-1}) = 2\sum_{x\in\mathbb{H}^2} \mathbf{f}_{1,e}(x)\mathbf{f}_{2,e}(x^{-1}) + \mathbf{f}_{1,o}(x)\mathbf{f}_{2,o}(x^{-1})$. □

Lemma 4.5. *Construction 2 has perfect completeness.*

Proof. Suppose $\sum_{x\in\mathbb{H}} \mathbf{f_1}^{(0)}(x)\mathbf{f_2}^{(0)}(x^{-1}) = u^{(0)}$. We first show that for every $j = 1, ..., n$, $\mathbf{P}^{(j)}(1) + \mathbf{P}^{(j)}(-1) = u^{(j-1)} = \sum_{x\in\mathbb{H}^{2^{j-1}}} \mathbf{f}_1^{(j-1)}(x)\mathbf{f}_2^{(j-1)}(x^{-1})$. It is trivial when $j = 1$. When $j \geq 2$, by the definition of $u^{(j)}$, we have

$$
\begin{aligned}
u^{(j-1)} &= \mathbf{P}^{(j-1)}(r_{j-1}) \\
&= \sum_{x\in\mathbb{H}^{2^{j-1}}} (\mathbf{f}_{1,e}^{(j-2)}(x) + r_{j-1}\mathbf{f}_{1,o}^{(j-2)}(x))(\mathbf{f}_{2,e}^{(j-2)}(x^{-1}) + r_{j-1}\mathbf{f}_{2,o}^{(j-2)}(x^{-1}))
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{x \in \mathbb{H}^{2^{j-1}}} \mathbf{f}_1^{(j-1)}(x)\mathbf{f}_2^{(j-1)}(x^{-1}) \\
&\overset{(1)}{=} 2 \sum_{x \in \mathbb{H}^{2^j}} \mathbf{f}_{1,e}^{(j-1)}(x)\mathbf{f}_{2,e}^{(j-1)}(x^{-1}) + \mathbf{f}_{1,o}^{(j-1)}(x)\mathbf{f}_{2,o}^{(j-1)}(x^{-1}) \\
&\overset{(2)}{=} \mathbf{P}^{(j)}(1) + \mathbf{P}^{(j)}(-1)
\end{aligned}
$$

where (1) is by Lemma 4.4 and (2) is by definition of $\mathbf{P}^{(j)}(r)$. Thus, $\mathbf{P}^{(j)}(1) + \mathbf{P}^{(j)}(-1) = u^{(j-1)}$.
Then we prove $\mathbf{f}_1^{(n)} \cdot \mathbf{f}_2^{(n)} = u^{(n)}$. By definition of $u^{(j)}$, $\mathbf{P}^{(j)}$ and $\mathbf{f}_i^{(j)}$ for $i = 1, 2$, we have $u^{(j)} = \mathbf{P}^{(j)}(r_j) = \sum_{x \in \mathbb{H}^{2^j}} (\mathbf{f}_{1,e}^{(j-1)}(x) + r_j \mathbf{f}_{1,o}^{(j-1)}(x))(\mathbf{f}_{2,e}^{(j-1)}(x^{-1}) + r_j \mathbf{f}_{2,o}^{(j-1)}(x^{-1})) = \sum_{x \in \mathbb{H}^{2^j}} \mathbf{f}_1^{(j)}(x)\mathbf{f}_2^{(j)}(x)$ for $j = 1, ..., n$. Bringing n into j, we get the proof.
The proof of $\mathbf{f}_i^{(j)}(\beta^2) = \frac{\mathbf{f}_i^{(j-1)}(\beta) + \mathbf{f}_i^{(j-1)}(-\beta)}{2} + r_j \frac{\mathbf{f}_i^{(j-1)}(\beta) - \mathbf{f}_i^{(j-1)}(-\beta)}{2\beta}$ follows the proof of Lemma 3.4. □

Lemma 4.6. *Construction 2 has soundness error* $\frac{2N}{|\mathbb{F}^*|}$.

Proof. Suppose $\sum_{x \in \mathbb{H}} \mathbf{f_1}^{(0)}(x)\mathbf{f_2}^{(0)}(x^{-1}) \neq u^{(0)}$. Since $\mathbf{f}_1^{(n)} \cdot \mathbf{f}_2^{(n)} = u^{(n)}$, there must be some j that $\sum_{x \in \mathbb{H}^{2^{j-1}}} \mathbf{f}_1^{(j-1)}(x)\mathbf{f}_2^{(j-1)}(x) \neq u^{(j-1)}$ and $\sum_{x \in \mathbb{H}^{2^j}} \mathbf{f}_1^{(j)}(x)\mathbf{f}_2^{(j)}(x) = u^{(j)}$. Let the largest one be j^*, and the quadratic polynomial in round j^* prover sends is $\mathbf{P}'(r)$ (instead of $\mathbf{P}^{(j^*)}(r)$). We have $\mathbf{P}'(1) + \mathbf{P}'(-1) = u^{(j^*-1)} \neq \sum_{x \in \mathbb{H}^{2^{j^*-1}}} \mathbf{f}_1^{(j^*-1)}(x)\mathbf{f}_2^{(j^*-1)}(x)$. Now we consider two cases.

- $\mathbf{f}_i^{(j^*)}(x) = \mathbf{f}_{i,e}^{(j^*-1)}(x) + r_{j^*}\mathbf{f}_{i,o}^{(j^*-1)}(x)$ for $i = 1, 2$. Let $\mathbf{P}^{(j^*)}(r) = \sum_{x \in \mathbb{H}^{2^{j^*}}} (\mathbf{f}_{1,e}^{(j^*-1)}(x) + r\mathbf{f}_{1,o}^{(j^*-1)}(x))(\mathbf{f}_{2,e}^{(j^*-1)}(x^{-1}) + r\mathbf{f}_{2,o}^{(j^*-1)}(x^{-1}))$. Since $\mathbf{P}'(1) + \mathbf{P}'(-1) \neq \sum_{x \in \mathbb{H}^{2^{j^*-1}}} \mathbf{f}_1^{(j^*-1)}(x)\mathbf{f}_2^{(j^*-1)}(x) = \mathbf{P}^{(j^*)}(1) + \mathbf{P}^{(j^*)}(-1)$ according to the proof in Lemma 4.5, $\mathbf{P}'(r)$ and $\mathbf{P}^{(j^*)}(r)$ are different quadratic polynomials. By $\sum_{x \in \mathbb{H}^{2^{j^*}}} \mathbf{f}_1^{(j^*)}(x)\mathbf{f}_2^{(j^*)}(x) = u^{(j^*)}$ and $\mathbf{f}_i^{(j^*)}(x) = \mathbf{f}_{i,e}^{(j^*-1)}(x) + r_{j^*}\mathbf{f}_{i,o}^{(j^*-1)}(x)$ for $i = 1, 2$, we have $u^{(j^*)} = \mathbf{P}^{(j^*)}(r_{j^*})$. But we also have $u^{(j^*)} = \mathbf{P}'(r_{j^*})$ according to verifier's computation. Since r_{j^*} is chosen after sending $\mathbf{P}'(r)$, the chance that they have the same value on r is $\frac{2}{|\mathbb{F}^*|}$ as they are both quadratic.
- $\mathbf{f}_i^{(j^*)}(x) \neq \mathbf{f}_{i,e}^{(j^*-1)}(x) + r_{j^*}\mathbf{f}_{i,o}^{(j^*-1)}(x)$ for at least one of $i = 1, 2$. Following Theorem 3.2, the chance that each one occurs is $\frac{N-1}{|\mathbb{F}^*|}$.

Summing the two cases by union bound, we have the overall soundness error $\frac{2N}{|\mathbb{F}^*|}$. □

Lemma 4.7. *Prover of Construction 2 can be implemented in $O(N)$ field operations in $\mathbb{F}$ with space complexity $O(N)$ of field elements.*

Proof. In round j, prover needs to compute $\mathbf{f}_i^{(j)}, \mathbf{P}^{(j)}$ from $\mathbf{f}_i^{(j-1)}$ for $i = 1, 2$, which both require $O(2^{n-j})$ field operations. Prover needs to store $\mathbf{f}_i^{(j)}$ for $i = 1, 2$ in order to compute $\mathbf{f}_i^{(j+1)}$ in the next round, which needs to store $O(2^{n-j})$ field elements. Thus the overall time and space complexity are both $O(2^n) = O(N)$. □

Remark 4.8 (Batching proofs)**.** For k instances of $\mathcal{R}_{IIP}$ with witness $\mathbf{f}_{i,j}(x), i = 0, ..., k-1, j = 1, 2$, verifier first sends a challenge ξ to prover and they prove $\sum_{i=0}^{k-1} \xi^i \mathbf{f}_{i,1}(x)\mathbf{f}_{i,2}(x^{-1}) = \sum_{i=0}^{k-1} \xi^i u_i$. In round 1 to n, instead of sending $\mathbf{P}_i(x)$ for each instance separately, prover sends $\sum_{i=0}^{k-1} \xi^i \mathbf{P}_i(x)$. To check the equations for $\mathbf{f}_{i,j}(\beta)$, prover and verifier still apply batch scheme for Construction 1. The soundness of such batching scheme can be proved through the same sketch of proof of Lemma 4.6, and the soundness error will be $\frac{2k(N-1)+2+3(k-1)}{|\mathbb{F}^*|}$. Using the batching scheme, prover sends $n + 2k - 1$ oracles, $3n + 2k$ field elements for $\mathbf{P}(x)$ and $\mathbf{f}_i^{(n)}$, and 2 challenges are needed.

4.2 Reversing the Vector

The relation $\mathcal{R}_{IIP}$ uses a polynomial $\mathbf{f}_2(x^{-1})$, which is not convenient for our full construction. We will introduce a new protocol to solve this problem. We first give a lemma for the new construction.

Lemma 4.9. *Suppose $\mathbf{f}(x)$ is a polynomial whose degree is less than N defined on a finite field $\mathbb{F}$ with a multiplicative subgroup $\mathbb{H}$ that $|\mathbb{H}| = N$. Let $\mathbf{f}'(x^{-1}) = \mathbf{f}(x)$ for every $x \in \mathbb{H}$ and $\mathbf{f}'(x)$ be a polynomial whose degree is also less than N, then $\mathbf{f}(x) = \mathbf{f}'(x^{-1}) \cdot x^N - \mathbf{f}'(0) \cdot x^N + \mathbf{f}'(0)$ for every $x \in \mathbb{F}$.*

Proof. Let $\mathbf{f}'(x) = b_0 + b_1 x + ... + b_{N-1}x^{N-1}$. Considering $\mathbf{f}'(x^{-1})$ on $\mathbb{H}$, since $x^N = 1$ in $\mathbb{H}$, we have

$$\mathbf{f}(x) = x^N \mathbf{f}'(x^{-1}) = x^N(b_0 + b_1 x^{-1} + ... + b_{N-1}x^{1-N}) = x^N(b_0 + b_1 x^{-1} + ... + b_{N-1}x^{1-N}) - b_0 x^N + b_0$$

Since $\mathbf{f}(x)$ and $x^N(b_0 + b_1 x^{-1} + ... + b_{N-1}x^{1-N}) - b_0 x^N + b_0$ are both polynomials whose degrees are less than N and agree on $\mathbb{H}$ that $|\mathbb{H}| = N$, they are the same polynomial. Therefore $\mathbf{f}(x) = x^N \mathbf{f}'(x^{-1}) - b_0 x^N + b_0 = x^N \mathbf{f}'(x^{-1}) - x^N \mathbf{f}'(0) + \mathbf{f}'(0)$. $\square$

We now give the definition of inner-product relation.

Definition 4.10. *The relation **inner product** $\mathcal{R}_{IP}$ is a set of tuple*

$$(\mathbb{i}, \mathbb{x}, \mathbb{w}) = (\perp, (\mathbb{F}, \mathbb{H}, N, u), (\mathbf{f_1}, \mathbf{f_2}))$$

where $\mathbb{H}$ is a multiplicative subgroup of field $\mathbb{F}$ with size $N = 2^n$, $\mathbf{f_1}, \mathbf{f_2} \in \mathbb{F}^N$, $u \in \mathbb{F}$ and $\sum_{x \in \mathbb{H}} \mathbf{f_1}(x)\mathbf{f_2}(x) = u$.

Theorem 4.11. *For every positive integer $N = 2^n$ and a finite field $\mathbb{F}$ that $N \mid |\mathbb{F}^*|$, there is a PIOP for the $\mathcal{R}_{IP}$ with proving time $O(N)$, proof size $O(\log N)$, verification time $O(\log N)$ in terms of field operations and field elements. The PIOP has a perfect completeness and soundness error $\frac{3N}{|\mathbb{F}^*|}$.*

We prove Theorem 4.11 by following construction based on Construction 2.

Construction 3. *Let $\mathbf{f}'_2(x^{-1}) = \mathbf{f}_2(x)$ on $\mathbb{H}$. Prover first sends oracle $\mathbf{f}'_2(x)$ and prover and verifier follow Construction 2 on relation $\mathcal{R}_{IIP} : (\mathbb{i}, \mathbb{x}, \mathbb{w}) = (\perp, (\mathbb{F}, \mathbb{H}, N, u), (\mathbf{f}_1, \mathbf{f}'_2))$. After the sub-protocol, verifier sends a challenge $\xi \leftarrow \mathbb{F}^*$ to prover and prover opens $\mathbf{f}'_2(x)$ at point ξ and $\mathbf{f}_2(x)$ at point ξ^{-1} along with point 0. Verifier checks $\mathbf{f}'_2(\xi) = \xi^N \mathbf{f}_2(\xi^{-1}) - \xi^N \mathbf{f}_2(0) + \mathbf{f}_2(0)$.*

By Lemma 4.9 and Theorem 4.2, the completeness and proving complexity of Construction 3 are inferred directly. The additional soundness error is due to checking $\mathbf{f}'_2(\xi) = \xi^N \mathbf{f}_2(\xi^{-1}) - \xi^N \mathbf{f}_2(0) + \mathbf{f}_2(0)$, which is $\frac{N}{|\mathbb{F}^*|}$ by Schwartz–Zippel lemma. 1 more oracle is needed in the proof than in Construction 2.

Remark 4.12 (Batching proofs)**.** For k instances of $\mathcal{R}_{IP}$ with witness $\mathbf{f}_{i,j}(x), i = 0, ..., k-1, j = 1, 2$, prover only needs to call batching proof of $\mathcal{R}_{IIP}$, and checks $\mathbf{f}'_{i,2}(\xi) = \xi^N \mathbf{f}_{i,2}(\xi^{-1}) - \xi^N \mathbf{f}_{i,2}(0) + \mathbf{f}_{i,2}(0)$ using random linear combination as for Construction 1. This will add $\frac{k}{|\mathbb{F}^*|}$ to soundness error. $n + 3k - 1$ oracles, $3n + 2k$ field elements, and 3 challenges are needed to batch k proofs.

Remark 4.13. If we are proving Hadamard product in vector form, we can directly construct an inverse polynomial to use Construction 2 instead of Construction 3 to avoid to check $\mathbf{f}'_2(\xi) = \xi^N \mathbf{f}_2(\xi^{-1}) - \xi^N \mathbf{f}_2(0) + \mathbf{f}_2(0)$.

4.3 Inner Product with Tensor Polynomial

To construct a Hadamard product, we need to do an "inner product" of 3 polynomials. We use the inner-product protocol along with the tensor polynomial to construct the "inner product".

Definition 4.14. *The relation **triple inner product** $\mathcal{R}_{TIP}$ is a set of tuple:*

$$(\mathbb{i}, \mathbb{x}, \mathbb{w}) = (\perp, (\mathbb{F}, \mathbb{H}, N, r_1, ..., r_n, u), (\mathbf{f}_1, \mathbf{f}_2))$$

,where $\mathbb{H}$ is a multiplicative subgroup of size $N = 2^n$, $\mathbf{f}_1, \mathbf{f}_2 \in \mathbb{F}^N, u \in \mathbb{F}$, and $\sum_{x \in \mathbb{H}} \mathbf{f}_1(x)\mathbf{f}_2(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,...,r_n)}(x) = u$.

Theorem 4.15. *For every positive integer $N = 2^n$ and a finite field $\mathbb{F}$ that $N \mid |\mathbb{F}^*|$, there is a PIOP for the $\mathcal{R}_{TIP}$ with proving time $O(N)$, proof size $O(\log N)$, verification time $O(\log N)$ in terms of field operations and field elements. The PIOP has perfect completeness and soundness error $\frac{5N+n-2}{|\mathbb{F}^*|}$.*

We prove Theorem 4.15 by following construction.

Construction 4. *We construct a PIOP with $2n + 3$ rounds for indexed $\mathcal{R}_{TIP}$. Prover $\mathcal{P}$ takes $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ of the relation as input; verifier $\mathcal{V}$ takes $(\mathbb{i}, \mathbb{x})$ as input.*

- *Prover computes $\mathbf{k} = \mathbf{f_1} \circ \boldsymbol{\rho}_{\mathbb{H},(r_1,...,r_n)}$ and sends oracle $\mathbf{k}(x)$ to verifier.*
- *Prover and verifier invoke protocol in Construction 3 to verify relation $\mathcal{R}_{IP} : (\mathbb{i}, \mathbb{w}, \mathbb{x}) = (\perp, (\mathbb{F}, \mathbb{H}, N, u), (\mathbf{k}, \mathbf{f}_2))$. Let randomness in round $1, 2, ..., n$ be $r'_1, ..., r'_n$.*
- *Let $\mathbf{f}_1^{(0)}(x) = \mathbf{f}_1(x)$. Prover computes oracle $\mathbf{f}_1^{(j)}(x^2) = \frac{x+r_j}{2x}\frac{x+r'_j}{2x}\mathbf{f}_1^{(j-1)}(x) + \frac{-x+r_j}{2x}\frac{-x+r'_j}{2x}\mathbf{f}_1^{(j-1)}(-x)$ for $j = 1, ..., n-1$ and $x \in \mathbb{H}^{2^{j-1}}$. Prover sends oracle $\mathbf{f}_1^{(1)}, ..., \mathbf{f}_1^{(n-1)}$ to verifier.*
- *Verifier samples $\gamma \leftarrow \mathbb{F}^*$ and sends it to prover.*
- *Prover opens $\mathbf{f}_1^{(j)}$ at $\gamma, -\gamma, \gamma^2$ and 0 for $j = 0, ..., n-1$ and sends it to verifier.*
- *For $j = 1, ..., n$, verifier checks $\mathbf{f}_1^{(j)}(\gamma^2) = \frac{r_j + r'_j}{4\gamma}(\mathbf{f}_1^{(j-1)}(\gamma) - \mathbf{f}_1^{(j-1)}(-\gamma)) + \frac{\mathbf{f}_1^{(j-1)}(\gamma)+\mathbf{f}_1^{(j-1)}(-\gamma)}{4} + \frac{r_j r'_j}{4\gamma^2}(\mathbf{f}_1^{(j-1)}(\gamma) + \mathbf{f}_1^{(j-1)}(-\gamma) - 2\mathbf{f}_1^{(j-1)}(0)) + \frac{r_j r'_j}{2}\mathbf{f}_1^{(j-1)}(0)\gamma^{\frac{N}{2^{j-1}}-2}$ and $\mathbf{f}_1^{(n)} = \mathbf{k}^{(n)}$.*

We first prove the following lemma related to oracle checking.

Lemma 4.16. *For every field $\mathbb{F}$, a multiplicative subgroup $\mathbb{H}$ with size $N = 2^n$ and a polynomial $\mathbf{f}(x)$ whose degree is less than N, defining a polynomial $\mathbf{f}'(x)$ with degree less than $\frac{N}{2}$ that $\mathbf{f}'(x^2) = \frac{x+r}{2x}\frac{x+r'}{2x}\mathbf{f}(x) + \frac{-x+r}{2(-x)}\frac{-x+r'}{2(-x)}\mathbf{f}(-x)$ for every $x \in \mathbb{H}$, then for every point $x \in \mathbb{F}$, $\mathbf{f}'(x^2) = \frac{r+r'}{4x}(\mathbf{f}(x) - \mathbf{f}(-x)) + \frac{\mathbf{f}(x)+\mathbf{f}(-x)}{4} + \frac{rr'}{4x^2}(\mathbf{f}(x) + \mathbf{f}(-x) - 2\mathbf{f}(0)) + \frac{rr'}{2}\mathbf{f}(0)x^{N-2}$.*

Proof. We first observe $\mathbf{f}(x)$ for $x \in \mathbb{H}$; for $x \in \mathbb{H}$, $\mathbf{f}'(x^2) = \frac{r+r'}{4x}(\mathbf{f}(x) - \mathbf{f}(-x)) + \frac{\mathbf{f}(x)+\mathbf{f}(-x)}{4} + \frac{rr'}{4x^2}(\mathbf{f}(x) + \mathbf{f}(-x))$. The problem is the right hand side of the equation is not a polynomial; in particular $\frac{rr'}{4x^2}(\mathbf{f}(x) + \mathbf{f}(-x))$ is not a polynomial. It is a rational function. What we do next is to find a polynomial that agrees with $\frac{rr'}{4x^2}(\mathbf{f}(x) + \mathbf{f}(-x))$ on $\mathbb{H}$ and has a degree less than N. Observing that $\frac{rr'}{4x^2}(\mathbf{f}(x) + \mathbf{f}(-x))$ is a polynomial if $\mathbf{f}$ has no constant term (linear term is cancelled), we can split the rational function as $\frac{rr'}{4x^2}(\mathbf{f}(x)+\mathbf{f}(-x)) = \frac{rr'}{4x^2}(\mathbf{f}(x)+\mathbf{f}(-x)-2\mathbf{f}(0)) + \frac{rr'}{2x^2}\mathbf{f}(0)$. Since $\frac{rr'}{2x^2}\mathbf{f}(0) = x^N \cdot \frac{rr'}{2x^2}\mathbf{f}(0) = \frac{rr'}{2}\mathbf{f}(0)x^{N-2}$ for $x \in \mathbb{H}$ as $\mathbb{H}$ is a multiplicative subgroup of size N, we have completed the proof. □

Lemma 4.17. *Construction 4 has perfect completeness.*

Proof. Suppose $\sum_{x\in\mathbb{H}} \mathbf{f}_1(x)\mathbf{f}_2(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x) = u$. Then $\mathbf{k} = \mathbf{f}_1 \circ \boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}$, $\sum_{x\in\mathbb{H}} \mathbf{k}(x)\mathbf{f}_2(x) = u$. Thus $(\mathbf{k}, \mathbf{f}_2)$ is witness for relation $\mathcal{R}_{IP}$, and the sub-protocol can be correctly implemented. By the proof of Lemma 3.4, $\mathbf{k}^{(n)} = \sum_{x\in\mathbb{H}} \mathbf{k}(x)\boldsymbol{\rho}_{\mathbb{H},(r'_1,\ldots,r'_n)}(x) = \sum_{x\in\mathbb{H}} \mathbf{f}_1(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x)\boldsymbol{\rho}_{\mathbb{H},(r'_1,\ldots,r'_n)}(x)$.
Recall the tree we construct in the proof of Lemma 3.4. When we replace the value on the edge $\frac{x+r_i}{2x}$ with $\frac{x+r_i}{2x}\frac{x+r'_i}{2x}$, we can get $\mathbf{f}_1^{(n)}(x) = \sum_{x\in\mathbb{H}}(\mathbf{f}^{(0)}(x)\prod_{i=1}^{n}\frac{x^{2^{i-1}}+r_i}{2x^{2^{i-1}}}\prod_{i=1}^{n}\frac{x^{2^{i-1}}+r'_i}{2x^{2^{i-1}}}) = \sum_{x\in\mathbb{H}} \mathbf{f}(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x)\boldsymbol{\rho}_{\mathbb{H},(r'_1,\ldots,r'_n)}(x)$ $= \mathbf{k}^{(n)}$. The oracle checking follows Lemma 4.16. □

To prove soundness error, we need a lemma to show $\boldsymbol{\rho}_{\mathbb{H},(r'_1,\ldots,r'_n)}$ is "random enough" to mask a polynomial.

Lemma 4.18. *For every field $\mathbb{F}$, a multiplicative subgroup $\mathbb{H}$ with size $N = 2^n$ and two polynomials $\mathbf{f}_1(x)$ and $\mathbf{f}_2(x)$ whose degrees are less than N, if $(r_1, \ldots, r_n)$ are all randomly chosen from $\mathbb{F}$ and $\sum_{x\in\mathbb{H}} \mathbf{f}_1(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x) = \sum_{x\in\mathbb{H}} \mathbf{f}_2(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x)$, then $\mathbf{f}_1(x) = \mathbf{f}_2(x)$ except for probability $\frac{n}{|\mathbb{F}^*|}$.*

Proof. Suppose $\mathbf{f}_1(x) = a_0 + a_1x + \ldots + a_{N-1}x^{N-1}$ and $\mathbf{f}_2(x) = b_0 + b_1x + \ldots + b_{N-1}x^{N-1}$. We can apply Construction 1 to both two polynomials. Following **Lemma 5.3** of [BCHO22], observing that when we compute $\mathbf{f}_1^{(n)}$, $a_{(l_1l_2\ldots l_n)_2}$ are 'folded' in sequence $(l_n, \ldots, l_1)$ when $(l_1l_2\ldots l_n)_2$ is binary representation of i for $i = 0, \ldots, N-1$, we can prove by induction that $\mathbf{f}_1^{(n)} = \sum_{(l_1,l_2,\ldots,l_n)\in\{0,1\}^n}(a_{(l_n\ldots l_1)_2})r_1^{l_1}\ldots r_n^{l_n} = \mathbf{f}_2^{(n)} = \sum_{(l_1,l_2,\ldots,l_n)\in\{0,1\}^n}(b_{(l_n\ldots l_1)_2})r_1^{l_1}\ldots r_n^{l_n}$. Viewing $\mathbf{S}(r_1, \ldots, r_n) = \sum_{(l_1,l_2,\ldots,l_n)\in\{0,1\}^n}(a_{(l_n\ldots l_1)_2} - b_{(l_n\ldots l_1)_2})r_1^{l_1}\ldots r_n^{l_n}$ as a multivariate polynomial on variable $(r_1, \ldots, r_n)$, then by Schwartz–Zippel lemma, if $\mathbf{f}_1(x) \neq \mathbf{f}_2(x)$, i.e. $\mathbf{S}(r_1, \ldots, r_n) \not\equiv 0$, the probability that $\mathbf{S}(r_1, \ldots, r_n) = 0$ is at most $\frac{n}{|\mathbb{F}|} \leq \frac{n}{|\mathbb{F}^*|}$ for random $(r_1, \ldots, r_n)$ from $\mathbb{F}$. □

Lemma 4.19. *Construction 4 has soundness error $\frac{5N+n-2}{|\mathbb{F}^*|}$.*

Proof. Suppose $\sum_{x\in\mathbb{H}} \mathbf{f}_1(x)\mathbf{f}_2(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x) \neq u$. Since the verification of sub-protocol is passed, $\sum_{x\in\mathbb{H}} \mathbf{k}(x)\mathbf{f}_2(x) = u$ except for $\frac{3N}{|\mathbb{F}^*|}$ probability. Along with $\sum_{x\in\mathbb{H}} \mathbf{f}_1(x)\mathbf{f}_2(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x) \neq u$, we have $\mathbf{f}_1(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)} \neq \mathbf{k}(x)$ at some point $x \in \mathbb{H}$ except for $\frac{3N}{|\mathbb{F}^*|}$ probability. Also, by proof of soundness of Construction 1 in [BCHO22], $\sum_{x\in\mathbb{H}} \mathbf{k}(x)\boldsymbol{\rho}_{\mathbb{H},(r'_1,\ldots,r'_n)} = \mathbf{k}^{(n)}$, except for the probability $\frac{N-1}{|\mathbb{F}^*|}$.
Now we consider the oracle that checks $\mathbf{f}_1^{(j)}$ on $j = 1, \ldots, n$. By Lemma 4.16, following the proof of soundness of Construction 1, $\sum_{x\in\mathbb{H}} \mathbf{f}_1(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}\boldsymbol{\rho}_{\mathbb{H},(r'_1,\ldots,r'_n)} = \mathbf{f}_1^{(n)}$ except for probability $\frac{N-1}{|\mathbb{F}^*|}$. Then $\sum_{x\in\mathbb{H}} \mathbf{f}_1(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}\boldsymbol{\rho}_{\mathbb{H},(r'_1,\ldots,r'_n)}$

$= \mathbf{f}_1^{(n)} = \mathbf{k}^{(n)} = \sum_{x\in\mathbb{H}} \mathbf{k}(x)\boldsymbol{\rho}_{\mathbb{H},(r'_1,\ldots,r'_n)}$ except for probability $\frac{2N-2}{|\mathbb{F}^*|}$.
We then suppose that $\mathbf{f}_1(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)} \neq \mathbf{k}(x)$ and $\sum_{x\in\mathbb{H}} \mathbf{f}_1(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}\boldsymbol{\rho}_{\mathbb{H},(r'_1,\ldots,r'_n)} = \mathbf{k}^{(n)}$ $= \sum_{x\in\mathbb{H}} \mathbf{k}(x)\boldsymbol{\rho}_{\mathbb{H},(r'_1,\ldots,r'_n)}$. Since $\mathbf{k}(x)$ is sent before $(r'_1,...,r'_n)$ are chosen, by Lemma 4.18, $\mathbf{f}_1(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}$ $= \mathbf{k}(x)$ except for probability $\frac{n}{|\mathbb{F}^*|}$. Thus, the overall soundness error is $\frac{3N}{|\mathbb{F}^*|} + \frac{2N-2}{|\mathbb{F}^*|} + \frac{n}{|\mathbb{F}^*|} = \frac{5N+n-2}{|\mathbb{F}^*|}$. □

Lemma 4.20. *Prover of Construction 4 can be implemented in $O(N)$ field operations in $\mathbb{F}$ with space complexity $O(N)$ of field elements.*

Proof. The sub-protocol of $\mathcal{R}_{IP}$ and computing and storing $\mathbf{k}(x)$ cost $O(N)$ field operations $O(N)$ space in terms of field elements. Computing and storing $\mathbf{f}_1^{(j)}(x)$ cost $O(2^{n-j})$ field operations and $O(2^{n-j})$ space in terms of field elements. Thus the overall time and space complexity for Construction 4 are both $O(N)$. □

Remark 4.21 (Batching proofs)**.** It is easy to batch the proofs on sub-protocol level. The sub-protocol for $\mathcal{R}_{IP}$ can be batched by the scheme in Remark 4.12. The rest part $\sum_{x\in H} \mathbf{f}_1(x)\rho_{\mathbb{H},r_1,\ldots,r_n}(x)\rho_{\mathbb{H},r'_1,\ldots,r'_n}(x) = k^{(n)}$ can be batched by a random linear combination as in Remark 3.5. Furthermore, Construction 4 can be 'partially batched', which means the sub-protocol for $\mathcal{R}_{IP}$ in the protocol can be batched with an independent instance of Construction 3.

4.4 Computing Tensor Polynomial

In Construction 4, prover needs to compute $\mathbf{k} = \mathbf{f}_1(x) \circ \boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}$. To maintain the proving time of $O(N)$, prover needs to compute $\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)} = \frac{1}{2^n}(x\prod_{i=1}^n (x^{2^{i-1}} + r_i))$ on $\mathbb{H}$ in $O(N)$ time. Directly computing requires $O(N\log N)$ time. Observing that all elements $x \in \mathbb{H}$ only has 2^j different evaluations on $x^{2^{n-j}}$, we can compute $\boldsymbol{\rho}_{r_1,\ldots,r_n}(x)$ in a reverse direction.
We set a full binary tree to save the intermediate results. The tree has $n+1$ levels from 0 to n, and each node saves 2 values. Level 0 (root) has value (1,0). Level i has 2^i nodes indexed from 0 to $2^i - 1$. We use $u_{i,j}$ to represent node j at level i. Let $\mathbb{H} = (1, \omega, ..., \omega^{N-1})$. Let $u_{i,j} = (u_{i-1,\lfloor j/2\rfloor,0} \cdot (\omega^{u_{i,j,1}} + r_{n-i+1}), u_{i-1,\lfloor j/2\rfloor,1}/2 + lsb(j)\cdot 2^{n-1})$ where $lsb(j)$ is the least significant bit of j. Since the tree has $O(N)$ nodes and the time to compute each node (i,j) from its parent $(i-1,\lfloor j/2\rfloor)$ is $O(1)$, we can compute the tree in $O(N)$ time.

Lemma 4.22. $u_{i,j,0} = \prod_{k=n-i+1}^n (\omega^{u_{i,j,1}\cdot 2^{k-(n-i+1)}} + r_k)$ *for every node (i,j) in the tree, and $(u_{i,j,1})$ for all nodes in level i is a permutation of* $(0\cdot\frac{N}{2^i}, ..., (2^i-1)\cdot\frac{N}{2^i})$.

Proof. We prove the lemma by induction.

1. When $i=0$, $u_{i,j,0} = 1$ and $u_{i,j,1} = 0$ is a permutation of (0).
2. Suppose that for level $i-1$ and $i \le n$, $u_{i-1,j,0} = \prod_{k=n-i+2}^n (\omega^{u_{i-1,j,1}\cdot 2^{k-(n-i+2)}} + r_k)$, and $(u_{i-1,j,1})$ for level $i-1$ is a permutation of $(0\cdot\frac{N}{2^{i-1}}, ..., (2^{i-1}-1)\cdot\frac{N}{2^{i-1}})$. We consider $u_{i,2j,1}$ and $u_{i,2j+1,1}$ for $j = 0, ..., 2^{i-1}$ together. We have $u_{i,2j,1} = u_{i-1,j,1}/2$ and $u_{i,2j+1,1} = u_{i-1,j,1}/2 + 2^{n-1}$. Since $(u_{i-1,j,1})$ for level $i-1$ is a permutation of $(0\cdot\frac{N}{2^{i-1}}, ..., (2^{i-1}-1)\cdot\frac{N}{2^{i-1}})$, we have $(u_{i,2j,1})$ is a permutation of $(0\cdot\frac{N}{2^i}, ..., (2^{i-1}-1)\cdot\frac{N}{2^i})$ and $(u_{i,2j+1,1})$ is a permutation of $(2^{i-1}\cdot\frac{N}{2^i}, ..., (2^i-1)\cdot\frac{N}{2^i})$, thus finishing the

proof about $u_{i,j,1}$.
Now we consider $u_{i,j,0}$. Observing that $(\omega^{u_{i,j,1}})^2 = \omega^{u_{i-1,\lfloor j/2\rfloor,1}}$, we have

$$u_{i-1,\lfloor j/2\rfloor,0} = \prod_{k=n-i+2}^{n} (\omega^{u_{i-1,\lfloor j/2\rfloor,1}\cdot 2^{k-(n-i+2)}} + r_k) = \prod_{k=n-i+2}^{n} (\omega^{u_{i,j,1}\cdot 2^{k-(n-i+1)}} + r_k)$$

Thus $u_{i,j,1} = u_{i-1,\lfloor j/2\rfloor,0} \cdot (\omega^{u_{i,j,1}} + r_{n-i+1}) = \prod_{k=n-i+1}^{n} (\omega^{u_{i,j,1}\cdot 2^{k-(n-i+1)}} + r_k)$.

□

Considering level n and definition of $\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}$, we have the following corollary.

Corollary 4.23. $\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(\omega^{u_{n,j,1}}) = \frac{\omega^{u_{n,j,1}}}{2^n} u_{n,j,0}$.

5 Hadamard Product

In this section, we will introduce a PIOP for Hadamard product based on PIOP for $\mathcal{R}_{TIP}$, $\mathcal{R}_{IP}$ and $\mathcal{R}_{TC}$.

Definition 5.1. *The* ***Hadamard-product*** *relation* $\mathcal{R}_{HP}$*is a set of tuples*

$$(\mathbb{i}, \mathbb{x}, \mathbb{w}) = (\perp, (\mathbb{F}, \mathbb{H}, N), (\mathbf{a}, \mathbf{b}, \mathbf{c}))$$

where $\mathbb{H}$ *is a multiplicative subgroup with size* $N = 2^n$*,* $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{F}^N, u \in \mathbb{F}$*, and for every* $x \in \mathbb{H}$*, we have* $\mathbf{a}(x)\mathbf{b}(x) = \mathbf{c}(x)$*.*

Remark 5.2. The indexed relation defined above does not contain index. It is reasonable when it is used as a sub-protocol of R1CS proving as [BCHO22]. But when it is considered as a standalone protocol, there should be oracles $\mathbf{a}, \mathbf{b}, \mathbf{c}$ as indices or being sent by prover depending on whether they are viewed as indices or witnesses.

Theorem 5.3. *For every positive integer* $N = 2^n$ *and a finite field* $\mathbb{F}$ *that* $N \mid |\mathbb{F}^*|$*, there is a PIOP for the* $\mathcal{R}_{HP}$ *with proving time* $O(N)$*, proof size* $O(\log N)$*, verification time* $O(\log N)$ *in terms of field operations and field elements. The PIOP has perfect completeness and soundness error* $\frac{9N+2n-4}{|\mathbb{F}^*|}$*.*

We prove Theorem 5.3 by following construction.

Construction 5. *We construct a PIOP with* $3n + 4$ *rounds for indexed* $\mathcal{R}_{HP}$*. Prover* $\mathcal{P}$ *takes* $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ *of the relation as input; verifier* $\mathcal{V}$ *takes* $(\mathbb{i}, \mathbb{x})$ *as input.*

- *Verifier sends* $(r_1, ..., r_n) \leftarrow (\mathbb{F}^*)^n$ *to prover.*
- *Prover computes* $u = \sum_{x\in\mathbb{H}} \mathbf{c}(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x)$ *by applying prover side of Construction 1.*
- *Prover sends* u *and oracle* $\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x)$ *to verifier.*
- *Prover and verifier invoke protocol in Construction 3 to verify relation* $\mathcal{R}_{IP} : (\mathbb{i}, \mathbb{x}, \mathbb{w}) = (\perp, (\mathbb{F}, \mathbb{H}, N, u), (\mathbf{c}, \boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x)))$ *and Construction 4 to verify relation* $\mathcal{R}_{TIP} : (\mathbb{i}, \mathbb{x}, \mathbb{w}) = (\perp, (\mathbb{F}, \mathbb{H}, N, r_1, ..., r_n, u), (\mathbf{a}, \mathbf{b}))$*. Verifier checks the consistency of oracle* $\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x)$ *by querying at point* β *and locally computing* $\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(\beta)$*. Note that the two sub-protocol can be batched by Remark 4.21.*

Remark 5.4 (Public-coin)**.** Since the protocol is public-coin, we can postpone all queries to oracles in each sub-protocol to the end of the protocol and choose the same challenge $\beta \leftarrow \mathbb{F}^*$ for all equations, which will be better for batch queries.

Lemma 5.5. *Construction 5 has perfect completeness.*

Proof. Suppose $\mathbf{a}(x)\mathbf{b}(x) = \mathbf{c}(x)$ for every $x \in \mathbb{H}$. Then $\sum_{x\in\mathbb{H}} \mathbf{a}(x)\mathbf{b}(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x)$
$= \sum_{x\in\mathbb{H}} \mathbf{c}(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x) = u$. By perfect completeness of Construction 1,Construction 3 and Construction 4, Construction 5 has perfect completeness. □

Lemma 5.6. *Construction 5 has soundness error* $\frac{9N+2n-4}{|\mathbb{F}^*|}$.

Proof. Suppose $\mathbf{a}(x)\mathbf{b}(x) \neq \mathbf{c}(x)$ for some $x \in \mathbb{H}$. Then by Lemma 4.18,
$\sum_{x\in\mathbb{H}} \mathbf{a}(x)\mathbf{b}(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x) \neq \sum_{x\in\mathbb{H}} \mathbf{c}(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x)$ except for $\frac{n}{|\mathbb{F}^*|}$ probability. Suppose prover sends u' in step 2, then
$\sum_{x\in\mathbb{H}} \mathbf{a}(x)\mathbf{b}(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x) \neq u'$ or $\sum_{x\in\mathbb{H}} \mathbf{c}(x)\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x) \neq u'$ except for $\frac{n}{|\mathbb{F}^*|}$ probability. By soundness of Construction 3 and Construction 4, and the soundness error by checking oracle $\boldsymbol{\rho}_{\mathbb{H},(r_1,\ldots,r_n)}(x)$, we get the overall soundness error $\frac{3N}{|\mathbb{F}^*|} + \frac{5N+n-2}{|\mathbb{F}^*|} + \frac{N-1}{|\mathbb{F}^*|} + \frac{n}{|\mathbb{F}^*|} = \frac{9N+2n-4}{|\mathbb{F}^*|}$. □

Remark 5.7 (Knowledge soundness)**.** The knowledge soundness for standalone Hadamard-product protocol is trivial, since verifier can rewind and query $\mathbf{a}, \mathbf{b}, \mathbf{c}$ for N time at distinct points if verifier accepts over the probability greater than soundness error above.

Remark 5.8. In order to reduce the randomness sampled from verifier, we may use $(r^{2^0}, r^{2^1}, \ldots, r^{2^{n-1}})$ to replace $(r_1, \ldots, r_n)$. The change will cause the soundness error of Lemma 4.18 increase from $\frac{n}{|F*|}$ to $\frac{N-1}{|F*|}$, and the soundness error of Construction 5 will increase from $\frac{9N+2n-4}{|\mathbb{F}^*|}$ to $\frac{11N-6}{|\mathbb{F}^*|}$.

Lemma 5.9. *Prover of Construction 5 can be implemented in $O(N)$ field operations in $\mathbb{F}$ with space complexity $O(N)$ of field elements.*

Proof. Prover needs to run prover side of Construction 1, Construction 3 and Construction 4, which will all cost $O(N)$ field operations and $O(N)$ space for field elements. Thus, prover's time and space complexity in terms of field operations and field elements, respectively, are all $O(N)$. □

Remark 5.10 (Batching equation checkings)**.** There are $3n + 2$ equations to check in Construction 5. After choosing the challenge β, verifier can also choose a challenge $\alpha \leftarrow \mathbb{F}^*$. Since β is chosen, all the equations can be written in form $c_{i,1}\mathbf{f}_{i,1}(\beta^2) = c_{i,2}\mathbf{f}_{i,2}(\beta) + c_{i,3}\mathbf{f}_{i,3}(-\beta) + c_{i,4}\mathbf{f}_{i,4}(0)$ for $i = 0, \ldots, 3n-1$ where $\mathbf{f}_{i,j}$ is one of the oracles sent by prover and $c_{i,j}$ is a fixed field element that can be computed in $O(n)$ by both prover and verifier, except the oracle checking in Construction 3 and oracle checking in Construction 5. By choosing α, prover and verifier can only check the equation $\sum_{i=0}^{3n-1} \alpha^i c_{i,1}\mathbf{f}_{i,1}(\beta^2) = \sum_{i=0}^{3n-1} \alpha^i c_{i,2}\mathbf{f}_{i,2}(\beta) + \sum_{i=0}^{3n-1} \alpha^i c_{i,3}\mathbf{f}_{i,3}(-\beta) + \sum_{i=0}^{3n-1} \alpha^i c_{i,4}\mathbf{f}_{i,4}(0)$. The linear combination will lead to $\frac{3n}{|\mathbb{F}^*|}$ extra soundness error, but it will be useful if the instantiation of oracle is additive.

Remark 5.11 (Batching proofs)**.** It is easy to batch the whole proof by simply batching the sub-protocols with the same randomness r. It also has the property of 'partial batching' as mentioned in Remark 4.21.

6 Polynomial Commitment Scheme

To instantiate the oracle in PIOP above, we will use the polynomial commitment scheme. In this section, we will describe a polynomial commitment scheme, which is a slight modification of the KZG commitment, to commit polynomials in Lagrange base. We first define polynomial commitment scheme following [KZG10; BCHO22].

6.1 Definition

The polynomial commitment scheme over a field family $\mathcal{F}$ contains a tuple of algorithm $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify})$ defined by following syntax.

- **Setup** $\mathsf{PC.Setup}(1^\lambda, D) \to (ck, vk)$. On input a security parameter λ in unary and a degree D, $\mathsf{PC.Setup}$ outputs a commitment key ck and a verification key vk containing a description of $\mathbb{F} \in \mathcal{F}$.
- **Commit** $\mathsf{PC.Com}(ck, \mathbf{P}) \to C$. On input the commitment key ck and a polynomial $\mathbf{P}$ of degree at most D, $\mathsf{PC.Com}$ outputs a commitment C for $\mathbf{P}$.
- **Open** $\mathsf{PC.Open}(ck, \mathbf{P}, z) \to \pi$. On input the commitment key ck, a polynomial $\mathbf{P}$ of degree at most D and an evaluation point z, $\mathsf{PC.Open}$ outputs a proof π for the evaluation.
- **Verify** $\mathsf{PC.Verify}(rk, C, \pi, z, v) \to \{0, 1\}$. On input the verification key rk, the commitment C, the proof π, an evaluation point z and an alleged value v, $\mathsf{PC.Verify}$ outputs 1 if π is a valid proof that $\mathbf{P}$ is a polynomial having degree at most D, committed in C and $\mathbf{P}(z) = v$, and 0 otherwise.

To prove knowledge soundness, a polynomial commitment scheme must satisfy completeness and extractability (as a stronger notion of evaluation binding in [CHMMVW20; BCHO22]).

Definition 6.1 (Completeness)**.** *For every degree bound $D \in \mathbb{N}$ and every PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c|r} & (ck, rk) \leftarrow \mathsf{PC.Setup}(1^\lambda, D) \\ \deg(\mathbf{P}) \leq D & (\mathbf{P}, z) \leftarrow \mathcal{A}(ck, rk) \\ \downarrow & C \leftarrow \mathsf{PC.Com}(ck, \mathbf{P}) \\ \mathsf{PC.Verify}(rk, C, \pi, z, v) = 1 & v = \mathbf{P}(z) \\ & \pi \leftarrow \mathsf{PC.Open}(ck, \mathbf{P}, z) \end{array}\right] \geq 1 - \mathrm{negl}(\lambda)$$

Definition 6.2 (Extractability)**.** *For any degree bound $D \in \mathbb{N}$ and any PPT adversary $\mathcal{A}$, there exists an extractor $\mathcal{E}$ such that for every round bound $r \in \mathbb{N}$, PPT query sampler $\mathcal{Q}$ and PPT adversary $\mathcal{B}$,*

$$\Pr\left[\begin{array}{c|r} & (ck, rk) \leftarrow \mathsf{PC.Setup}(1^\lambda, D) \\ \mathsf{PC.Verify}(rk, C, \pi, z, v) = 1 & C \leftarrow \mathcal{A}(ck, rk) \\ \downarrow & \mathbf{P} \leftarrow \mathcal{E}(ck, rk) \\ \deg(\mathbf{P}) \leq D \wedge v = \mathbf{P}(z) & z \leftarrow \mathcal{Q}(ck, rk) \\ & (\pi, v, st) \leftarrow \mathcal{B}(ck, rk, z) \end{array}\right] \geq 1 - \mathrm{negl}(\lambda)$$

6.2 Modified Polynomial Commitment Scheme

To introduce our commitment scheme, we first need to describe original KZG commitment[KZG10; CHMMVW20; BCHO22]. In summary, for a polynomial in coefficient base $\mathbf{P}(x)$, the KZG commitment scheme commits it to $C = \mathbf{P}(\tau)G$ where G is a group generator in bilinear pairing and τ is a field element chosen by a trusted third party. To prove $v = \mathbf{P}(z)$, prover computes $\mathbf{w}(x) = \frac{\mathbf{P}(x)-\mathbf{P}(\mathbf{z})}{x-z}$ and outputs a proof $\mathbf{w}(\tau)G$. We want our commitment scheme to have the same outputs on the same polynomial. Since we need a polynomial commitment scheme for Lagrange-based polynomial, we need to construct a commitment key Σ with a vector of Lagrange polynomial(Definition 2.4). One of the drawbacks is that only polynomials which have a proper degree, such as $2^k - 1$ for some integer $2^k - 1$, can be committed. We can solve this by padding origin polynomials to such degree and force D to be $2^n - 1$ for some integer n. Another drawback is that Lagrange polynomials to commit polynomials of some degree cannot be used in polynomials of other degrees. Thus, to commit polynomials of different degrees, we need Lagrange polynomials for all degrees. Since we need to commit to degree $2^k - 1$ in our protocol, the size of vector Σ will be doubled.

Below we will introduce our modification of the KZG commitment constructed by Lagrange polynomials. The idea is the same as [TABDFK20].

- **Setup** $\mathsf{PC.Setup}(1^\lambda, n) \to (ck, vk)$. First, $\mathsf{PC.Setup}$ samples a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, G, H, e) \leftarrow \mathsf{GroupGen}(1^\lambda)$. $\mathsf{PC.Setup}$ then samples $\tau \leftarrow \mathbb{F}_q$ and a multiplicative subgroup $\mathbb{H} \subset \mathbb{F}$ that $|\mathbb{H}| = 2^n$, computes τH and vector

$$\begin{aligned}\Sigma = (&\mathbf{L}_{0,0}(\tau)G, \mathbf{L}_{1,0}(\tau)G, \mathbf{L}_{1,1}(\tau)G, ..., \\ &\mathbf{L}_{n,0}(\tau)G, ..., \mathbf{L}_{n,2^n-1}(\tau)G) = (G_{0,0}, ..., G_{n,2^n-1}) \in \mathbb{G}_1^{2^{n+1}-1}\end{aligned}$$

  where $\mathbf{L}_{i,j}$ is the j-th Lagrange polynomial defined on $\mathbb{H}^{2^{n-i}}$. $\mathsf{PC.Setup}$ outputs $ck = ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, G, H, e), \Sigma, \omega)$ and $vk = (G, H, \tau H)$. Only polynomials whose degrees are $2^k - 1$ for $k = 1, ..., n$ can be committed by ck.
- **Commit** $\mathsf{PC.Com}(ck, \mathbf{P}) \to C$. On input $\mathbf{P}(x)$ in Lagrange base of degree $d = 2^k - 1 \le 2^n - 1$, $\mathsf{PC.Com}$ outputs $C = \sum_{j=0}^{d} \mathbf{P}(\omega^{2^{n-k}\cdot j})G_{k,j} \in \mathbb{G}_1$.
- **Open** $\mathsf{PC.Open}(ck, \mathbf{P}, z) \to \pi$. On input $\mathbf{P}(x)$ in Lagrange base of degree $d = 2^k - 1 \le 2^n - 1$, $\mathsf{PC.Open}$ first computes $\mathbf{P}(z) = \frac{z^{d+1}-1}{d+1} \sum_{j=0}^{d} \frac{\mathbf{P}(\omega^{2^{n-k}\cdot j})\cdot \omega^{2^{n-k}\cdot j}}{z-\omega^{2^{n-k}\cdot j}}$ [4], then outputs $\pi = \sum_{j=0}^{d} \frac{\mathbf{P}(\omega^{2^{n-k}\cdot j}) - \mathbf{P}(z)}{\omega^{2^{n-k}\cdot j} - z} G_{k,j}$ if $z \notin \mathbb{H}^{2^{n-k}}$, and aborts if $z \in \mathbb{H}^{2^{n-k}}$.
- **Verify** $\mathsf{PC.Verify}(rk, C, \pi, z, v) \to \{0, 1\}$. $\mathsf{PC.Verify}$ outputs 1 if and only if $e(C - vG, H) = e(\pi, \tau H - zH)$.

Remark 6.3. Note that $\mathsf{PC.Open}$ aborts when the query is in $\mathbb{H}^{2^{n-k}}$. We argue that it will not affect soundness very much. Actually, it forces challenges to be chosen from $\mathbb{F}^*\backslash\mathbb{H}$, which will cause the denominator of soundness to be $|\mathbb{F}^*| - |\mathbb{H}|$, which is still negligible.

Since the outputs of $\mathsf{PC.Com}$ and $\mathsf{PC.Open}$ are $\mathbf{P}(\tau)G$ and $\mathbf{w}(\tau)G$ both in standard KZG commitment scheme and our modification, the completeness directly follows the standard KZG.
To prove the time complexity of our commitment scheme, we first introduce an algorithm for batched inversions of field elements.

Construction 6 ([BZ10],Section 2.5.1). *For field elements $a_1, ..., a_N$, we first compute $b_j = \prod_{i=1}^{j} a_i$ and let $b_0 = 1$. Then we compute $(b_N)^{-1}$ using an extended Euclidean algorithm. Since $(a_i)^{-1} = b_{i-1} \cdot (b_i)^{-1}$ and $(b_{i-1})^{-1} = (b_i)^{-1} \cdot a_i$, it is trivial to run a loop from N to 1 to obtain all inversions of $a_1, ..., a_N$. The algorithm costs $O(N)$ field multiplications and 1 field inversion for N elements.*

[4] Barycentric evaluation

Lemma 6.4. *Let $N = 2^n$. PC.Setup runs in $O(N \log N)$ time. PC.Com and PC.Open run in $O(N)$ time.*

Proof. We first prove the running time of PC.Setup. We only consider $G_{n,\cdot}$ and compute others in the same way. Let $N = 2^n$. To calculate $G_{n,\cdot}$ on $\mathbb{H} = \{1, \omega, ..., \omega^{N-1}\}$, PC.Setup first computes $\tau^i G$ for $i = 0, ..., 2^n - 1$ as in the standard KZG commitment. Then PC.Setup constructs a polynomial $\mathbf{f} : \mathbb{F} \to \mathbb{G}_1$ $\mathbf{f}(x) = \frac{1}{N} \sum_{k=0}^{N-1} \tau^k x^k G$. Observing that

$$\mathbf{f}(\omega^i) = \frac{1}{N} \sum_{k=0}^{N-1} (\tau\omega^i)^k G = \frac{1}{N} \frac{(\omega^i \tau)^N - 1}{\omega^i \tau - 1} G = \frac{1}{N} \frac{\tau^N - 1}{\omega^i \tau - 1} G = \frac{1}{N} \frac{\tau^N - 1}{\tau - \omega^{N-i}} \omega^{N-i} G = G_{n,N-i}$$

for every $\omega^i \in \mathbb{H}$. Thus, we can apply FFT to the polynomial $\mathbf{f}$ on $\mathbb{H}$ to obtain the vector Σ. So PC.Setup costs $O(\sum_{i=0}^{n} i \cdot 2^i) = O(N \log N)$ time.
It is trivial that PC.Com runs in $O(N)$ time. To prove that PC.Open runs in $O(N)$ time, the only thing we need is to compute $\frac{1}{z - \omega^{2^{n-k} \cdot j}}$ in $O(1)$ time. Since there is an amortized $O(1)$ algorithm for batch inversions, PC.Open runs in $O(N)$ time. $\square$

Remark 6.5 (Batching proofs)**.** In Remark 5.10, we combine all equations except for the two in Construction 3 and Construction 5. Thanks to the additive property of KZG commitment, prover and verifier can sum up the commitments in the protocol to get an equation for only 4 polynomials with their commitments. Along with the equation in Construction 3 and Construction 5, which needs 4 polynomial openings, prover only needs to open 8 polynomials at 8 points, respectively.

6.3 Achieving Knowledge Soundness

By the proof of Lemma 6.4, we can observe that the vector Σ in our modification can be computed by Σ in the standard KZG commitment and vice versa. Thus the extractability of the modification can be inferred from standard KZG commitment [CHMMVW20; BCHO22], since the extractor can first compute the transcript of standard KZG commitment from our modification, and adversaries in our modification cannot break the extractability, otherwise adversaries of standard KZG commitment can break the extractability by calling former adversaries as oracles. The update of Σ can also be derived from the standard KZG commitment with an additional $O(\log N)$ overhead by FFT.
By **Section 7** of [CHMMVW20], since Construction 5 has knowledge soundness and our commitment has extractability, a pre-processing SNARK for the Hadamard product can be constructed.

References

[BBBPWM18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. “Bulletproofs: Short Proofs for Confidential Transactions and More”. In: *Proceedings of the 39th IEEE Symposium on Security and Privacy*. S&P ’18. 2018, pp. 315–334.

[BCCGP16] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. “Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting”. In: *Proceedings of the 35th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT ’16. 2016, pp. 327–357.

[BCG20] J. Bootle, A. Chiesa, and J. Groth. “Linear-Time Arguments with Sublinear Verification from Tensor Codes”. In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC ’20. 2020, pp. 19–46.

[BCGGHJ17] J. Bootle, A. Cerulli, E. Ghadafi, J. Groth, M. Hajiabadi, and S. K. Jakobsen. "Linear-Time Zero-Knowledge Proofs for Arithmetic Circuit Satisfiability". In: *Proceedings of the 23rd International Conference on the Theory and Applications of Cryptology and Information Security*. ASIACRYPT '17. 2017, pp. 336–365.

[BCHO22] J. Bootle, A. Chiesa, Y. Hu, and M. Orru. "Gemini: Elastic SNARKs for diverse environments". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '22. 2022, pp. 427–457.

[BCL22] J. Bootle, A. Chiesa, and S. Liu. "Zero-Knowledge Succinct Arguments with a Linear-Time Prover". In: *Proceedings of the 42nd Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '22. 2022.

[BCRSVW19] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. "Aurora: Transparent Succinct Arguments for R1CS". In: *Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '19. 2019, pp. 103–128.

[BFS20] B. Bünz, B. Fisch, and A. Szepieniec. "Transparent SNARKs from DARK Compilers". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020, pp. 677–706.

[BMMTV21] B. Bünz, M. Maller, P. Mishra, N. Tyagi, and P. Vesely. "Proofs for Inner Pairing Products and Applications". In: *Proceedings of the 27th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT '21. 2021, pp. 65–97.

[BZ10] R. P. Brent and P. Zimmermann. *Modern Computer Arithmetic (version 0.5.1)*. ArXiv abs/1004.4710. 2010.

[CBBZ23] B. Chen, B. Bünz, D. Boneh, and Z. Zhang. "HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates". In: *Proceedings of the 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '23. 2023, pp. 499–530.

[CHMMVW20] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020, pp. 738–768.

[DRZ20] V. Daza, C. Ràfols, and A. Zacharakis. "Updateable Inner Product Argument with Logarithmic Verifier and Applications". In: *Proceedings of the 23rd International Conference on Practice and Theory of Public-Key Cryptography*. PKC '20. 2020, pp. 527–557.

[GGPR13] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. "Quadratic Span Programs and Succinct NIZKs without PCPs". In: *Proceedings of the 32nd Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '13. 2013, pp. 626–645.

[GWC19] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Report 2019/953. 2019.

[ILV11] M. Izabachène, B. Libert, and D. Vergnaud. "Block-Wise P-Signatures and Non-interactive Anonymous Credentials with Efficient Attributes". In: *Cryptography and Coding - 13th IMA International Conference, IMACC 2011, Oxford, UK, December 12-15, 2011. Proceedings*. Lecture Notes in Computer Science. Springer, 2011, pp. 431–450.

[KMP20] A. Kothapalli, E. Masserova, and B. Parno. *A Direct Construction for Asymptotically Optimal zkSNARKs*. Cryptology ePrint Archive, Report 2020/1318. 2020.

[KZG10] A. Kate, G. M. Zaverucha, and I. Goldberg. "Constant-Size Commitments to Polynomials and Their Applications". In: *Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT '10. 2010, pp. 177–194.

[Lee20] J. Lee. *Dory: Efficient, Transparent arguments for Generalised Inner Products and Polynomial Commitments*. Cryptology ePrint Archive, Report 2020/1274. 2020.

[LSZ22] H. Lipmaa, J. Siim, and M. Zajac. "Counting Vampires: From Univariate Sumcheck to Updatable ZK-SNARK". In: *Proceedings of the 28th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT '22. 2022, pp. 249–278.

[MBKM19] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. "Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updateable Structured Reference Strings". In: *Proceedings of the 26th ACM Conference on Computer and Communications Security*. CCS '19. 2019.

[RR22] N. Ron-Zewi and R. D. Rothblum. "Proving as Fast as Computing: Succinct Arguments with Constant Prover Overhead". In: *Proceedings of the 54th Annual ACM Symposium on Theory of Computing*. STOC '22. 2022.

[RZ21] C. Ràfols and A. Zapico. "An Algebraic Framework for Universal and Updatable SNARKs". In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO '21. 2021, pp. 774–804.

[Set20] S. T. V. Setty. "Spartan: Efficient and General-Purpose zkSNARKs Without Trusted Setup". In: *40th Annual International Cryptology Conference*. CRYPTO '20. 2020, pp. 704–737.

[SZ22] A. Szepieniec and Y. Zhang. "Polynomial IOPs for Linear Algebra Relations". In: *Proceedings of the 25th IACR International Conference on Practice and Theory of Public-Key Cryptography*. PKC '22. 2022, pp. 523–552.

[TABDFK20] A. Tomescu, I. Abraham, V. Buterin, J. Drake, D. Feist, and D. Khovratovich. "Aggregatable Subvector Commitments for Stateless Cryptocurrencies". In: *Proceedings of the 12th International Conference on Security and Cryptography for Networks*. SCN '20. 2020, pp. 45–64.

[XZZPS19] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song. "Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation". In: *Proceedings of the 39th Annual International Cryptology Conference*. CRYPTO '19. 2019, pp. 733–764.

[Zha+21] J. Zhang et al. "Doubly Efficient Interactive Proofs for General Arithmetic Circuits with Linear Prover Time". In: *Proceedings of the 28th ACM Conference on Computer and Communications Security*. CCS '21. 2021, pp. 159–177.

[ZSZSWG22] Y. Zhang, A. Szepieniec, R. Zhang, S. Sun, G. Wang, and D. Gu. "VOProof: Efficient zkSNARKs from Vector Oracle Compilers". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. CCS '22. 2022, pp. 3195–3208.