Dual Polynomial Commitment Schemes and Applications to Commit-and-Prove SNARKs

Chaya Ganesh[1], Vineet Nair[2], and Ashish Sharma[2]

[1]Indian Institute of Science
[2]Arithmic Labs
chaya@iisc.ac.in, vineet@arithmic.com, ashish@arithmic.com

June 12, 2024

Abstract

We introduce a primitive called a dual polynomial commitment scheme that allows linking together a witness committed to using a univariate polynomial commitment scheme with a witness inside a multilinear polynomial commitment scheme. This yields commit-and-prove (CP) SNARKs with the flexibility of going back and forth between univariate and multilinear encodings of witnesses. This is in contrast to existing CP frameworks that assume compatible polynomial commitment schemes between different component proofs systems. In addition to application to CP, we also show that our notion yields a version of Spartan with better proof size and verification complexity, at the cost of a more expensive prover.

We achieve this via a combination of the following technical contributions: (i) we construct a new univariate commitment scheme in the updatable SRS setting that has better prover complexity than KZG (ii) we construct a new multilinear commitment scheme in the updatable setting that is compatible for linking with our univariate scheme (iii) we construct an argument of knowledge to prove a given linear relationship between two witnesses committed using a two-tiered commitment scheme (Pedersen+AFG) using Dory as a black-box. These constructions are of independent interest.

We implement our commitment schemes and report on performance. We also implement the version of Spartan with our dual polynomial commitment scheme and demonstrate that it outperforms Spartan in proof size and verification complexity.

1 Introduction

Zero-knowledge proofs and argument systems (ZK) [GMR85] allow proving that a statement is valid without revealing any additional information. Zero-knowledge *Succinct Non-interactive ARguments of Knowledge* (zk-SNARKs), are non-interactive ZK arguments with the additional property that the size of the proof, and verifier work to check the proof is sublinear in the size of the statement. zk-SNARKs are a fundamental building block in modern cryptographic systems where it is crucial that the verification time does not scale with the size of the computation.

Polynomial Commitment Scheme (PCS). At a high level, a PCS enables a prover to initially commit to a polynomial f of bounded degree. Later, the prover can reveal evaluations of f at chosen points, accompanied by proofs verifying that the disclosed values align with the original commitment. A PCS is a central cryptographic tool used to obtain a SNARK in a modular way. A SNARK resulting from compiling an information-theoretic protocol inherits the complexity of the PCS; that is, the proof size depends on the commitment size and evaluation proof size of the PCS.

Commit-and-prove SNARKs (CP-SNARKs). An important family of SNARKs is one with a commit-and-prove extension, called a *commit-and-prove* SNARKs (CP-SNARKs) [CFQ19] where the inputs are

separately committed to. A CP-SNARK allows verification of a proof through this commitment, that, crucially, can be reused across proofs. The presence of these commitments allow to glue together different proof systems that use parts of same witness. CP-SNARKs are useful in a variety of applications where one needs to prove composite statements using the most efficient tool for each part of the statement. CP-SNARKs allow modularity of proof systems thus providing *interoperability* with different protocols specialized for efficiently proving certain class of relations. For instance, consider a "mixed" computation that naturally presents different components, like Boolean/arithmetic circuit for a hash function, and algebraic representation for group operations. Using a general-purpose zkSNARK for such a computation requires one homogeneous intermediate representation of this computation. This incurs a high cost in performance; for example, writing a modular exponentiation as a circuit requires number of gates that grows with the size of the modulus. A CP-SNARK takes advantage of the native representation of different parts of the computation and does a mix-and-match of the best proof system for each component, e.g., SNARKs for an arithmetic circuit and a Sigma-protocol for an algebraic relation.

Existing CP-SNARK frameworks assume compatibility of the cryptographic compilers used in the different proof components. If one information theoretic proof component is compiled using a PCS, then the linking proofs are designed to be compatible with the representation of the polynomials (vectors of coefficients/ vectors of evaluations) used by the polynomial commitment scheme. That is, the linking that is essential for a CP-SNARK works as long as the PCS match up in how they interpret the polynomial (univariate vs multivariate, vector of coefficients vs vector of evaluations etc.). What if we want a CP-SNARK where one component uses a polynomial represented as a vector of coefficients and committed to using a univariate PCS (like KZG [KZG10]), and another component represents its polynomials as a vector of evaluations and are committed to using a multilinear scheme (like PST [PST13] or Hyrax [WTS+18]) and these polynomials encode the same shared witness?

We put forth the notion of a *dual polynomial commitment scheme*, that links univariate and multilinear PCS. Specifically, a dual polynomial commitment scheme can be used to prove evaluations of a univariate and a multilinear polynomial derived from the *same* witness. Towards this, we construct an efficient linking proof for connecting a witness committed to using a univariate polynomial commitment scheme with a witness inside a multivariate polynomial commitment scheme. The dual polynomial commitment scheme allows one to go back and forth between univariate and multilinear encodings of witnesses. To further motivate the need for such flexibility, we now outline some example applications.

1.1 Applications

Commit-and-prove Lookup. There exist general compilers [CFF+21, ABC+22] that take an information-theoretic proof, like an Algebraic Holographic Proof (AHP) or Polynomial Interactive Oracle Proof (PIOP) and compile them into a CP-SNARK using a cryptographic compiler like a PCS. Concrete instantiations of these compilers yield frameworks that can glue together general purpose SNARKS for an arithmetic circuit/Rank 1 constraint system (R1CS),like Marlin/Sonic/Plonk with proof systems for algebraic statements, like Sigma protocols. However, the above works do not directly handle *lookup arguments*. They can be extended in a straightforward way to lookups, but they are not general since they are limited to lookup arguments that use the same PCS as the compiler or a "compatible" one. For example, the Lasso [STW23] lookup argument can be extended to give CP-lookup but can only be linked to Spartan [Set20]; and Plookup [GW20] can be extended to be CP-lookup linked to Plonk [GWC19]. We briefly discuss why lookup arguments are useful and it is desirable to have general CP-SNARKs that allow linking to lookups. A lookup argument, at a high level, allows a prover to convince a verifier that $v_i = t_{u_i}$ for all i given committed vectors $\vec{t}, \vec{u}$ and $\vec{v}$. Circuit based representations are inefficient in expressing certain relations, like bit-decomposition. Lookups are used as custom gates in SNARKs (for instance plookup[GW20], Arya[BCG+18]) where "SNARK-unfriendly" operations like bit-decomposition, range proof etc. are performed via a table lookup instead of a circuit representation using addition/multiplication gates. This rich and growing body of work on lookup arguments culminated in the notion of lookup singularity[Whi] where SNARK front ends produce circuit representations consisting of *only* lookups. Lasso [STW23] and Jolt [AST23] together achieve this lookup singularity: (i) a circuit that executes an instruction at each step (SNARK-unfriendly gates like OR, XOR, AND etc)

is modeled as a single lookup into a large lookup table (ii) efficient lookup into large tables via table decomposition. Jolt uses Lasso for step (ii) which uses sumcheck arguments and hence requires a multilinear PCS. Thus, while Lasso naturally admits commit-and-prove, this is restricted to SNARKs that encode the witness in a multivariate PCS to match the PCS used in the lookup argument of Lasso. In the spirit of commit-and-prove, our goal is to include lookup as a gadget in the toolbox of CP-SNARKs and modularly build complex schemes by mixing and matching lookup with other widely deployed circuit-based SNARKs. Via our dual polynomial commitment scheme, we can construct a CP-SNARK that combines general purpose SNARKs with lookup arguments, regardless of how the underlying PCS work (univariate/multilinear).

Let us consider the following scenario: The verifier would like to check that $a_1, \ldots, a_m$ which are certain intermediate values in a computation are all in a large range, like, in $\{0, 1, \ldots, 2^{32} - 1\}$. Checking m values are in the range $< 2^N$ requires $O(mN)$ R1CS constraints on an $O(mN)$ length witness. Using STARK, this incurs a cost of at least $3m$ FFTs of length $O(N)$ resulting in a prover complexity of $O(mN \log N)$ field operations. This cost stems from expensive operations involving bit-decomposition due to the range checks inside the circuit. This cost is avoided by moving the range checks "outside" the SNARK circuit to lookups. Now, there is a need to "tie" the $a_1, \ldots, a_m$ that were used in the lookup to the intermediate values of the computation. That is, given commitments to $a_1, \ldots, a_m$, the lookup argument proves each a_i is in the desired range, a general-purpose zkSNARK proves the computation, and this combination is sound only if the verifier is convinced that the each a_i used in the lookup argument is indeed the output of the sub-computation up to the range check and the input to the sub-computation after the range check. This gluing step is straightforward if the polynomial commitment scheme used in the lookup argument and the rest of the SNARK is the same (like Lasso [STW23] and Spartan [Set20], or Caulk [ZBK+22] and Plonk [GWC19]). This falls short of the goal of commit-and-prove which is generality: plug-and-play different gadgets for different sub-computations. In particular, if one were to use a SNARK like Groth16 (based on KZG, a univariate PCS) and a lookup argument like Lasso (based on multilinear PCS), then the approach described above will not work as-is. Our constructions overcome the technical difficulty in establishing consistency of witnesses used in the SNARK and the lookup argument encoded via a univariate PCS and a multilinear PCS respectively.

No Of Constraints	Eval Prover (sec)		Eval Verifier (sec)		Proof size (KB)	
	Spartan	Spartan AIR	Spartan	Spartan AIR	Spartan	Spartan AIR
2^{16}	17.37	97.76	0.97	0.58	69.28	35.64
2^{18}	32.55	936.24	1.22	0.62	82.16	39.07

Table 1: Metrics comparing Spartan and Spartan-AIR. Spartan denotes Spartan with grand-product check using [GKR08, Tha13], and Spartan AIR denotes spartan with grand-product check using the AoK from Section 7. The ratio of sparsity to constraints in the R1CS matrices is maintained to one.

Grand Product and Efficient Spartan. We design a new special-purpose proof system for proving grand product relations that could be of independent interest. One concrete benefit of our grand product argument is that it yields an efficient version of Spartan obtained as a CP-SNARK using our dual polynomial commitment scheme. The polynomials that encode the intermediate computation states of the grand product computation are committed to using our univariate PCS KZG-FFT. This grand product argument can be put together with the rest of the Spartan proof that uses a multilinear PCS relying on the linking soundness guarantee of the dual PCS. The resulting SNARK, Spartan AIR, has concretely better proof complexity and verifier complexity, albeit at the cost of worse prover complexity (Table 1). We believe this is a worthwhile trade-off in some applications. One example is the use of a folding-based recursive proof system like Nova [KST22] on the blockchain. Here, most of the time, the system does folding based recursion, and a layer-1 verifier checks the final folded instances which are only infrequently proven after many folding steps. When Nova's folding scheme is applied to Spartan, the large proof-sizes and verifier complexity requires wrapping the Spartan verifier of the relaxed instances inside a more verifier-friendly SNARK like Groth16 [Gro16]. While Groth16 is verifier efficient, it's SRS is circuit dependent and not updatable. In contrast, using our approach, we can obtain better proof size and verifier complexity directly

for the Spartan proof while retaining transparent SRS.

zkRollups. Combining lookups with other SNARKs is also relevant in the context of *rollups*. Blockchain rollups are a scaling solution that move expensive computation off-chain to layer two chains. Here, the network participants only need to verify succinct proofs attesting to the correctness of the off-chain computation. This approach allows verifying the L2 state resulting from several rolled up transactions as part of *one transaction* verified on the main chain. Commit-and-prove provides better rollup solutions by allowing signature verification to be done using suitable SNARKs and checking validity of transactions using table lookups. While this is an important general application of commit-and-prove lookup, we do not provide details of an end-to-end rollup solution in this work.

1.2 Our Contributions

- We propose a univariate polynomial commitment scheme KZG-FFT where the prover need not perform expensive FFT operations. It is variant of the KZG scheme where we commit to the vector of evaluations over the FFT domain instead of the coefficient vector. Interpolating the witness vector to obtain evaluations is not done by the prover, instead the setup phase creates the SRS after applying the corresponding linear transformation. We then show how to make the SRS universal and updatable.

- We propose a new multilinear polynomial commitment scheme KZG-FOURIER that is suited for linking with our univariate scheme KZG-FFT.

- We define the notion of a *dual polynomial commitment scheme* that allows committing to a witness both as a univariate and a multilinear polynomial. We provide two candidate instantiations of dual polynomial commitment scheme, one in the transparent setting and another in the updatable SRS setting. We implement our schemes in Rust and show setup/prover/verifier time for growing witness sizes.

- We present an argument of knowledge to prove a given linear relationship between two witnesses committed using a two-tiered commitment scheme (Pedersen+AFG) that uses Dory as a black-box. We use this argument in our instantiation of a transparent dual polynomial commitment scheme. In our instantiation, we preprocess an FFT matrix that determines the linear relationship we need for our dual polynomial commitment.

- We construct a new argument of knowledge for proving grand product relations. This is an instantiation of existing proof system using our KZG-FFT commitment scheme as the polynomial commitment scheme. However, our dual polynomial commitment scheme enables us to use the grand product argument with Spartan or Lasso yielding smaller proof sizes. We report implementation results of performance of the version of Spartan using our grand product argument.

1.3 Technical Overview

Univariate scheme KZG-FFT. We begin by outlining the ideas behind the KZG PCS [KZG10] which is our starting point. The KZG scheme works as follows: the commitment to a univariate polynomial $f(X)$ is the encoding of the evaluation of the polynomial at a secret point. The encoding used is exponentiation in a bilinear group, which makes the commitment one group element: $C := g^{f(\tau)}$ In order to enable the prover to compute this, the setup produces a structured reference string (srs) consisting of encodings of powers of the secret point τ. Now the prover can use the additive homomorphism of the encoding to compute $g^f(\tau)$ without knowing τ using the coefficient vector of f and $\{g^{\tau^i}\}$. Now to provably open the committed polynomial at a random point z, the prover produces a proof for the claim $f(z) = v$ as follows. It computes the quotient polynomial $q(X) := (f(X) - v)/X - z$ and provides a commitment to $q(X)$ as the proof; $\pi = g^{q(\tau)}$. The verifier now checks the polynomial division by checking the above equation at the random point τ. The verifier uses the bilinear map to check $e(C \cdot g^{-v}, g) = e(\pi, g^{\tau} \cdot g^{-z})$. Now, the way typical PIOPs (underlying SNARKs like Marlin/Plonk) encode the witness is by interpolating a polynomial

f such that the witness vector agrees with f on a nice domain. That is, let $w \in \mathbb{F}^n$ be the witness vector, then the prover first constructs a *witness-carrying polynomial* $f(X) = \sum_{i\in[n]} w_i L_i(X)$ where $L_i(X)$ are the Lagrange bases, and the "nice domain" is typically the multiplicative subgroup generated by primitive root of unity. The univariate polynomial $f(X)$ is then committed to using a PCS like KZG. Therefore, the prover is required to perform FFT operations in order to obtain $f(X)$ in the required coefficient representation. Our first idea is to move the costly FFT to the setup phase: instead of the setup string consisting of encoding of powers of τ, the setup now consists of powers of α such that α and τ are related via the FFT matrix.

The commitment is still evaluation of the polynomial at a secret point (just like in KZG), but the prover need not perform expensive operations since this work has already been done apriori by setup. The prover uses the witness vector w and commits to it as $g^{f(\tau)}$ where f agrees with w without having to explicitly obtain the coefficients of f, for $\tau = r^{2^n}$, for a uniformly random $r \in \mathbb{F}$, and n is the degree. The downside now is that the srs is not universal since it depends on the degree of the polynomial being committed to. In order to achieve universality, we let the commitment be $g^{f(\tau)}$ for $\tau = r^{2^{n-d}}$ where n is the bound on the universal srs and d is the degree bound claimed by the prover. The idea behind the evaluation protocol for the PCS remains the same as in KZG: commit to the quotient polynomial and check polynomial division at τ. We show that this scheme satisfies extractability in the AGM assuming N-DLOG. We note that the scheme also satisfies the *updatable* srs property which is more desirable than a fully trusted setup.

Multilinear scheme $\mathsf{KZG\text{-}FOURIER}$. Our new multilinear PCS relies on a new homomorphic map and polynomial decomposition. First, we map the Fourier basis of a multilinear polynomial in n variables to the FFT basis of a univariate polynomial of degree 2^n. The commit algorithm interprets the witness vector as evaluations of a multilinear polynomial f over the Fourier basis, and equivalently as the evaluations of a univariate polynomial f' over the FFT domain of size 2^n. Committing to f is done by committing to f' using $\mathsf{KZG\text{-}FFT}$. The evaluation protocol relies on polynomial decomposition that has been used in prior works as well [PST13]. For an n-variate polynomial f, and $\mathbf{z} \in \mathbb{F}^n$, there exist polynomials $q_i(\mathbf{x})$ such that, $f(\mathbf{x}) - f(\mathbf{z}) = \sum_{i\in[n]} (x_i - z_i) q_i(\mathbf{x})$. Ignoring the technical details that we explain in Section 4, in the evaluation protocol, the verifier checks this polynomial identity but underneath the linear map, so it reduces to checking this in the univariate case.

1.4 Related Work

CP-SNARKs. Recent works like Lunar [CFF+21] and Eclipse [ABC+22] present general compilers from information-theoretic objects to CP-SNARKs with a universal and updatable SRS. While the underlying information-theoretic objects can be compiled using any PCS, Eclipse assume Pedersen vector commitments and Lunar assumes KZG commitments for the linking gadgets of the CP-SNARKs. Thus they are limited as a framework to only those proof systems that use compatible commitment schemes. These compilers can be extended to obtain commit-and-prove lookups, but will be limited to lookup arguments like Caulk [ZBK+22], Baloo [ZGK+22], CQ [EFG22] that are KZG-based.

PCS. Our multilinear PCS is reminiscent of the techniques in [BCHO22] and [KT23] in that the central idea is to leverage a linear homomorphism from the $\mathbb{F}$-linear space of multilinear polynomials to the $\mathbb{F}$-linear space of univariate polynomials. The difference is the linear homomorphism itself: in [KT23], the Fourier basis of the multilinear polynomial is mapped to the monomial basis of the univariate polynomial. In [BCHO22], the monomial basis of the multilinear polynomial is mapped to the monomial basis of the univariate polynomial. In our scheme, the Fourier basis of the multilinear is mapped to the Fourier basis of the univariate polynomial. As a consequence, our multilinear PCS renders itself to be efficiently linked to our $\mathsf{KZG\text{-}FFT}$ scheme.

2 Preliminaries

Notations and Facts. Throughout $\mathbb{F}$ denotes a prime field of order p, and $\mathbb{N}$ denotes the set of natural numbers. We denote by λ a security parameter, by negl a negligible function. For any integer $c > 0$, there exists $n \in \mathbb{N}$, such that $\forall\ x > n$, $\mathsf{negl}(x) \leq 1/n^c$, and if a function is not negligible then we call it non-negligible.

We denote vectors by boldface letters, and inner product between $\mathbf{a}$ and $\mathbf{b}$ by $\langle \mathbf{a}, \mathbf{b} \rangle$. If $\mathbf{a} \in \mathbb{F}^N$, then a_i denotes the i-th component of the vector for $i \in [0, N-1]$. We use $r \in_R \mathbb{F}$ to denote r sampled independently and uniformly at random from $\mathbb{F}$. Primitive root of unity of order N is denoted by ω_N. We state a well-known fact regarding primitive roots of unity next.

Fact 1. *For $N \in \mathbb{N}$, if N is divisible by 2 then $\omega_{N/2} = \omega_N^2$. Further, ω_2, is unique and is equal to -1.*

The $N \times N$ Fast Fourier Transform (FFT) matrix whose (i,j)-th entry is $\omega_N^{i \cdot j}$ for $i, j \in [0, N-1]$ is denoted M_{ω_N}. We note down a well-known fact regarding the FFT matrices.

Fact 2. *For $N \in \mathbb{N}$, $M_{\omega_N}^T = M_{\omega_N}$, and further if N is a power of 2 then $M_{\omega_N}^{-1} = \frac{1}{N} M_{\omega_N^{-1}}$*

We use $\mathbf{H}_N$ to denote the set $\{\omega_N^0, \omega_N^1, \ldots, \omega_N^{N-1}\}$. We refer to $\mathbf{H}_N$ as the FFT domain of size N. Similarly, let $p' \in \mathbb{F}$ be such that p, p' are coprime integers. Then we refer to the set $\{p' \cdot \omega_N^0, p' \cdot \omega_N^1, \ldots, p' \cdot \omega_N^{N-1}\}$ as the offset of the FFT domain of size N. We have the following easy to see fact.

Fact 3. *For z in the offset of the FFT domain of size N, $\prod_{i \in [0,N-1]} (z - \omega_{2N}^i) \neq 0$.*

$\mathbb{F}_{<N}[Y]$ denotes the $\mathbb{F}$-linear space of univariate polynomials with degree at most N, and $\{Y^0, \ldots, Y^{N-1}\}$ is its standard basis. Let $f \in \mathbb{F}_{<N}[Y]$. Then $f(\mathbf{H}_N) \in \mathbb{F}^N$ denotes the evaluation vector of $f(Y)$ over $\mathbf{H}_N$, that is the i-th coordinate of $f(\mathbf{H}_N)$, denotes $f(\mathbf{H}_N)_i = f(\omega_N^i)$ for $i \in [0, N-1]$. If N is clear from the context then we drop N from the subscript and simply write $f(\mathbf{H})$. If $\mathbf{a} \in \mathbb{F}^N$ and $f \in \mathbb{F}_{<N}[Y]$ satisfies $\mathbf{a} = f(\mathbf{H})$ then we say $\mathbf{a}$ agrees with f over the FFT domain of size N. Let $\mathbf{Y} = (Y^0, \ldots, Y^{N-1})$ be an N length vector constituting of the standard basis of $\mathbb{F}_{<N}[Y]$, and $\mathbf{U}^{(n)} = \frac{1}{N} \cdot M_{\omega_N^{-1}} \cdot \mathbf{Y}$, where $n = \log N$. Since $M_{w_N^{-1}}$ is invertible, $\{U_i^{(n)}\}_{i \in [0,n-1]}$ is an $\mathbb{F}$-linear basis of $\mathbb{F}_{<N}[Y]$, where U_i is the i-th component of $\mathbf{U}^{(n)}$. We refer to $\mathbf{U}^{(n)}$ as the FFT basis of $\mathbb{F}_{<N}[Y]$.

Similarly, $\mathbb{F}_{\leq 1}[X_0, \ldots, X_{n-1}]$ denotes the $\mathbb{F}$-linear space of multilinear polynomials in n variables, and $\{L_0^{(n)}, \ldots, L_{2^n-1}^{(n)}\}$ be the standard Fourier basis of $\mathbb{F}_{\leq 1}[X_0, \ldots, X_{n-1}]$. Specifically, if $i \in [0, 2^n - 1]$, and $i_j \in \{0,1\}$ for $j \in [0, n-1]$ is the j-th least significant bit of i (when i is viewed as an n bit number) then

$$L_i^{(n)} = \prod_{j=0}^{n-1} ((1 - X_j) \cdot (1 - i_j) + X_j \cdot i_j)$$

Let $f \in \mathbb{F}_{\leq 1}[X_0, \ldots, X_{n-1}]$, and $f(X_0, \ldots, X_{n-1}) = \sum_{i \in [0, 2^n - 1]} f_{L_i^{(n)}} \cdot L_i^{(n)}(X_0, \ldots, X_{n-1})$. Then $\{f_{L_i^{(n)}} \in \mathbb{F}\}_{i \in [0, 2^{n-1}]}$ are referred to as the 2^n Fourier coefficients of f. Let $\mathbf{a} \in \mathbb{F}^{2^n}$. Then $\tilde{a} \in \mathbb{F}[X_0, \ldots, X_{n-1}]$ defined as $\tilde{a}(X_0, \ldots, X_{n-1}) = \sum_{i \in [0, 2^n - 1]} a_i \cdot L_i(X_0, \ldots, X_{n-1})$ denotes the Multilinear Extension (MLE) of $\mathbf{a}$. It is easy to see that $\tilde{a}(i_0, \ldots, i_{n-1}) = a_i$, where again $i_j \in \{0,1\}$ for $j \in [0, n-1]$ is the j-th least significant bit of i. We note the following well-known fact regarding multilinear polynomials (see [PST13] for a proof).

Fact 4. *Let $f \in \mathbb{F}_{\leq 1}[X_0, \ldots, X_{n-1}]$. Then $f(x_0, \ldots, x_{n-1}) = y$ if and only if there exists $q_k \in \mathbb{F}_{\leq 1}[X_0, \ldots, X_{k-1}]$ for $k \in [1, n-1]$, and $q_0 \in \mathbb{F}$ such that*

$$f(X_0, \ldots, X_{n-1}) - f(x_0, \ldots, x_{n-1}) = \sum_{k=0}^{n-1} (X_k - x_k) \cdot q_k(X_0, \ldots, X_{n-1})$$

Let $\mathbb{G}$ be a group of order p. For $\mathbf{g} = (g_1, \ldots, g_n) \in \mathbb{G}^n$ and $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$, the multi-exponentiation $\mathbf{g}^{\mathbf{x}}$ is defined by $\mathbf{g}^{\mathbf{x}} = g_1^{x_1} \cdots g_n^{x_n}$.

Bilinear Group. A bilinear group is denoted by the tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order p, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable non-degenerate bilinear map. We denote by $\mathcal{G}$ a bilinear group generator that outputs these parameters: $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow_R \mathcal{G}(1^\lambda)$ where p is superpolynomial in λ. Let $\mathbf{h} \in \mathbb{G}_1^N$, $\mathbf{q} \in \mathbb{G}_2^N$. Then we use the $\langle \mathbf{h}, \mathbf{q} \rangle$ to denote the inner-pairing product $\prod_{i \in [0, N-1]} e(h_i, q_i)$. We remark here that notation $\langle \cdot, \cdot \rangle$ is overloaded to mean both inner products over fields, and inner-pairing products between vectors from $\mathbb{G}_1$ and $\mathbb{G}_2$, and is implicit from context.

Algebraic Group Model. The Algebraic Group Model (AGM) introduced in [FKL18] is an idealized model. An adversary $\mathcal{A}$ is said to be algebraic if every group element output by $\mathcal{A}$ is accompanied by its representation with respect to all the groups elements $\mathcal{A}$ has seen so far. Let $y_1, \ldots, y_k$ be all the group elements previously input and output by $\mathcal{A}$. Then, every group element y output by $\mathcal{A}$, is accompanied by its representation $(x_1, \ldots, x_k)$ such that $y = \prod_{i=1}^k y_i^{x_i}$.

SRS model. We describe our constructions as public-coin interactive protocols in the structured reference string (SRS) model where where both the parties have access to a SRS. The SRS is universal and updatable, where the SRS can be used to prove statements about any computation, as opposed to a circuit-dependent setup required in preprocessing based SNARKs. This universal SRS is, in addition, *updatable*, meaning parties can continuously contribute to the randomness of the SRS, and an SRS is trusted as long as at least one of the updates was honest.

2.1 Assumptions

Definition 1 (DDH Assumption)**.** *For a group $\mathbb{G}$, the decisional Diffie-Hellman (DDH) assumption holds for $\mathbb{G}$ if for all PPT $\mathcal{A}$, the following probability is $1/2 + \mathsf{negl}(\lambda)$:*

$$\Pr\left(\begin{matrix} b' = b \\ b' = \mathcal{A}(\mathsf{pp}) \end{matrix} \; : \; \begin{matrix} x, y, z \leftarrow_R \mathbb{F}, b \leftarrow_R \{0,1\} \\ z' = xy \text{ if } b = 0, z' = z \text{ if } b = 1 \\ \mathsf{pp} = (g, g^x, g^y, g^{z'}) \end{matrix}\right)$$

Definition 2 (SXDH Assumption)**.** *For $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H) \leftarrow_R \mathcal{G}(1^\lambda)$, the Symmetric External Diffie-Hellman (SXDH) assumption states that the decisional Diffie-Hellman (DDH) assumption holds for both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

Definition 3 (N-DLOG Assumption)**.** *The N-DLOG assumption with respect to $\mathcal{G}$ holds if for all λ, for all PPT $\mathcal{A}$, the following probability is $\mathsf{negl}(\lambda)$:*

$$\Pr\left(\begin{matrix} \tau = \tau' \\ \tau' \leftarrow \mathcal{A}(\mathsf{pp}, \mathbf{v}) \end{matrix} \; : \; \begin{matrix} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ \tau \leftarrow_R \mathbb{F} \\ \mathbf{v} := (g_1^\tau, g_1^{\tau^2}, \ldots, g_1^{\tau^N}, g_2^\tau, g_2^{\tau^2}, \ldots, g_2^{\tau^N}) \end{matrix}\right)$$

2.2 Commitment Schemes

AFG Commitment Scheme [AFG$^+$10] The pairing based commitment AFG is defined as follows:

- $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e, g_1, g_1) \leftarrow \mathsf{AFG.setup}(1^\lambda)$: Outputs $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ of order p with g_1 and g_2 being generators for $\mathbb{G}_1$ and $\mathbb{G}_2$ and e is the bilinear map.
- $\mathsf{ck} = (w_r, w_0, w_1, \ldots, w_{N-1}) \leftarrow \mathsf{AFG.KeyGen}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e, g_1, g_2, N)$: Outputs the commitment key ck. Here N is the number of group elements to be committed.
- $c = \mathsf{AFG.com}_{\mathsf{ck}}(\mathbf{m}, r) = e(w_r, r) \prod_{i=0}^{N-1} e(w_i, m_i)$: Outputs the commitment of message $\mathbf{m}$ where $\mathbf{m} = (m_0, \ldots, m_{N-1})$.

The AFG commitment scheme is perfectly hiding and binding under the SXDH assumption (see Def. 2).

2.3 Interactive Arguments

We consider interactive arguments for relations, where a prover P convinces the verifier that it knows a witness w such that for a public statement $x, (x, w) \in \mathcal{R}$. Given a pair of PPT interactive algorithms P, V, we denote by $\langle P, V\rangle(x; w)$, the output of V on interaction with P. Here, w is P's private input and x is a common input. Let $\mathcal{R} = \{(x, w)\}$, be a relation and $\mathcal{L}$ be the corresponding NP language.

Definition 4 (Succinct Argument of knowledge)**.** *An interactive argument of knowledge (AoK) for a relation $\mathcal{R}$ consists of a PPT algorithm $\mathsf{Setup}(1^\lambda)$ that takes a security parameter λ and outputs public parameters srs, and a pair of PPT interactive algorithms $\langle P, V\rangle$. The triple (Setup, P, V) satisfy the following properties.*

1. *Completeness. For all $\lambda \in \mathbb{N}$, $(x, w) \in \mathcal{R}$,*

$$\Pr\left(\langle P, V\rangle(\mathsf{srs}, x; w) = 1 \,:\, \mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda)\right) = 1.$$

2. *Knowledge Soundness. An argument system (P, V) for a relation $\mathcal{R}$ is knowledge sound with error κ if there exists an expected polynomial time extractor $\mathcal{E}$ such that for every efficient adversary $\tilde{P}$, for every $x \in \{0,1\}^*$, whenever $\tilde{P}$ makes V accept with probability $\epsilon > \kappa$, $\mathcal{E}^{\tilde{P}}(x)$ outputs w^* such that $(x, w^*) \in \mathcal{R}$ with probability at least $\frac{\epsilon-\kappa}{q}$ for some polynomial q.*

3. *Succinctness. An argument system is succinct if the communication complexity between prover and verifier is sublinear in the size of the statement.*

We do not focus on zero-knowledge in this work; we believe it is easily achievable via minor adaptations applying standard techniques to our constructions.
Fiat-Shamir transform. The protocols in this work are *public coin* interactive arguments where the verifier's messages are uniformly random strings. Public coin protocols can heuristically be compiled into non-interactive arguments by applying the Fiat-Shamir [FS87] transform (FS) in the Random Oracle Model.

2.4 Polynomial Commitment Scheme

A polynomial commitment scheme (PCS) allows the prover to open evaluations of the committed polynomial succinctly ([KZG10]). A PCS over $\mathbb{F}$ is a tuple $\mathsf{PC} = (\mathsf{setup}, \mathsf{commit}, \mathsf{open}, \mathsf{eval})$ where:

- $\mathsf{pp} \leftarrow \mathsf{setup}(1^\lambda, n, N)$. On input security parameter λ, number of variables n, an upper bound $N \in \mathbb{N}$ on the number of distinct monomials in n variables, setup generates public parameters pp.
- $(C, \tilde{\mathbf{c}}) \leftarrow \mathsf{commit}(\mathsf{pp}, f, D)$. On input the public parameters pp, an n-variate polynomial $f(X) \in \mathbb{F}[X]$ with total monomials at most $D \leq N$, commit outputs a commitment to the polynomial C, and additionally an opening hint $\tilde{\mathbf{c}}$.
- $b \leftarrow \mathsf{open}(\mathsf{pp}, f(X), d, C, \tilde{\mathbf{c}})$ On input the public parameters pp, the commitment C and the opening hint $\tilde{\mathbf{c}}$, a polynomial $f(X)$ with number of monomials $D \leq N$, open outputs a bit indicating accept or reject.
- $b \leftarrow \mathsf{eval}(\mathsf{pp}, C, D, \mathbf{x}, y; f(X))$. A public coin interactive protocol $\langle P_{\mathsf{eval}}, V_{\mathsf{eval}}\rangle(\mathsf{pp}, C, d, \mathbf{x}, y; f(X))$ between a PPT prover and a PPT verifier. The parties have as common input public parameters pp, commitment C, monomial bound D, evaluation point $\mathbf{x} \in \mathbb{F}^n$, and claimed evaluation y. The prover has, in addition, the opening f of C with number of monomials at most D. At the end of the protocol, the verifier outputs 1 indicating accepting the proof that $f(\mathbf{x}) = y$, or outputs 0 indicating rejecting the proof.

A polynomial commitment scheme must satisfy completeness, binding and extractability.

Definition 5 (Completeness)**.** *For all polynomials $f(X) \in \mathbb{F}[X]$ with number of monomials at most $D \leq N$, and for all $\mathbf{x} \in \mathbb{F}^n$,*

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

Definition 6 (Binding)**.** *A polynomial commitment scheme* PC *is binding if for all PPT $\mathcal{A}$, the following probability is negligible in λ:*

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

Definition 7 (Knowledge soundness)**.** eval *is an AoK for the relation $\mathcal{R}_{\mathsf{eval}}$ defined as follows:*

$$\begin{aligned} \mathcal{R}_{\mathsf{eval}} = \{((\mathsf{pp}, C, \mathbf{x} \leftarrow_R \mathbb{F}^n, \ y \in \mathbb{F}); \ (f(X), \tilde{\mathbf{c}})) : \\ (\mathsf{open}(\mathsf{pp}, f, D, C, \tilde{\mathbf{c}}) = 1) \wedge y = f(\mathbf{x})\} \end{aligned}$$

Though in general, the eval protocol can be run on points $\mathbf{x}$ chosen by the verifier, in applications to SNARKs, these are uniformly random. This is because when a PCS is used to compile a *public-coin* argument, the verifier samples the evaluation points uniformly at random. Therefore, we define knowledge soundness for random $\mathbf{x}$.

Definition 8 (Succinctness)**.** *We require the commitments and the evaluation proofs to be of size* $\mathsf{poly}(\lambda) \cdot \mathsf{log}(d)$ *where π is the transcript obtained by applying FS to* eval*. Additionally, the scheme is* verifier succinct *if* eval *runs in time* $\mathsf{poly}(\lambda) \cdot \mathsf{log}(d)$ *for the verifier. The prover is required to run in time $\tilde{O}(d)$.*

For our univariate scheme, we achieve a strong notion of succinctness: the commitment and evaluation proofs are of size independent of the degree of the polynomial.

3 Univariate Polynomial Commitment Scheme

In this section, we introduce a variant of the KZG commitment scheme known as KZG-FFT. This variant commits directly to the evaluations of a polynomial over the FFT domain of an appropriate size, as opposed to using the coefficients of the corresponding polynomial. In many proof systems, the coefficients of the underlying univariate polynomial are derived through interpolation from a witness vector, which serves as the evaluations of the desired polynomial. By committing directly using these evaluations or witness values, the prover can circumvent the costly FFT operation. Moreover, employing multi-scalar multi-exponentiation (MSME) over the witness vector allows us to leverage the low bit representation of the witness values, resulting in computational advantages. It's worth noting that, in practice, witnesses typically possess low bit complexity despite being represented as elements of large prime fields (see, for instance, [STW23], [AST23]). In Section 3.1, we state the algorithm to generate the setup for KZG-FFT, and in Section 3.2 we state the details of the polynomial commitment scheme.

3.1 Setup Generation

We state the algorithm to generate the structured reference string (srs) in this section. We assume N is a power of 2, and $\log N = n$. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow_R \mathcal{G}(1^\lambda)$ be a bilinear group. The srs has two disjoint parts $(\mathsf{srs}_\mathcal{P}, \mathsf{srs}_\mathcal{V})$: $\mathsf{srs}_\mathcal{P}$ is used by the prover and $\mathsf{srs}_\mathcal{V}$ is used by the verifier. The setup algorithm, stated in Algorithm 1 takes as input the security parameter λ and the degree bound N, and outputs srs and a proof π. The proof π attests to the correctness of the generated srs. We show in Appendix A.3, how the srs can be verified using π, and that the setup generated by KZG-FFT.Setup is updatable as defined in

Algorithm 1 KZG-FFT.Setup(1^λ): Setup Generation for KZG-FFT

Input: $\{1^\lambda, N\}$
Output: $\{\mathsf{srs} = (\mathsf{srs}_\mathcal{P}, \mathsf{srs}_\mathcal{V}), \pi\}$

1: Sample $r \in_R \mathbb{F}$, uniformly at random.
2: Let $\alpha_i = N^{-1} \cdot \prod_{j\in[0,n-1]}(1 + (\omega_N^{-i} \cdot r)^{2^j})$ for $i \in [0, N-1]$.
3: Compute $\mathsf{srs}_\mathcal{P} = \{h_{1,i}^{(n)} = g_1^{\alpha_i} \in \mathbb{G}_1 \mid i \in [0, N-1]\}$, $\mathsf{srs}_\mathcal{V} = \{h_{2,j} = g_2^{r^{2^j}} \in \mathbb{G}_2 \mid j \in [0, n]\}$.
4: Let $\pi = (g_1^r, g_2^r)$.
5: Output $\{\mathsf{srs} = (\mathsf{srs}_\mathcal{P}, \mathsf{srs}_\mathcal{V}), \pi\}$.

[MBKM19]. Let r be as sampled at Step 2, and α_i be as defined at Step 3 of Algorithm 1. Let $\alpha, \mathbf{r} \in \mathbb{F}^N$ be vectors such that α_i and r^i are the i-th component of α and $\mathbf{r}$ respectively for $i \in [0, N-1]$. The following claim shows that α is linearly related to $\mathbf{r}$ via the FFT matrix $M_{\omega_N^{-1}}$.

Claim 3.1. *Let $\alpha, \mathbf{r} \in \mathbb{F}^N$ be as defined above. Then $\alpha = \frac{1}{N} M_{\omega_N^{-1}} \cdot \mathbf{r}$.*

In Section 3.2, we present the commit algorithm for KZG-FFT, and as part of completeness in Lemma 4 argue the structure of the commitment in $\mathbb{G}_1$. The proof of the above claim would follow from proof of Claim A.1 stated in the proof of Lemma 4.

3.2 Protocol

Protocol 1: KZG-FFT polynomial commitment scheme

1. $\{\mathsf{srs}, \pi\} \leftarrow_R \mathsf{KZG\text{-}FFT.setup}(1^\lambda, N)$.
2. $C_f \leftarrow \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_\mathcal{P}, D, f(\mathbf{H}_D))$, where
$$C_f = \prod_{i\in[0,D-1]} (h_{1,i}^{(d)})^{a_i}$$
and $\mathbf{h}^{(d)} \in \mathbb{G}_1^D$ is as defined in Equation 1
3. accept/reject $\leftarrow \mathsf{KZG\text{-}FFT.eval}(\mathsf{srs}, C_f, D, u, v;\ f(\mathbf{H}_D))$, where $D \le N$ is a one round public coin interactive protocol $\langle P_{\mathsf{eval}}, V_{\mathsf{eval}}\rangle(\mathsf{srs}, C_f, D, u, v; f(\mathbf{H}_D))$ between a PPT prover and a PPT verifier.
   (a) P_{eval}: The prover computes the evaluations of $q(X)$ over the FFT domain of size D, where $q(X)$ satisfies $f(X) - v = (X - u) \cdot q(X)$. Commits to $q(X)$ (denoted C_q) using $\mathsf{srs}_\mathcal{P}$. Sends C_q to V_{eval}.
   (b) V_{eval}: Let $d = \log D$. Computes $C_f \cdot g_1^{-v} \in \mathbb{G}_1$ and $h_{2,n-d} \cdot g_2^{-u} \in \mathbb{G}_2$, and accepts if $e(C_f \cdot g_1^{-v}, g_2) = e(C_q, h_{2,n-d} \cdot g_2^{-u})$ and rejects otherwise.

The KZG-FFT polynomial commitment scheme is presented in Protocol 1. KZG-FFT.Setup is presented in Algorithm 1, Section 3.1. Let $D = 2^d$, and $f \in \mathbb{F}_{<D}[Y]$, and $\mathbf{a} \in \mathbb{F}^D$ be such that $\mathbf{a}$ agrees with $f(\mathbf{H})$ over the FFT domain of size D. The KZG-FFT.commit algorithm takes as input $\mathsf{srs}_\mathcal{P}$, the degree bound D, and the evaluation vector $\mathbf{a}$ of the polynomial $f(Y)$. To commit to polynomials of degree at most $D = 2^d$ for $d \in [0, n]$, the commit algorithm folds the public parameters $\mathsf{srs}_\mathcal{P} = \{h_{1,i}^{(n)}\}_{i\in[0,N-1]}$ recursively as follows:

$$h_{1,i}^{(k)} = h_{1,i}^{(k+1)} \cdot h_{1,i+2^k}^{(k+1)} \quad \forall i \in [0, 2^k - 1] \tag{1}$$

and outputs $C_f = \prod_{i\in[0,D-1]} (h_{1,i}^{(d)})^{a_i}$. We show in Lemma 4 that $C_f = g_1^{f(r^{2^{n-d}})}$.

Lemma 1. *Let $h_{1,i}^{(d)}$ for $d \in [1,n]$ be as defined in Protocol 1, and Equation 1. Further, let f and $\mathbf{a}$ be as defined above. Then $C_f = \prod_{i\in[0,D-1]}(h_{1,i}^{(d)})^{a_i} = g_1^{f(r^{2^{n-d}})}$.*

$\mathsf{KZG\text{-}FFT.eval}(\mathsf{srs}, C_f, d, u, v; f(\mathbf{H}_D))$ is an argument of knowledge for the following relation

$$\{(\mathsf{srs}, (C_f, D, u, v)); f(\mathbf{H}_D) \mid \ f \in \mathbb{F}_{<D}[Y], f(u) = v, \\ \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_{\mathcal{P}}, D, f(\mathbf{H}_D)) = C_f\}$$

The main idea behind the evaluation protocol, like in KZG, is to prove the following polynomial relation at a random point $r^{2^{n-d}}$.[1]

$$f(Y) - f(u) = (Y - u)q(Y)$$

To begin with, an honest prover computes $f(u)$ from $\mathbf{a}$ using Claim 3.2, the proof of which is similar to Claim 3.1, and Lemma 4. We note that to compute $f(u)$, an honest prover has to spend $O(D \cdot d)$ field operations.

Claim 3.2. *Let $f \in \mathbb{F}_{<D}[Y]$, and $\mathbf{a}$ agree with $f(\mathbf{H}_D)$ over the FFT domain of size D. Then for any $u \in \mathbb{F}$,*

$$f(u) = \frac{\sum_{i\in[0,D-1]} \prod_{j\in[0,d-1]}(1 + (\omega_D^{-i} \cdot u)^{2^j})}{D} \ .$$

In the evaluation protocol, the prover P_{eval} computes the evaluations of $q(Y)$ over the FFT domain of size D as follows: for $i \in [0, D-1]$ $q(\omega_D^i) = \frac{a_i - f(x)}{\omega_D^i - x}$. We note that similar to C_f, C_q can be computed from the evaluations of $q(Y)$ over the FFT domain of size D. The verifier V_{eval} checks if $e(C_f \cdot g_1^{-v}, g_2) = e(C_q, h_{2,n-d} \cdot g_2^{-u})$. We show in Theorem 1 that this check ensures either $f(r^{2^{n-d}}) - v = (r^{2^{n-d}} - u) \cdot q(r^{2^{n-d}})$ or there exists $k \in [1, n-d]$ and $q'(Y) \in \mathbb{F}_{<N}[Y]$ such that $f(r) - v = (r^{2^k} - u) \cdot q(r^k)$. Since r is chosen independently and uniformly at random, and is not known to the prover, under the N-DLOG assumption the check passes with high probability if and only if either $f(u) = v$ or $Y^{2^k} - u$ divides $f(Y) - v$. For a given $f \in \mathbb{F}_{<N}$ there exists at most $N/2$ such u's, and hence if u is sampled uniformly at random from $\mathbb{F}$, with high probability u is such that $(Y^{2^k} - u)$ does not divide $f(Y) - v$, and hence $f(u) = v$.

On Updatability of the SRS. We remark that the proof of Theorem 1 shows updatable knowledge-soundness for KZG-FFT in the AGM (as in Definition 3.3 [MBKM19], and [GKM+18]). The work of [GKM+18] shows a negative result about SRS updatability. It roughly says that that for any updatable pairing-based SNARK where the srs consists of encoded polynomials, the prover will also know encodings of the monomials that make up the polynomial. This is negative results since for some SNARKs where the polynomial encoded consist of secrets, soundness is broken if the prover learns the monomials. However, in our construction, this is not a barrier. The prover cannot break soundness even if it learns the encodings of the monomials. This is the reason we are able to show updatable knowledge-soundness even though the srs in KZG-FFT does not consist of encodings of only monomials in r. We provide the update and verify-update algorithms in Appendix A.3.

Theorem 1. *(proof in Appendix A.2) Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow_R \mathcal{G}(1^\lambda)$ be a bilinear group. The protocol* (KZG-FFT.Setup, KZG-FFT.commit, KZG-FFT. eval) *is a PCS for $\mathbb{F}_{<N}[Y]$ in the AGM assuming N-DLOG holds with respect to $\mathcal{G}$.*

Prover and Verifier Complexity: The commit algorithm is cheaper than KZG as it does not have to interpolate the evaluations to obtain the coefficient vector. The evaluation prover performs $O(D)$ field divisions to compute the evaluations of the quotient polynomial over the appropriate FFT domain, and an MSME of size D to commit to $q(X)$. We remark that computing the evaluations of the quotient polynomial can be parallelized across the FFT domain unlike the computation of its coefficients which is inherently serial. The verifier in the evaluation protocol performs 1 exponentiation and 1 group addition over $\mathbb{G}_1$, and $\mathbb{G}_2$ respectively, and 1 pairing check, which is similar to the KZG verifier.

[1] In KZG the random point is r irrespective of the degree.

4 Multilinear Commitment Scheme

In this section, we introduce a novel multilinear commitment scheme denoted as KZG-FOURIER. Drawing inspiration from [BCHO22] and [KT23], our approach leverages a linear homomorphism from the F-linear space of multilinear polynomials to the $\mathbb{F}$-linear space of univariate polynomials. However, our scheme, $\mathcal{U}n$, diverges by mapping the Fourier basis of $\mathbb{F}_{\leq 1}[X_0, \ldots, X_{n-1}]$ to the FFT basis of $\mathbb{F}_{<N}[Y]$ (see Section 2), where $N = 2^n$. Combining the univariate polynomial commitment scheme outlined in Section 3 with the multilinear commitment scheme presented here, we establish a dual commitment scheme, a concept elucidated in Section 6.[2] Section 4.1 is dedicated to defining and proving key properties of $\mathcal{U}_n$, while the setup generation algorithm is elaborated in Section 4.2, and the multilinear polynomial commitment scheme itself is delineated in Section 4.3. Throughout this exposition, we maintain the assumptions that $N = 2^n$, $D = 2^d$, and $K = 2^k$.

4.1 Technical Preliminaries

Readers are directed to Section 2 for pertinent notation. The proof of all the claims and lemmas in this section is deferred to Appendix B.1. We commence by presenting the subsequent claim, akin to Claim 3.1. Recall that $U_i^{(n)}$ denotes the i-th component of the FFT basis of $\mathbb{F}_{<N}[Y]$, and $L_i^{(n)}$ denotes the i-th Fourier basis of $\mathbb{F}_{\leq 1}[X_0, \ldots, X_{n-1}]$.

Claim 4.1. $U_i^{(n)} = \prod_{j \in [0,n-1]} (1 + (\omega_N^{-i} \cdot Y)^{2^j}) \frac{1}{N}$

The linear isomorphism $\mathcal{U}_n$ between $\mathbb{F}_{\leq 1}[X_0, \ldots, X_{n-1}]$ and $\mathbb{F}_{<N}[Y]$ is defined as follows

$$\mathcal{U}_n(L_i^{(n)}) = U_i^{(n)} \quad \text{for } i \in [0, N-1].$$

Below, we proceed to establish several properties of $\mathcal{U}_n$.

Claim 4.2. $\mathcal{U}_n(1) = 1$

Claim 4.3. $\mathcal{U}_n(L_i^{(d)}) = \mathcal{U}_n(L_i^{(d+1)}) + \mathcal{U}_n(L_{i+D}^{(d+1)})$ *for* $d \in [1, n-1]$.[3]

Lemma 2. $\mathcal{U}_n(L_i^{(d)}) = U_i^{(d)}(Y^{2^{n-d}})$, *holds for* $d \in [1, n]$. *Moreover, for* $f \in \mathbb{F}_{\leq 1}[X_0, \ldots, X_{d-1}]$, $\{f_{L_i^{(d)}} \in \mathbb{F}\}_{i \in [0, D-1]}$ *represents the Fourier coefficients of* f *if and only if there exists a* $g \in \mathbb{F}_{<D}[Y]$ *such that* $\mathcal{U}_n(f) = g(Y^{2^{n-d}})$ *and* $g(\omega_D^i) = f_{L_i^{(d)}}$.

Lemma 3. *For* $d \in [1, n-1]$, $f \in \mathbb{F}_{\leq 1}[X_0, \ldots, X_{d-1}]$ *if and only if there exists* $\psi_f \in \mathbb{F}_{<D}[Y]$ *such that*

$$\mathcal{U}_n(X_d \cdot f) = \psi_f(Y^{2^{n-d-1}}) \cdot \prod_{j \in [0, D-1]} (Y^{2^{n-d-1}} - \omega_{2D}^j)$$

$$\mathcal{U}_n((1 - X_d) \cdot f) = \psi_f(-Y^{2^{n-d-1}}) \cdot \prod_{j \in [0, D-1]} (Y^{2^{n-d-1}} + \omega_{2D}^j)$$

Claim 4.4. *Let* $\psi \in \mathbb{F}_{<D}[Y]$. *Then there exists* $\psi_o, \psi_e \in \mathbb{F}_{<2^{d-1}}[Y]$ *such that* $\psi(Y) = \psi_e(Y^2) + Y \cdot \psi_o(Y^2)$, *and* $\psi(-Y) = \psi_e(Y^2) - Y \cdot \psi_e(Y^2)$. *Further,*

$$\psi_e(\omega_{2^{d-1}}^i) = \frac{\psi(\omega_D^i) + \psi(-\omega_D^i)}{2} \qquad \forall i \in [0, 2^{d-1} - 1]$$

$$\psi_o(\omega_{2^{k-1}}^i) = \frac{\psi(\omega_D^i) + \psi(-\omega_D^i)}{2\omega_{2^k}} \qquad \forall i \in [0, 2^{d-1} - 1]$$

[2] In [KT23], the Fourier basis of $\mathbb{F}\leq 1[X_0, \ldots, X_{n-1}]$ is mapped to the monomial basis of $\mathbb{F}_{<N}[Y]$, whereas in [BCHO22], the monomial basis of $\mathbb{F}\leq 1[X_0, \ldots, X_{n-1}]$ is mapped to the monomial basis of $\mathbb{F}_{<N}[Y]$.

[3] It is important to note that $\mathcal{U}_n(L_i^{(d)})$ differs from $\mathcal{U}_d(L_i^{(d)})$.

4.2 Setup Generation

Algorithm 2 KZG-FOURIER.Setup(1^λ): Setup Generation for KZG-FOURIER

Input: $\{1^\lambda, N\}$
Output: $\{\mathsf{srs} = (\mathsf{srs}_\mathcal{P}, \mathsf{srs}_\mathcal{V}), \pi, \}$

1: $((\mathsf{srs}'_\mathcal{P}, \mathsf{srs}'_\mathcal{V}), \pi) \leftarrow \mathsf{KZG\text{-}FFT.setup}(1^\lambda)$.
2: Let $\mathsf{srs}_\mathcal{P} = \mathsf{srs}'_\mathcal{P} = \{h^{(n)}_{1,i}\}_{i\in[0,N-1]}$.
3: For $d \in [0, n-1]$, let $\phi_d(Y) = \prod_{j=0}^{D-1}(Y - \omega_{2D}^j)$, and compute the evaluations of ϕ_d over the FFT domain of size $2D$. Compute $C_{\phi_d} = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_\mathcal{P}, 2D, \phi_d(\mathbf{H}_{2D}))$ as the commitment to ϕ_d for $d \in [0, n-1]$.
4: Let $\mathsf{srs}_\mathcal{V} = (\mathsf{srs}'_\mathcal{V}, \{C_{\phi_d}\}_{d\in[1,n-1]})$
5: Output $\{\mathsf{srs} = (\mathsf{srs}_\mathcal{P}, \mathsf{srs}_\mathcal{V}), \pi\}$.

Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow_R \mathcal{G}(1^\lambda)$ be a bilinear group. The algorithm for generating the setup for KZG-FOURIER is outlined in Algorithm 2. It takes the security parameter λ and the bound on the number of multilinear monomials N as input and produces srs along with a proof π. The srs, like in KZG-FFT, has two disjoint parts $(\mathsf{srs}_\mathcal{P}, \mathsf{srs}_\mathcal{V})$: $\mathsf{srs}_\mathcal{P}$ is used by the prover and $\mathsf{srs}_\mathcal{V}$ is used by the verifier. In Step 1, Algorithm 2 invokes KZG-FFT.setup, yielding srs′. It sets $\mathsf{srs}_\mathcal{P}$ identical to $\mathsf{srs}'_\mathcal{P}$. For the evaluation protocol of KZG-FOURIER, the verifier needs to evaluate $\phi_d(Y) = \prod_{j=0}^{D-1}(Y - \omega_{2D}^j)$ at a randomly chosen point. Hence, $\mathsf{srs}_\mathcal{V}$ includes commitments to $\phi_d(Y)$, for $d \in [1, n-1]$, in addition to $\mathsf{srs}'_\mathcal{V}$. Particularly in Step 3, the algorithm computes evaluations of the polynomial $\phi_d(Y)$ over the FFT domain of size $2D$, for $d \in [1, n-1]$, and at Step 4, it commits to it using KZG-FFT.commit. It's worth noting that despite the degree of ϕ_d being at most D, Algorithm 2 commits to it as a polynomial of degree at most $2D$. This is required in batch evaluation check for all the univariate polynomials in KZG-FOURIER.eval. Additionally, even though $\phi_d(\mathbf{H}_{2D})$ is of size $2D$, at least D of its components are 0 since $\phi_d(\omega_{2D}^{2i}) = 0$, for $i \in [0, D-1]$. Thus for $d \in [1, n-1]$, KZG-FFT.commit requires an MSME of length D (not $2D$).

Updatability. The srs returned by KZG-FOURIER.Setup can be updated using KZG-FFT.update_setup (Algorithm 4) and KZG-FFT.verify_setup (Algorithm 5) algorithms. The KZG-FOURIER. update_setup algorithm additionally computes new commitments to the public polynomials $\phi_d(Y)$ for $d \in [1, n-1]$ using the updated SRS returned by KZG-FFT.update_setup. The KZG-FOURIER.verify_ setup algorithm uses KZG-FFT.verify_setup to verify the first part of the SRS. Additionally, it verifies the commitments to $\phi_d(Y)$ for $d \in [1, n-1]$ by running the KZG-FFT.eval protocol: it evaluates $\phi_d(Y)$ at a random point $u \in_R \mathbb{F}$, computes a proof of $\phi_d(u) = z$ and checks that the proof verifies.

4.3 Protocol

The KZG-FOURIER commitment scheme is detailed in Protocol 2. During Step 1, the setup is generated following Algorithm 2. Let $f \in \mathbb{F}_{\leq 1}[X_0, \ldots, X_{d-1}]$ and $f_{L_i^{(d)}} \in \mathbb{F}$ represent its D Fourier coefficients. Throughout this section, we associate f_i with $f_{L_i^{(d)}}$ for $i \in [0, D-1]$. Consider $\mathbf{f} \in \mathbb{F}^D$ such that the i-th component of $\mathbf{f}$ is f_i for $i \in [0, D-1]$. The KZG-FOURIER.commit algorithm, in Step 2, accepts $\mathsf{srs}_\mathcal{P}$, $\mathbf{f} \in F^D$, and D as input, and produces $C_f = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_\mathcal{P}, D, \mathbf{f})$ as output. From Lemma 10, it follows that there exists $w_f \in \mathbb{F}_{<D}[Y]$ such that $\mathcal{U}_d(f) = w_f(Y)$, and $w_f(\omega_D^i) = f_i$ for $i \in [0, D-1]$. When f is evident from the context, we drop it from the subscript of w_f. The commit algorithm treats $\mathbf{f}$ as the evaluations of w over the FFT domain of size D and commits to it using KZG-FFT.commit. From Lemma 4,

$$C_f = C_{w_f} = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_\mathcal{P}, D, w_f(\mathbf{H}_D)) = g_1^{w(r^{2^{n-d}})}.$$

We interchangeably use C_f and C_{w_f}, as they denote the same commitment.

We denote $(X_0, \ldots, X_{d-1})$ as $\mathbf{X}^{(d)}$, and $(x_0, \ldots, x_{d-1})$ as $\mathbf{x}^{(d)}$, where $x_i \in \mathbb{F}$ for $i \in [0, d-1]$. The KZG-FOURIER.eval algorithm at Step 3 is a ten round public coin protocol $\langle P_{\mathsf{eval}}, V_{\mathsf{eval}} \rangle(\mathsf{srs}, C_f, D, \mathbf{x}^{(d)}, y; \mathbf{f})$. According to Fact 4, ensuring correctness of KZG-FOURIER.eval requires V_{eval} to only check

$$f(\mathbf{X}^{(d)}) - f(\mathbf{x}^{(d)}) = \sum_{k=0}^{d-1} (X_k - x_k) \cdot q_k(\mathbf{X}^{(k)}). \tag{2}$$

As $\mathcal{U}_n$ is a linear isomorphism, it suffices for V_{eval} to verify the aforementioned relation under $\mathcal{U}_n$. In essence, the primary concept behind the evaluation protocol is to empower the verifier to verify the following (univariate) polynomial relation evaluated at a random point z:

$$\mathcal{U}_d(f) - \mathcal{U}_d(y) = \left(\sum_{k=0}^{d-1} \big(\mathcal{U}_d(X_k \cdot q_k) - \mathcal{U}_d(x_k \cdot q_k) \big) \right) \tag{3}$$

At Step 3a, P_{eval} calculates the Fourier coefficients of such $q_k(\mathbf{X}^{(k)}$ for $k \in [1, d-1]$. Let ψ_{q_k}, $\psi_{q_k,e}$ and $\psi_{q_k,o}$ correspond to $q_k(\mathbf{X}^{(k)}$ as described in Lemma 11 and Claim B.4. As outlined in Claim B.5, at Step 3b, for each $k \in [1, d-1]$, P_{eval} can determine the evaluations of $\psi_{q_k,e}(Y)$ and $\psi_{q_k,o}(Y)$ over the FFT domain of size D. P_{eval} computes commitments to $\psi_{q_k,e}$ and $\psi_{q_k,o}$, for $k \in [1, d-1]$ as follows:

$$C_{\psi_{q_k,e}} = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_{\mathcal{P}}, D, \psi_{q_k,e}(\mathbf{H}_D)) \tag{4}$$

$$C_{\psi_{q_k,o}} = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_{\mathcal{P}}, D, \psi_{q_k,o}(\mathbf{H}_D)) \tag{5}$$

We note that even though the degrees of $\psi_{q_k,e}$ and $\psi_{q_k,o}$ are at most $K/2$, P_{eval} commits to them using their evaluations over FFT domain of size D. This helps in batch evaluation of all the univariate polynomials, and establishing degree bounds on them in the steps ahead.

At Step 3c, P_{eval} computes $w(z)$, $\psi_{q_k,e}(z^{2^{d-k}})$, and $\psi_{q_k,o}(z^{2^{d-k}})$ for $k \in [1, d-1]$, utilizing the evaluations of the respective polynomials over the FFT domains of size D (refer to Claim 3.2), and $\phi_k(z^{2^{d-k-1}})$ for $k \in [1, d-1]$ directly by evaluating the expression. At Step 3d, V_{eval} calculates $\psi_{q_k}(z^{2^{d-k-1}})$ and $\psi_{q_k}(-z^{2^{d-k-1}})$ using the following equations, derived from Claim B.4:

$$\psi_{q_k}(z^{2^{d-k-1}}) = \psi_{q_k,e}(z^{2^{d-k}}) + z^{2^{d-k-1}} \cdot \psi_{q_k,o}(z^{2^{d-k}}) \tag{6}$$

$$\psi_{q_k}(-z^{2^{d-k-1}}) = \psi_{q_k,e}(z^{2^{d-k}}) - z^{2^{d-k-1}} \cdot \psi_{q_k,o}(z^{2^{d-k}}) \tag{7}$$

V_{eval} computes $\mathcal{U}_d(q_k)(z)$ and $\mathcal{U}_d(X_k q_k)(z)$ using the following equations, for $k \in [1, d-1]$

$$\begin{aligned} \mathcal{U}_d(q_k)(z) = {} & \psi_{q_k}(z^{2^{d-k-1}}) \cdot \phi_k(z^{2^{d-k-1}}) \\ & + \psi_{q_k}(-z^{2^{d-k-1}}) \cdot \frac{z^{2D \cdot 2^{d-k-1}} - 1}{\phi_k(z^{2^{d-k-1}})} \end{aligned} \tag{8}$$

$$\mathcal{U}_d(X_k q_k)(z) = \psi_{q_k}(z^{2^{d-k-1}}) \cdot \phi_k(z^{2^{d-k-1}}) \tag{9}$$

Equations 8 and 9 above are derived from Lemma 11, and the following fact concerning the primitive root of unity

$$\prod_{j \in [0, D-1]} (Y^{2^{d-k-1}} - \omega_{2D}^j) \cdot \prod_{j \in [0, D-1]} (Y^{2^{d-k-1}} + \omega_{2D}^j) = (Y^{2^{d-k-1}})^{2D} - 1 \ .$$

Additionally, from Claim B.2, we have $\mathcal{U}_n(x_0 q_0) = x_0 q_0$. Moreover, according to Lemma 10,

$$\mathcal{U}_d(X_0 q_0) = \frac{(1 - Y^{2^{d-1}}) q_0}{2}, \text{ and hence, } \mathcal{U}_d(X_0 q_0)(z) = \frac{(1 - z^{2^{d-1}}) q_0}{2}.$$

V_{eval} checks whether these values satisfy

$$\mathcal{U}_d(f)(z) - y = \sum_{k=0}^{n-1} \mathcal{U}_d(X_k \cdot q_k)(z) - x_k \cdot \mathcal{U}_d(q_k)(z) \tag{10}$$

Conditioned on values sent by P_{eval} corresponding to $w(z)$, and $\psi_{q_k,e}(z^{2^{d-k}})$, $\psi_{q_k,o}(z^{2^{d-k}})$, $\phi_k(z^{2^{d-k-1}})$ for $k \in [1, d-1]$, are correct, the above check ensures Equation 3 holds with high probability over the random choice of z (using Schwartz-Zippel). Hence, the protocol hereafter ensures the correctness of these vales sent by P_{eval}. To this end, for $k \in [1, d-1]$, define $\zeta_{q_k,e}, \zeta_{q_k,o} \in \mathbb{F}_{<D}[Y]$ as follows:

$$\zeta_{q_k,e}(Y) = Y^{2^d - 2^{k-1}} \cdot \psi_{q_k,e}(Y), \text{ and } \zeta_{q_k,o}(Y) = Y^{2^d - 2^{k-1}} \cdot \psi_{q_k,o}(Y)$$

The key concept in the upcoming steps involves defining the polynomial $\eta(Y) \in \mathbb{F}_{<D}[Y]$ (refer to Equations 11 and 12 below) and verifying its evaluation at a random point. The accurate evaluation of $\eta(Y)$ at the specified random point confirms the correctness of the aforementioned values sent by P_{eval}. We elaborate upon these next steps below. At Step 3e, $\psi(Y) \in \mathbb{F}_{<D}[Y]$ is defined as follows:

$$\begin{aligned}
\psi(Y) = \gamma_w \cdot \frac{w(Y) - w(z)}{Y - z} + \\
\sum_{k \in [1,d-1]} \Bigg(\delta_k \cdot \Big(\frac{\phi_k(Y^{2^{d-k-1}}) - \phi_k(z^{2^{d-k-1}}))}{Y^{2^{d-k-1}} - z} \Big) \; + \\
\frac{\gamma_{k,e} \cdot \big(\psi_{q_k,e}(Y) - \psi_{q_k,e}(z^{2^{d-k}})\big) + \gamma'_{k,e} \cdot \big(\zeta_{q_k,e}(Y) - \zeta_{q_k,e}(z^{2^{d-k}})\big)}{Y - z^{2^{d-k}}} \; + \\
\frac{\gamma_{k,o} \cdot \big(\psi_{q_k,o}(Y) - \psi_{q_k,o}(z^{2^{d-k}})\big) + \gamma'_{k,o} \cdot \big(\zeta_{q_k,o}(Y) - \zeta_{q_k,o}(z^{2^{d-k}})\big)}{Y - z^{2^{d-k}}} \Bigg)
\end{aligned} \tag{11}$$

At Step 3i, $\eta(Y)$ is defined as follows

$$\begin{aligned}
\eta(Y) = \psi(Y) + \beta_w \cdot w(Y) + \sum_{k \in [1,d-1]} \Bigg(\beta_{k,e} \cdot \psi_{q_k,e}(Y) \; + \\
\beta_{k,o} \cdot \psi_{q_k,o}(Y) + \kappa_k \cdot \phi_k(Y^{2^{d-k-1}}) \Bigg)
\end{aligned} \tag{12}$$

From Equation 12, and Lemma 4, it follows that V_{eval} at the last can compute the commitment to $\eta(Y)$ as follows:

$$C_\eta = C_\psi \cdot C_w^{\beta_w} \cdot \Big(\prod_{k \in [1,d-1]} C_{\psi_{q_k,e}}^{\beta_{k,e}} \cdot C_{\psi_{q_k,o}}^{\beta_{k,o}} \cdot C_{\phi_k}^{\kappa_k} \Big) \tag{13}$$

We show in Theorem 2 that (KZG-FOURIER.Setup, KZG-FOURIER. commit KZG-FOURIER.eval) specified in Protocol 2 forms a multilinear polynomial commitment scheme. Additionally, like in Theorem 1, we prove that the evaluation protocol KZG-FOURIER.eval is updatable knowledge sound. The proof is deferred to Appendix B.2.

Theorem 2. *The protocol* (KZG-FOURIER.Setup, KZG-FOURIER. commit KZG-FOURIER.eval) *forms a polynomial commitment scheme for* $\mathbb{F}_{\leq 1}[X_0, \ldots, X_{n-1}]$ *in the AGM under the N-DLOG assumption.*

Prover and Verifier Complexity: Although the evaluation prover of KZG-FOURIER works in $O(D \log D)$ time, concretely the prover spends a lot of time at Step 3a computing the evaluations of $\psi_{q_k,o}$ and $\psi_{q_k,e}$. This step alone requires $4 \log D - 4$ FFTs of appropriate sizes and $O(D \log D)$ field multiplications and divisions. The evaluation verifier of KZG-FOURIER operates in $O(\log D)$ time. The check performed by verifier at Step 3d requires $O(\log D)$ field operations. The computation of C_η at Step 3j requires $3 \log D - 1$ $\mathbb{G}_1$ exponentiations.

Protocol 2: KZG-FOURIER multilinear commitment scheme

1. $\{\mathsf{srs}, \pi\} \leftarrow_R \mathsf{KZG\text{-}FOURIER.setup}(1^\lambda, N)$, where λ is security parameter.
2. $C_f \leftarrow \mathsf{KZG\text{-}FOURIER.commit}(\mathsf{srs}_\mathcal{P}, D, \mathbf{f})$, where
$$C_f = C_{w_f} = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_\mathcal{P}, D, w_f(\mathbf{H}_D))$$
and $\mathcal{U}_d(f) = w_f(Y)$.
3. accept/reject $\leftarrow \mathsf{KZG\text{-}FOURIER.eval}(\mathsf{srs}, C_f, D, \mathbf{x}^{(d)}, y; \mathbf{f})$ is a ten round public coin interactive protocol $\langle P_{\mathsf{eval}}, V_{\mathsf{eval}}\rangle(\mathsf{srs}, C_f, D, \mathbf{x}^{(d)}, y; \mathbf{f})$ between a PPT prover and a PPT verifier.
   (a) $P_{\mathsf{eval}} \to V_{\mathsf{eval}}$: For $k \in [1, d-1]$ - Compute the K Fourier coefficients of $q_k(\mathbf{X}^{(k)})$ for $k \in [1, d-1]$ and $q_0 \in \mathbb{F}$, satisfying Equation 2. Let ψ_{q_k} correspond to $q_k(\mathbf{X}^{(k)})$ as described in Lemma 11; Compute the evaluations of $\psi_{q_k,e}$ and $\psi_{q_k,o}$ (see Claim B.4) over the FFT domain of size D, and commit to them as in Equations 4 and 5 respectively. Send these $2(d-1)$ commits and $q_0 \in \mathbb{F}$ to V_{eval}.
   (b) $V_{\mathsf{eval}} \to P_{\mathsf{eval}}$: Sample $z \in_r \mathbb{F}$ and send to P_{eval}.
   (c) $P_{\mathsf{eval}} \to V_{\mathsf{eval}}$: Send $w_f(z)$, and $\psi_{q_k,e}(z^{2^{d-k}})$, $\psi_{q_k,o}(z^{2^{d-k}})$ and $\phi_k(z^{2^{d-k-1}})$ for $k \in [1, d-1]$ to V_{eval}.
   (d) $V_{\mathsf{eval}} \to P_{\mathsf{eval}}$: For $k \in [1, d-1]$ - Compute $\psi_{q_k}(z^{2^{d-k-1}})$ and $\psi_{q_k}(-z^{2^{d-k-1}})$ using Equations 6 and 7, respectively. Then calculate $\mathcal{U}_n(q_k)(z)$ and $\mathcal{U}_n(X_k q_k)(z)$ using Equations 8 and 9, respectively. Use these values to check and continue if Equation 10 holds, else output reject. Sample $\gamma_w \in_r \mathbb{F}$, and $\gamma_{k,e}, \gamma'_{k,e}, \gamma_{k,o}, \gamma'_{k,o}, \delta_k \in_r \mathbb{F}$ for $k \in [1, d-1]$. Send these values to P_{eval}.
   (e) $P_{\mathsf{eval}} \to V_{\mathsf{eval}}$: $\psi(Y) \in \mathbb{F}_{<D}[Y]$ is as defined in Equation 11. Compute the evaluations of $\psi(Y)$ over $\mathbf{H}_D$, and commit to it as follows: $C_\psi = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_\mathcal{P}, D, \psi(\mathbf{H}_D))$. Send C_ψ to V_{eval}.
   (f) $V_{\mathsf{eval}} \to P_{\mathsf{eval}}$: Sample $s \in_r \mathbb{F}$ and send it to P_{eval}.
   (g) $P_{\mathsf{eval}} \to V_{\mathsf{eval}}$: Send $w(s)$, $\psi_{k,e}(s)$, $\psi_{k,o}(s)$, $\phi_k(s)$ for all $k \in [1, d-1]$ to V_{eval}.
   (h) $V_{\mathsf{eval}} \to P_{\mathsf{eval}}$: Sample $\beta_w \in_r \mathbb{F}$, and $\beta_{k,e}$, $\beta_{k,o}$, $\kappa_k \in_r \mathbb{F}$ for $k \in [1, d-1]$ independently and uniformly at random from $\mathbb{F}$ for $k \in [1, d-1]$, and send it to P_{eval}.
   (i) $P_{\mathsf{eval}} \to V_{\mathsf{eval}}$: Let $\eta(Y)$ be as defined in Equation 12. Compute the evaluations of $\mu(Y) \in \mathbb{F}_{<D}[Y]$ over $\mathbf{H}_D$, where $\eta(Y) - \eta(s) = (Y - s)\mu(Y)$. Compute the commit to $\mu(Y)$: $C_\mu = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_\mathcal{P}, D, \mu(\mathbf{H}_D))$, and send it to V_{eval}.
   (j) $V_{\mathsf{eval}} \to P_{\mathsf{eval}}$: Compute C_η using Equation 13 and $\eta(u)$ using Equations 11, and 12, and check
$$e(C_\eta \cdot g_1^{-\eta(s)}, g_2) = e(C_\mu, h_{2,d} \cdot g_2^{-s})$$

5 Argument of Knowledge to Establish Linear Relations

In this section, we present the the argument of knowledge that facilitates a prover to demonstrate a predefined linear relationship between two AFG committed witnesses. This helps in designing a succinct dual polynomial commitment scheme in Section 6.2 requiring only a transparent setup. Consider a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow_R \mathcal{G}(1^\lambda)$. Let $d, n, D, N \in \mathbb{N}$ be such that $2^d = D$, $2^n = N$, and $D \leq N$ Within this context, let $\tau^{(1,d)} \in \mathbb{G}_1^D$, and $\tau^{(2,d)} \in \mathbb{G}_2^D$ be publicly known vectors (also refer to Section D.1). Furthermore, let $\mathbf{a}, \mathbf{f} \in \mathbb{F}^D$ represent two vectors, and denote the commitments to $\mathbf{f}$ and $\mathbf{a}$ as $C_\mathbf{f}$ and $C_\mathbf{a}$ respectively, defined

as follows:

$$C_{\mathbf{f}} = \prod_{i=0}^{D-1} e(\tau_i^{(1,d)}, g_2^{f_i}) \tag{14}$$

$$C_{\mathbf{a}} = \prod_{i=0}^{D-1} e(\tau_i^{(1,d)}, g_2^{a_i}) \tag{15}$$

Here a_i, and f_i represent the i-th component of $\mathbf{a}$, and $\mathbf{f}$ respectively. We note that C_f and $C_{\mathbf{a}}$ is the AFG commitment to $g_2^{\mathbf{f}}$ and $g_2^{\mathbf{a}}$ (and therefore $\mathbf{f}$ and $\mathbf{a}$) with respect to commitment key $\tau^{(1,d)}$. The relation corresponding to which we present the argument of knowledge is as defined below:

$$\begin{aligned}\mathcal{R}_L = \{C_{\mathbf{f}}, C_{\mathbf{a}}, D \mid &\exists \mathbf{a}, \mathbf{f} \in \mathbb{F}^D \text{ such that } M_{\omega_D} \cdot \mathbf{f} = \mathbf{a}, \\ &\text{and } C_{\mathbf{f}}, C_{\mathbf{a}} \text{ satisfy Eq. } 14, 15\}\end{aligned} \tag{16}$$

Recall, M_{ω_D} is the FFT matrix of size D. We remark that, in the argument system, we can replace M_{ω_D} with any matrix, but for convenience we work with FFT matrices. The argument of knowledge presented in this section requires a transparent setup, which we show how to generate in Section 5.1. We state the protocol itself and its correctness in Section 5.2.

5.1 Transparent Setup for Linear Relation

The algorithm to generate the setup for the argument system is presented in Algorithm 3. The protocol presented in Section 5.2, runs three instances of Dory [Lee21] (also see Appendix D). Hence, the protocol requires the dory setup denoted $\mathrm{pp}_{\mathsf{dory}}$. We refer the reader to Section D.1 for an exact description of $\mathrm{pp}_{\mathsf{dory}}$. At Steps 2-5, Algorithm 3 computes n commitments $\{C_{\omega_D}\}_{d\in[1,n]}$ to matrices $\{M_{\omega_D}\}_{d\in[1,n]}$ respectively, where $C_{\omega_D} \in \mathbb{G}_T$ and is computed as in Steps 3 and 4. Here, $\tau^{(2,d)}$ is part of $\mathsf{pp}_{\mathsf{dory}}$. Finally at Step 7, Algorithm 3, outputs the public parameters $\mathrm{pp} = (\mathrm{pp}_{\mathcal{P}}, \mathrm{pp}_{\mathcal{V}})$, where $\mathrm{pp}_{\mathcal{P}} = \mathrm{pp}_{\mathsf{dory},\mathcal{P}}$, and $\mathrm{pp}_{\mathcal{V}} = \{\mathrm{pp}_{\mathsf{dory},\mathcal{V}}, \{C_{M_{\omega_D}}\}_{d\in[1,n]}\}$.

Algorithm 3 Setup Generation for Linear Relation

Input: $\{1^\lambda, n\}$
Output: $\mathrm{pp} = (\mathrm{pp}_{\mathcal{P}}, \mathrm{pp}_{\mathcal{V}})$

1: $\mathrm{pp}_{\mathsf{dory}} \leftarrow \mathsf{dory.setup}(1^\lambda)$.
2: **for** $d \in [1, n]$ **do**
3: Let $m_{D,j} = \prod_{i\in[0,D-1]} (\tau_i^{(1,d)})^{\omega_D^{i\cdot j}}$
4: Compute $C_{M_{\omega_D}} = \prod_{j\in[0,D-1]} e(m_{d,j}, \tau_j^{(2,d)})$
5: **end for**
6: Let $\mathsf{pp}_{\mathcal{P}} = \mathsf{pp}_{\mathsf{dory},\mathcal{P}}$, and $\mathsf{pp}_{\mathcal{V}} = \mathsf{pp}_{\mathsf{dory},\mathcal{V}} \cup \{C_{M_{\omega_D}}\}_{d\in[1,n]}$
7: Output $\mathrm{pp} = (\mathrm{pp}_{\mathcal{P}}, \mathrm{pp}_{\mathcal{V}})$.

5.2 Proving Linear Relations

The argument system for $\mathcal{R}_L$ from Equation 16 is presented in Protocol 3. The public parameters are as generated by Algorithm 3. The central idea of the protocol is to enable the verifier to check

$$C_{\mathbf{a}} = \prod_{j\in[0,D-1]} e(m_{D,j}, g_2^{f_j}) = \prod_{i\in[0,D-1]} e(\tau_i^{(1,d)}, g_2^{a_i}) \tag{17}$$

Here, $m_{D,j}$ is as defined in Step 4 of Algorithm 3. We argue as part of the completeness of the protocol that $M_{\omega_D} \cdot \mathbf{f} = \mathbf{a}$ implies the above equality. Hence, the linear relation can be established if the verifier is able to check the following two statements:

1. $\exists \mathbf{a} \in \mathbb{F}^D$ such that $C_{\mathbf{a}} = \prod_{i=0}^{D-1} e(\tau_i^{(1,d)}, g_2^{a_i})$.

2. $\exists \mathbf{f} \in \mathbb{F}^D$ such that $C_{\mathbf{f}} = \prod_{i=0}^{D-1} e(\tau_i^{(1,d)}, g_2^{f_i})$, $C_{M_{\omega_D}} = \prod_{j \in [0,D-1]} e(m_{D,j}, \tau_j^{(2,d)})$, and $C_{\mathbf{a}} = \prod_{j \in [0,D-1]} e(m_{D,j}, g_2^{f_j})$

At Step 1, Statement 1 above is proved using one instance of Dory (Appendix D.2, Protocol 7): $\langle P_{\mathsf{dory}}, V_{\mathsf{dory}} \rangle(\mathrm{pp}, C_{\mathbf{a}}, C_{\tau_d}, C_{\mathbf{a}}, D; \mathbf{a})$ Here, $C_{\tau_d} = \langle \tau^{(1,d)}, \tau^{(2,d)} \rangle$, and is part of $\mathsf{pp}_{\mathsf{dory}}$ (see Section D.1). Also, we remark that although as per Protocol 7 the private input to the prover at Step 1 $g_2^{\mathbf{a}}$ respectively, we use $\mathbf{a}$ for clarity as $g_2^{\mathbf{a}}$ can be computed from $\mathbf{a}$. At Step 2, Statement 2 above is proved using an instance of dory: $\langle P_{\mathsf{dory}}, V_{\mathsf{dory}} \rangle(\mathrm{pp}, C_{\mathbf{a}}, C_{M_{\omega_D}}, C_{\mathbf{f}}, D; \mathbf{f})$. Here, the private input to the prover are $(m_{D,0}, \ldots, m_{D,D-1})$ and $g_2^{\mathbf{f}}$, which can be computed from M_{ω_D} and $\mathbf{f}$. We argue the correctness of Protocol 3 in Theorem 3 (proof in Appendix C).

Theorem 3. *Assuming SXDH in the bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, Protocol 3 is an argument of knowledge for relation $\mathcal{R}_L$ stated in Equation 16.*

Protocol 3: Argument System for Linear Relations

$\mathrm{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\langle P_{\mathsf{lin}}, V_{\mathsf{lin}} \rangle(\mathrm{pp}, C_{\mathbf{f}}, C_{\mathbf{a}}, D; \mathbf{a})$:

1. $P_{\mathsf{lin}}, V_{\mathsf{lin}}$ execute $\langle P_{\mathsf{dory}}, V_{\mathsf{dory}} \rangle(\mathrm{pp}, C_{\mathbf{a}}, C_{\tau_d}, C_{\mathbf{a}}, D; \mathbf{a})$. Here $C_{\tau_d} = \langle \tau^{(1,d)}, \tau^{(2,d)} \rangle$ is part of pp.
2. $P_{\mathsf{lin}}, V_{\mathsf{lin}}$ execute $\langle P_{\mathsf{dory}}, V_{\mathsf{dory}} \rangle(\mathrm{pp}, C_{\mathbf{a}}, C_{M_{\omega_D}}, C_{\mathbf{f}}, D; M_{\omega_D}, \mathbf{f})$. Here $C_{M_{\omega_D}}$ is as explained in Section 5.1 and is part of pp.
3. V_{lin} accepts if and only if the verifier accepts in all the above steps.

6 Linking Univariate and Multilinear Polynomial Commitments

In this section we introduce the notion of dual polynomial commitment scheme, that links univariate and multilinear polynomial commitment schemes. Specifically, a dual polynomial commitment scheme can be used to prove evaluations of a univariate and multilinear polynomial derived from the same witness. Formally, a dual polynomial commitment scheme over $\mathbb{F}$ is a tuple $\mathsf{PC} = (\mathsf{setup}, \mathsf{commit}, \mathsf{open}, \mathsf{prove\_link}, \mathsf{eval\_mult}, \mathsf{eval\_uni})$ where:

- $\mathsf{pp} \leftarrow \mathsf{setup}(1^\lambda, D)$. On input security parameter λ, an upper bound $N \in \mathbb{N}$ on the degree of the univariate polynomial (or alternatively on the size of the Fourier basis in case of multilinear polynomials), setup generates public parameters pp.
- $(C_f, C_{\mathbf{a}}, \mathsf{aux}) \leftarrow \mathsf{commit}(\mathsf{pp}, \mathbf{a})$. On input the public parameters pp, a vector $\mathbf{a} \in \mathbb{F}^D$, where $D \leq N$, commit outputs auxiliary information aux and commitments to two polynomials: a) C_f is the commitment to the univariate polynomial $f(Y)$ such that $\mathbf{a}$ agrees with $f(\mathbf{H}_D)$ b) $C_{\mathbf{a}}$ is the commitment to the MLE $\tilde{a}$ of $\mathbf{a}$.
- $b \leftarrow \mathsf{open}(\mathsf{pp}, \mathbf{a}, D, (C_f, C_{\tilde{a}}), \mathsf{aux})$ On input the public parameters pp, the commitments $(C_f, C_{\tilde{a}})$ and auxiliary information aux, a vector $\mathbf{a} \in \mathbb{F}^D$, open outputs a bit indicating accept or reject.
- $b \leftarrow \mathsf{prove\_link}$. A public coin interactive protocol $\langle P_{\mathsf{link}}, V_{\mathsf{link}} \rangle(\mathsf{pp}, C_f, C_{\mathbf{a}}, D; \mathbf{a})$ between a PPT prover and a PPT verifier. The parties have as common input public parameters pp, commitments C_f and $C_{\mathbf{a}}$, degree $D \leq N$. The prover has, in addition, the

witness vector $\mathbf{a} \in F^D$ corresponding to C_f and $C_{\mathbf{a}}$. At the end of the protocol, the verifier outputs 1 indicating accepting the proof or outputs 0 indicating rejecting the proof. The verifier accepts the proof if there exists $\mathbf{a} \in \mathbb{F}^D$ and $f \in \mathbb{F}_{<D}[Y]$ such that a) $\mathbf{a}$ agrees with $f(\mathbf{H}_D)$, b) $C_{\mathbf{a}}$ is the commitment to $\tilde{a}$, and c) C_f is the commitment to $f(Y)$.

- $b \leftarrow \mathsf{eval}, \mathsf{mult}(\mathsf{pp}, C_{\mathbf{a}}, d, \mathbf{x}, y; \mathbf{a})$. A public coin interactive protocol $\langle P_{\mathsf{eval},\mathsf{mult}}, V_{\mathsf{eval},\mathsf{mult}} \rangle(\mathsf{pp}, C_{\mathbf{a}}, d, \mathbf{x}, y; \mathbf{a})$ between a PPT prover and a PPT verifier. The parties have as common input public parameters pp, commitment $C_{\mathbf{a}}$, the bound on the number variables d, evaluation point $\mathbf{x} \in \mathbb{F}^d$, and claimed evaluation y. The prover has, in addition, the opening $\mathbf{a}$ of $C_{\mathbf{a}}$, where $\tilde{a}$ is the MLE of $\mathbf{a} \in \mathbb{F}^D$. At the end of the protocol, the verifier outputs 1 indicating accepting the proof that $\tilde{a}(\mathbf{x}) = y$, or outputs 0 indicating rejecting the proof.

- $b \leftarrow \mathsf{eval}, \mathsf{uni}(\mathsf{pp}, C_f, D, u, v; f(Y))$. A public coin interactive protocol $\langle P_{\mathsf{eval},\mathsf{uni}}, V_{\mathsf{eval},\mathsf{uni}} \rangle(\mathsf{pp}, C_f, D, u, v; f(Y))$ between a PPT prover and a PPT verifier. The parties have as common input public parameters pp, commitment C_f, degree bound D, evaluation point $u \in \mathbb{F}$, and claimed evaluation v. The prover has, in addition, the polynomial $f(Y)$ bound to C_f, with degree of f at most D. At the end of the protocol, the verifier outputs 1 indicating accepting the proof that $f(u) = v$, or outputs 0 indicating rejecting the proof.

A dual polynomial commitment scheme must satisfy completeness, binding, extractability, and linking soundness.

Definition 9 (Completeness)**.** *For all $\mathbf{a} \in \mathbb{F}^D$, $D \leq N$, and for all $\mathbf{x} \in \mathbb{F}^d$, $u \in \mathbb{F}$,*

$$\Pr\left[b = 1 \ : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{setup}(1^\lambda, N) \\ (C_f, C_{\mathbf{a}}) \leftarrow \mathsf{commit}(\mathsf{pp}, \mathbf{a}) \\ v \leftarrow f(u), \mathbf{a} \textit{ agrees with } f(\mathbf{H}_D) \\ y \leftarrow \tilde{a}(\mathbf{x}) \\ b \leftarrow \mathsf{eval}, \mathsf{mult}(\mathsf{pp}, C_{\mathbf{a}}, d, \mathbf{x}, y; \mathbf{a}) \\ \wedge \ \ \mathsf{eval}, \mathsf{uni}(\mathsf{pp}, C_f, d, u, v; f(X)) \end{array} \right] = 1.$$

Definition 10 (Binding)**.** *A dual polynomial commitment scheme* PC *is binding if for all PPT $\mathcal{A}$, the following probability is negligible in λ:*

$$\Pr\left[\begin{array}{c} \mathsf{open}(\mathsf{pp}, \mathbf{a}_0, D, C, \mathsf{aux}_0) = 1 \wedge \\ \mathsf{open}(\mathsf{pp}, \mathbf{a}_1, D, C, \mathsf{aux}_1) = 1 \wedge \\ \mathbf{a}_0 \neq \mathbf{a}_1 \end{array} : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{setup}(1^\lambda, N) \\ (C, \mathbf{a}_0, \mathbf{a}_1, \mathsf{aux}_0, \mathsf{aux}_1, D) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \textit{where } C = (C_f, C_{\mathbf{a}}) \end{array} \right]$$

Definition 11 (Evaluation Knowledge soundness)**.** $\mathsf{eval}, \mathsf{mult}$ *and* $\mathsf{eval}, \mathsf{uni}$ *are succinct AoKs for the relations* $\mathcal{R}_{\mathsf{eval},\mathsf{mult}}$ *and* $\mathcal{R}_{\mathsf{eval},\mathsf{uni}}$ *respectively defined as follows:*

$$\begin{aligned} \mathcal{R}_{\mathsf{eval},\mathsf{mult}} = \{ \big((\mathsf{pp}, C_{\mathbf{a}}, \mathbf{x} \in \mathbb{F}^d, \ y \in \mathbb{F}); \ \mathbf{a} \big) \ : \\ (C_f, C_{\mathbf{a}}) \leftarrow \mathsf{commit}(\mathsf{pp}, \mathbf{a})) \wedge y = \tilde{a}(\mathbf{x}) \} \end{aligned}$$

$$\begin{aligned} \mathcal{R}_{\mathsf{eval},\mathsf{uni}} = \{ ((\mathsf{pp}, C_f, u \in \mathbb{F}, \ v \in \mathbb{F}); \ \mathbf{a}) \ : \\ (C_f, C_{\mathbf{a}}) \leftarrow \mathsf{commit}(\mathsf{pp}, \mathbf{a})) \\ \wedge \quad v = f(u), \textit{where } \mathbf{a} \textit{ agrees with } f(\mathbf{H}_D) \} \end{aligned}$$

Definition 12 (Linking Soundness)**.** $\mathsf{prove\_link}$ *is a succinct argument for the relation* $\mathcal{R}_{\mathsf{link}}$ *defined as follows:*

$$\mathcal{R}_{\mathsf{link}} = \{ (\mathsf{pp}, C_f, C_{\mathbf{a}}; \ \mathbf{a}) : (C_f, C_{\mathbf{a}}) \leftarrow \mathsf{commit}(\mathsf{pp}, \mathbf{a})) \}$$

We give two dual polynomial commitment schemes in this section. The first DualPCS, denoted KZG-FFT-FOURIER, is given in Section 6.1. It requires an updatable setup and leverages the univariate and multilinear polynomial commitment schemes presented in Sections 3 and 4 respectively. The second DualPCS, denoted dory-link is given in Section 6.2. It requires a transparent setup and uses Dory to individually commit and prove evaluations of univariate and multilinear polynomials. Additionally, dory-link uses the argument system from Section 5 to prove linking soundness.

6.1 Dual Polynomial Commitment Scheme with Updatable Setup

Protocol 4: KZG-FFT-FOURIER

1. $\{\mathsf{srs}, \pi\} \leftarrow_R$ KZG-FFT-FOURIER.setup $(1^\lambda, N)$, where λ is security parameter.
2. $C_f, C_{\mathbf{a}} \leftarrow_R$ KZG-FFT-FOURIER.commit$(\mathsf{srs}_\mathcal{P}, D, \mathbf{a})$. Here, $C_{\mathbf{f}} = C_{\mathbf{a}} =$ KZG-FFT.commit$(\mathsf{srs}_\mathcal{P}, D, \mathbf{a})$
3. accept/reject $\leftarrow$ KZG-FFT-FOURIER.prove_link$(C_f, C_{\mathbf{a}}, D; \mathbf{a})$ is the trivial protocol $\langle P_{\mathsf{link}}, V_{\mathsf{link}}\rangle(C_f, C_{\mathbf{a}}, D; \mathbf{a})$ between P_{link} and V_{link}, where V_{link} checks $C_f = C_{\mathbf{a}}$.
4. KZG-FFT-FOURIER.eval_mult = KZG-FOURIER.eval $(\mathsf{srs}, C_{\mathbf{a}}, d, \mathbf{x}^{(d)}, y; \mathbf{a})$.
5. KZG-FFT-FOURIER.eval_uni = KZG-FFT.eval$(\mathsf{srs}, C_f, D, u, v; \mathbf{a})$

KZG-FFT-FOURIER is presented in Protocol 4. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow_R \mathcal{G}(1^\lambda)$ be a bilinear group. KZG-FFT-FOURIER. setup takes as input the security parameter λ, and N the degree bound on the set of univariate polynomials (or alternatively the bound on the number of multilinear monomials). The function internally calls KZG-FOURIER.Setup(λ, N) and returns its output $\{\mathsf{srs}, \pi\}$ as its output. It is important to note here that the srs returned by KZG-FOURIER.Setup(λ, n) is a subset of the srs returned by KZG-FFT.Setup(λ, N). Let $f \in \mathbb{F}_{<D}[Y]$ be such that $\mathbf{a}$ agrees with $f(\mathbf{H}_D)$. KZG-FFT-FOURIER.commit takes as input $\mathsf{srs}_\mathcal{P}$, $\mathbf{a} \in \mathbb{F}^D$, and commits to $f(Y)$ as $C_f =$ KZG-FFT. commit$(\mathsf{srs}_\mathcal{P}, D, \mathbf{a})$. It easily follows from the definition of KZG-FOURIER.commit that

$$\begin{aligned} C_{\mathbf{a}} &= \mathsf{KZG\text{-}FOURIER.commit}(\mathsf{srs}_\mathcal{P}, D, \mathbf{a}) \\ &= \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_\mathcal{P}, D, \mathbf{a}), \end{aligned}$$

where $C_{\mathbf{a}}$ is the commitment to the MLE $\tilde{a}$ of $\mathbf{a}$. Hence, KZG-FFT- FOURIER.prove_link$(C_f, C_{\mathbf{a}}, D; \mathbf{a})$ is the trivial protocol where the verifier just checks $C_f = C_{\mathbf{a}}$. KZG-FFT-FOURIER.eval_mult is the evaluation protocol employed by KZG-FOURIER, whereas KZG-FFT -FOURIER.eval_uni is the evaluation protocol employed by KZG-FFT. It is evident from Theorems 1 and 2 that KZG-FFT-FOURIER is a dual polynomial commitment scheme.

6.2 Dual Polynomial Commitment Scheme with a Transparent Setup

Protocol 5 states dory-link, a dual polynomial commitment scheme that leverages the argument system for linear relations from Section 5.2 to prove linking soundness. The public parameters (pp) is as generated by Algorithm 3 in Section 5.1, and the same notation is used below to refer to elements in pp. Here, we note that the public parameters output by Algorithm 3 also contains the public parameters of the Dory argument. The dory-link.commit algorithm computes $C_{\mathbf{a}}$ as defined in Step 2. To compute C_f, the algorithm computes the coefficient representation of f such that $\mathbf{a}$ agrees with $f(\mathbf{H}_D)$, and computes C_f as in Step 2. At Step 3, dory-link.prove_link$(C_f, C_{\mathbf{a}}, D;\ \mathbf{a})$ runs the argument system for proving linear relations in Section 5.2. Specifically, P_{link} and V_{link} is the PPT prover and verifier respectively from Protocol 3. Here,

we remark that it is sufficient to run only Step 2 of Protocol 3, as $C_{\mathbf{a}}$ would already be opened as part of dory-link.eval_mult. At Steps 4 and 5, to prove multilinear (resp. univariate) evaluations, Protocol 8 from Appendix D.3 is employed. In particular, Steps 4 and 5 are AoK's for the following relations respectively:

$$(C_{\mathbf{a}}, D, \mathbf{x}, y; \mathbf{a} \mid \exists \mathbf{a} \in \mathbb{F}^D, C_{\mathbf{a}} = \langle \tau^{(1,d)}, g_2^{\mathbf{a}} \rangle, \tilde{a}(\mathbf{x}) = y\}$$

$$(C_f, D, u, v; \mathbf{f} \mid \exists f \in \mathbb{F}_{<D}[Y], C_f = \langle \tau^{(1,d)}, g_2^{\mathbf{f}} \rangle, f(u) = v\}$$

From Theorem 3, it follows that Protocol 5 is a dual polynomial commitment scheme.

Protocol 5: Linking Proof using Argument System

1. $\mathsf{pp} \leftarrow \mathsf{dory\text{-}link.Setup}(1^\lambda, N)$, where λ is security parameter.
2. $C_f, C_{\mathbf{a}} \leftarrow_R \mathsf{dory\text{-}link.commit}(\mathsf{pk}, \mathbf{a}, D)$, where
$$C_{\mathbf{a}} = \prod_{i \in [0, D-1]} e(\tau_i^{(1,d)}, g_2^{a_i}), \text{ and } C_f = \prod_{i \in [0, D-1]} e(\tau_i^{(1,d)}, g_2^{f_i})$$
3. $\mathsf{dory\text{-}link.prove\_link}(C_f, C_{\mathbf{a}}, D; \mathbf{a})$ is defined as $\langle P_{\mathsf{lin}}, V_{\mathsf{lin}} \rangle\ (\mathsf{pp}, C_f, C_{\mathbf{a}}, D; \mathbf{a})$.
4. $\mathsf{dory\text{-}link.eval\_mult}(C_{\mathbf{a}}, \mathbf{x}, y; \mathbf{a})$ is defined as $\langle P_{\mathsf{dory\_eval}}, V_{\mathsf{dory\_eval}} \rangle (\mathsf{pp}, C_{\mathbf{a}}, D, \mathbf{x}, y; \mathbf{a})$. Here $\langle P_{\mathsf{dory\_eval}}, V_{\mathsf{dory\_eval}} \rangle$ is the Dory evaluation protocol restated in Protocol 8 in Appendix D.3.
5. $\mathsf{dory\text{-}link.eval\_uni}(C_f, u, v; \mathbf{f})$ is defined as $\langle P_{\mathsf{dory\_eval}}, V_{\mathsf{dory\_eval}}) \rangle (\mathsf{pp}, C_{\mathbf{a}}, D, u, v; \mathbf{a})$.

7 AIR for Grand-Product

In this section, we present a simple argument of knowledge for the grand-product relation which checks if the product of the elements in two distinct vectors are the same. The grand-product check is widely used as sub-components in bigger SNARKS to ensure that two distinct vectors are permutations of each other A prominent example of this is the offline memory checking procedure in Spartan [Set20], Lasso [STW23]. We present a dedicated algebraic intermediate representation (AIR) for the same and employ our KZG-FFT commitment scheme to check the AIR at random point. While this approach is simple, our dual polynomial commitment scheme KZG2-MPCS enables us to combine it with Spartan or Lasso, and obtain smaller proof sizes and better verifier time while trading-off prover time. The grand-product relation is formally stated as follows:

$$\mathsf{GPR} = \{(q \in \mathbb{F}; \mathbf{a} \in \mathbb{F}^N) \mid u = \prod_{i \in [0, N-1]} a_i)\}$$

Our protocol for GPR is given in Protocol 6. We state the proof system using KZG-FFT commitment scheme but any univariate commitment scheme can be used. At Steps 1-2, P_{GPR} computes the vector $\mathbf{a}' \in \mathbb{F}^D$ which iteratively computes q. It is easily seen that for an honest prover $a'_{D-1} = q = \prod_{i \in [0, N-1]} a_i$. The AIR enables the verifier to check $\mathbf{a}'$ is well-formed, and that the last value of $\mathbf{a}'$, a'_{D-1} is indeed the claimed product. Let $f_{\mathbf{a}}, f_{\mathbf{a}'} \in \mathbb{F}_{<D}[Y]$ such that $\mathbf{a}, \mathbf{a}'$ agrees with $f_{\mathbf{a}}(\mathbf{H}_D)$, $f_{\mathbf{a}'}(\mathbf{H}_D)$. Further, define the constraint polynomials f_1, f_2, f_3 as follows:

$$f_1(Y) = \frac{f_{\mathbf{a}'}(Y) - f_{\mathbf{a}}(Y)}{Y - 1}, f_2(Y) = \frac{f_{\mathbf{a}'}(Y) - q}{Y - \omega_D^{D-1}}$$

$$f_3(Y) = \frac{\left(f_{\mathbf{a}'}(Y) \cdot f_{\mathbf{a}}(\omega_D \cdot Y) - f_{\mathbf{a}'}(\omega_D \cdot Y)\right) \cdot \left(Y - \omega_D^{D-1}\right)}{Y^D - 1}$$

The polynomials $f_1(Y)$, $f_2(Y)$, and $f_3(Y)$ ensure that $\mathbf{a}'$ is well-formed. Specifically, $f_1(Y)$, $f_2(Y)$, and $f_3(Y)$ ensure that $a'_0 = a_0$, $a'_{D-1} = q$, and $a'_{i+1} = a_{i+1} \cdot a'_i$ for $i \in [0, D-2]$ respectively. Protocol 6 requires P_{GPR} at Step 2 to commit to univariate polynomials $f_{\mathbf{a}}, f_{\mathbf{a}'}$, and then at Step 4 to commit to f defined as a random linear combination of $f_1(Y)$, $f_2(Y)$, and $f_3(Y)$ as follows:

$$f(Y) = \sum_{i \in [1,3]} \gamma_i \cdot f_i(Y) \tag{18}$$

Protocol 6: AoK for GPR

$\{\mathsf{srs}, \pi\} \leftarrow_R \mathsf{KZG\text{-}FFT.setup}(1^\lambda, N)$, where λ is security parameter.
$\langle P_{\mathsf{GPR}}, V_{\mathsf{GPR}} \rangle (q; \mathbf{a} \in \mathbb{F}^D, D \leq N)$

1. P_{GPR}: P_{GPR} computes a vector $\mathbf{a}' \in F^D$ such that $a'_0 = a_0$, and $a'_i = a_{i-1} \cdot u_i$ for $i \in [1, D-1]$.
2. $P_{\mathsf{GPR}} \rightarrow V_{\mathsf{GPR}}$: P_{GPR} computes $C_{\mathbf{a}} = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}, D, \mathbf{a})$, $C_{\mathbf{a}'} = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}, D, \mathbf{a}')$, and sends it to V_{GPR}
3. $V_{\mathsf{GPR}} \rightarrow P_{\mathsf{GPR}}$: Samples three values $\gamma_1, \gamma_2, \gamma_3 \in_r \mathbb{F}$ and sends it to P_{GPR}.
4. $P_{\mathsf{GPR}} \rightarrow V_{\mathsf{GPR}}$: Compute the evaluations of constraint polynomials $f \in \mathbb{F}_{<D}[Y]$ (see Equation 18) over the FFT domain of size D. Commits to f with $C_f = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}, D, f(\mathbf{H}_D))$.
5. $V_{\mathsf{GPR}} \rightarrow P_{\mathsf{GPR}}$: Samples $u \in_R \mathbb{F}$.
6. $P_{\mathsf{GPR}} \rightarrow V_{\mathsf{GPR}}$: Send $f(u)$, $f_{\mathbf{a}}(u)$, $f_{\mathbf{a}'}(u)$, $f_{\mathbf{a}}(\omega_D \cdot u)$, $f_{\mathbf{a}'}(\omega_D \cdot u)$ to V_{GPR}.
7. $V_{\mathsf{GPR}} \rightarrow P_{\mathsf{GPR}}$: Checks $f(u)$, $f_{\mathbf{a}}(u)$, $f_{\mathbf{a}'}(u)$, $f_{\mathbf{a}}(\omega_D \cdot u)$, $f_{\mathbf{a}'}(\omega_D \cdot u)$ satisfy Equation 19. If yes then samples $\delta_1, \delta_2, \delta_3 \in_R \mathbb{F}$, and sends it to P_{GPR}.
8. Let $h_1 \in \mathbb{F}_{<D}[Y]$ be as defined in Equation 20. $P_{\mathsf{GPR}}, V_{\mathsf{GPR}}$ run $\mathsf{KZG\text{-}FFT.eval}(\mathsf{srs}, C_h, D, u, v_1)$, where v_1 is claimed value of h_1 at u.
9. Let $h_2 \in \mathbb{F}_{<D}[Y]$ be as defined in Equation 21. $P_{\mathsf{GPR}}, V_{\mathsf{GPR}}$ run $\mathsf{KZG\text{-}FFT.eval}(\mathsf{srs}, C_h, D, \omega_D \cdot u, v_2)$, where v_2 is claimed value of h_2 at $\omega_D \cdot u$.

At the end of Step 4, it is easily seen that $f(Y) \in \mathbb{F}_{<D}[Y]$ and $f(Y)$ is as defined in Equation 18 if and only if $\mathbf{a}'$ is well-formed. We remark here that in order to commit to f, P_{GPR} has to compute $f(\mathbf{H}_D)$, which is computationally intensive. This requires the prover to first compute $f_{\mathbf{a}}(Y)$ and $f_{\mathbf{a}'}(Y)$ at the offset FFT domain of size D, and use it to compute $f(Y)$ at the offset FFT domain, and then derive $f(\mathbf{H}_D)$ from it. The remaining steps of the protocol help V_{GPR} to check with high probability that $f(Y) \in \mathbb{F}_{<D}[Y]$ and $f(Y)$ is as defined in Equation 18. At Step 7, V_{GPR} checks

$$\begin{aligned} f(u) = \gamma_1 \cdot \frac{f_{\mathbf{a}'}(u) - f_{\mathbf{a}}(u)}{u - 1} + \gamma_2 \cdot \frac{f_{\mathbf{a}'}(u) - q}{u - \omega_D^{D-1}} + \\ \gamma_3 \cdot \frac{\left(f_{\mathbf{a}'}(u) \cdot f_{\mathbf{a}}(\omega_D \cdot u) - f_{\mathbf{a}'}(\omega_D \cdot u)\right) \cdot \left(u - \omega_D^{D-1}\right)}{u^D - 1} \end{aligned} \tag{19}$$

Using Schwartz-Zippel, the above check ensures $f(Y)$ satisfies Equation 18 with high probability over the random choice of u. At Steps 8-9 V_{GPR} checks the claimed values sent by P_{GPR} at Step 6. This is done by batching checks corresponding to polynomials evaluated at the same point. To this end $P_{\mathsf{GPR}}, V_{\mathsf{GPR}}$ run $\mathsf{KZG\text{-}FFT.eval}$ corresponding to polynomials h_1, and h_2 defined as follows.

$$h_1(Y) = \delta_1 \cdot f_{\mathbf{a}'}(Y) + \delta_2 \cdot f_{\mathbf{a}}(Y) + \delta_3 \cdot f(Y) \tag{20}$$

$$h_2(Y) = \delta_1 \cdot f_{\mathbf{a}'}(Y) + \delta_2 \cdot f_{\mathbf{a}}(Y) \tag{21}$$

V_{GPR} can compute the commitments to h_1, and h_2, which follows from Lemma 4

$$C_{h_1} = C_{\mathbf{a}}^{\delta_1} \cdot C_{\mathbf{a}'}^{\delta_2} \cdot C_f^{\delta_3} \ , \ C_{h_2} = C_{\mathbf{a}}^{\delta_1} \cdot C_{\mathbf{a}'}^{\delta_2}$$

V_{GPR} can also compute the claimed values v_1, and v_2 as follows

$$v_1 = h_1(u) = \delta_1 \cdot f_{\mathbf{a}'}(u) + \delta_2 \cdot f_{\mathbf{a}}(u) + \delta_3 \cdot f(u)$$

$$v_2 = h_2(\omega_D \cdot u) = \delta_1 \cdot f_{\mathbf{a}'}(\omega_D \cdot u) + \delta_2 \cdot f_{\mathbf{a}}(\omega_D \cdot u)$$

Finally, from the proof of Theorem 1, we have that if V_{GPR} accepts at Steps 8, and 9 then with high probability $f_{\mathbf{a}}, f_{\mathbf{a}'}, f \in \mathbb{F}_{<D}[Y]$ and that their corresponding claimed values are correct.

8 Implementation

In this section, we discuss the concrete costs of our dual commitment schemes and present an application of this notion using our AoK for grand-product check from Section 7. In Section 8.1, we present the relevant metrics of KZG-FFT-FOURIER and dory-link, and also compare it with other relevant commitment schemes. We provide a reference implementation of KZG-FFT-FOURIER, and dory-link in Rust. Our implementation is based on the BLS12-381 curve. In Section 8.2, we discuss our implementation of the argument of knowledge for the grand-product check we presented in Section 7, and compare it to the grand-product check using the techniques in [GKR08, Tha13]. Finally, we integrate our grand-product check with our implementation of Spartan for Extended-R1CS from [KST22] to obtain reduced proof complexity and verifier complexity for the same. This is important for the actual deployment of its verifier using a smart contract on ethereum. We remark here that since spartan requires a multilinear commitment scheme, and our grand-product check requires a univariate commitment scheme, this integration requires a dual polynomial commitment scheme like KZG-FFT-FOURIER.

All measurements are taken on **QCT RACK Mount server** with, **256GB RAM** and **46 cores**. Throughout we report numbers for verifier on single core, and for prover on multiple cores. Our code is available at the following link [git24]. We use the efficient bilinear group called BLS12-381 field. Specifically, the proof systems are simulated over the BLS12-381 Scalar field which is a 255 bit prime field, and SXDH is known to be at least 128 bits hard over this bilinear group.

8.1 Commitment Schemes

Witness size	Commit(s)	Eval Prover		Eval Verifier		Eval Proof size(KB)	
		Uni(s)	Mult(s)	Uni(ms)	Mult(ms)	Uni(KB)	Mult(KB)
2^{15}	0.26	0.29	7.21	19.83	60.21	0.12	5.56
2^{16}	0.45	0.55	21.75	19.89	79.46	0.12	5.93
2^{17}	0.60	0.79	73.99	19.91	96.10	0.12	6.31
2^{18}	1.08	1.32	282.35	19.88	125.38	0.12	6.68
2^{19}	2.33	3.06	1096.54	19.86	244.70	0.12	7.06
2^{20}	3.74	5.04	4616.05	19.86	387.71	0.12	7.43

Table 2: Performance of KZG-FFT-FOURIER

Tables 2 and 3 give the commitment times, proof sizes, evaluation prover and verifier times corresponding to KZG-FFT-FOURIER, and dory-link. Figure 4 states the prover and verifier times, and the proof complexity for the AoK corresponding to linear relations from Section 7 which is used to prove linking soundness in

Witness size	Commit		Eval Prover		Eval Verifier		Eval Proof size(KB)	
	Uni(ms)	Mult(ms)	Uni(s)	Mult(s)	Uni(ms)	Mult(ms)	Uni	Mult
2^9	43.78	41.59	3.76	3.79	756.69	737.45	35.75	35.75
2^{10}	52.12	53.95	5.68	5.93	816.74	812.71	39.68	39.68
2^{11}	67.19	62.22	10.49	10.47	909.61	887.17	43.62	43.62
2^{12}	88.92	79.18	19.57	19.70	967.61	854.22	47.56	47.56
2^{13}	125.80	116.66	37.84	38.19	1046.92	842.04	51.50	51.50
2^{14}	171.43	178.84	74.00	74.31	1120.45	878.97	55.43	55.43
2^{15}	298.69	279.46	147.22	147.37	1187.06	902.49	59.37	59.37

Table 3: Performance of dory-link

case of dory-link. We report the setup generation time and setup size for both KZG-FFT, KZG-FOURIER and dory-link in Figures 1, 2 and 3. We note here that for KZG-FFT-FOURIER the commitment is the same for both the univariate and multilinear polynomials (see Section 3 and Section 4. In particular, the total time to commit the witness in this case is equal to generating one of them and not their addition as in the case of dory-link.

Degree	Setup time(s)		Setup size(MB)	
	KZG-FFT	KZG	KZG-FFT	KZG
2^{15}	0.74	0.62	6	3
2^{16}	1.42	1.19	12	6
2^{17}	2.60	2.36	24	12
2^{18}	4.83	4.38	48	24
2^{19}	9.47	8.51	96	48
2^{20}	18.94	16.80	192	96

Figure 1: Setup Generation of KZG and KZG-FFT

Degree	Setup time(s)		Setup size(MB)	
	KZG	FOURIER	KZG	FOURIER
2^{15}	0.64	1.72	3	6
2^{16}	1.20	4.20	6	12
2^{17}	2.39	12.81	12	24
2^{18}	4.42	44.78	24	48
2^{19}	8.62	167.81	48	96
2^{20}	17.04	643.42	96	192

Figure 2: Setup Generation of multilinear KZG and KZG-FOURIER

Degree Bound	Setup time(sec)	Setup size(MB)
2^9	3.06	0.24
2^{10}	6.81	0.48
2^{11}	17.95	0.95
2^{12}	53.68	1.89
2^{13}	175.98	3.76
2^{14}	618.40	7.51
2^{15}	2227.39	15.02

Figure 3: Linking Proof System Setup time and Setup size

Witness size	Prover time(s)	Verifier time(sec)	Proof size(KB)
2^9	4.55	1.05	46.12
2^{10}	7.26	1.18	51.18
2^{11}	12.52	1.27	56.25
2^{12}	23.54	1.38	61.31
2^{13}	45.59	1.48	66.37
2^{14}	89.29	1.59	71.43
2^{15}	177.43	1.68	76.50

Figure 4: Performance of the AoK for Linear Relations

We also compare the performance of KZG-FFT with KZG, and KZG-FOURIER with multilinear KZG in Tables 4 and 5.

KZG-FFT vs KZG: For polynomials of degree 2^{20}, the commitment time of KZG-FFT is at least 1.5x better while having remaining parameters almost same. The setup time KZG-FFT is mostly comparable to KZG but setup size is 2x.

KZG-FOURIER vs multilinear KZG: Performance of KZG-FOURIER on all fronts is worse compared to multilinear KZG. Additionally, the evaluation prover, concretely has huge prover time and is one of the bottlenecks in KZG-FFT-FOURIER dual polynomial commitment scheme.

Witness size	Commitment time(s)		Prover time(s)		Verifier time(ms)		Proof size(KB)	
	KZG-FFT	KZG	KZG-FFT	KZG	KZG-FFT	KZG	KZG-FFT	KZG
2^{15}	0.26	0.29	0.31	0.25	19.83	19.51	0.12	0.12
2^{16}	0.45	0.55	0.51	0.37	19.89	19.66	0.12	0.12
2^{17}	0.60	0.79	0.87	0.71	19.91	19.71	0.12	0.12
2^{18}	1.08	1.32	1.35	1.30	19.88	19.69	0.12	0.12
2^{19}	2.33	3.06	2.97	2.86	19.86	19.59	0.12	0.12
2^{20}	3.74	5.04	5.15	4.61	19.86	19.67	0.12	0.12

Table 4: Comparison of KZG vs KZG-FFT

Witness size	Commitment time(s)		Prover time(s)		Verifier time(ms)		Proof size(KB)	
	kzg	fourier	kzg	fourier	kzg	fourier	kzg	fourier
2^{15}	0.26	0.25	0.42	7.21	182.66	44.48	1.43	5.56
2^{16}	0.41	0.39	0.72	21.75	276.40	47.77	1.53	5.93
2^{17}	0.92	0.60	1.24	73.99	300.46	68.63	1.62	6.31
2^{18}	1.21	1.73	2.00	282.35	299.43	93.42	1.71	6.68
2^{19}	2.51	2.77	2.91	1096.54	308.66	152.23	1.81	7.06
2^{20}	4.23	4.65	4.23	4616.05	317.92	243.04	1.90	7.43

Table 5: Comparison of multilinear KZG vs KZG-FOURIER

8.2 Spartan using Grand-Product AIR

It is well-known how to perform grand-product checks using specialized GKR [GKR08, Tha13] for circuits consisting of only multiplication gates. Additionally, [SL20] introduced a dedicated sum-check based argument system to perform grand-product checks. The argument system from both the above approaches require the witness vector to be committed as multilinear polynomial, that is, the witness vector is viewed as the evaluations of a multilinear polynomial over a boolean hypercube. Further, even though we employ verifier efficient commitment schemes with these proof systems the proof sizes are poly-logarithmic in the case of [GKR08, Tha13] and logarithmic in case of [SL20], owing to the proofs corresponding to the relevant sum-checks. Our AoK for grand-product check presented in Section 7 requires only constant size proofs. We concretely compare the performance of our AoK for grand-product check with the proof system for grand-product check from [GKR08, Tha13] in Table 6. We employ multilinear KZG to commit to the witness in

Witness size	Eval Prover(sec)		Eval Verifier(ms)		Eval Proof size(KB)	
	GKR	AIR	GKR	AIR	GKR	AIR
2^{15}	1.59	2.83	262.84	155.19	15.59	0.94
2^{16}	2.64	5.03	275.49	158.72	17.62	0.94
2^{17}	3.70	6.80	285.88	164.91	19.78	0.94
2^{18}	6.33	14.29	301.04	198.31	22.06	0.94
2^{19}	12.47	32.59	306.60	285.91	24.46	0.94
2^{20}	22.22	48.68	320.98	428.31	27.00	0.94

Table 6: Metrics corresponding to Grand Product. GKR denotes Grand product using techniques from [GKR08, Tha13],and AIR denotes grand-product check using AoK from Section 7.

No Of Constraints	Eval Prover(sec)		Eval Verifier(s)		Eval Proof size(KB)	
	Spartan	Spartan AIR	Spartan	Spartan AIR	Spartan	Spartan AIR
2^{12}	10.80	7.71	0.76	0.55	49.10	29.26
2^{13}	11.38	13.66	0.81	0.56	54.53	30.98
2^{14}	13.57	25.45	0.87	0.56	60.22	32.70
2^{15}	16.84	57.53	0.92	0.57	66.16	34.42
2^{16}	21.56	153.94	1.00	0.60	72.35	36.14
2^{17}	27.76	497.75	1.10	0.64	78.78	37.85

Table 7: Metrics comparing Spartan and Spartan-AIR. Spartan denotes spartan with grand-product check using [GKR08, Tha13], and Spartan AIR denotes spartan with grand-product check using the AoK from Section 7. The ratio of sparsity to constraints in the R1CS matrices is maintained to two.

latter. For vectors of length 2^{20}, the prover of GKR is 2.2x faster compared to our AoK, whereas the verifier of our AoK is 1.32x faster compared to GKR. As noted before the proof size of GKR grows poly-logarithmically and is 27 KB at vector lengths of 2^{20}, whereas the proof size of our AoK is a constant at 0.94 KB.

Grand-product checks as noted before are widely used in many argument systems to prove two vectors are permutations of each other. A special instance of this is how the spartan proof system employs the grand-product check as part of offline memory check in sparse multilinear polynomial evaluations. In Table 7, we compare the performance of spartan with grand-product check done using GKR, and spartan with grand-product check done using the AoK from Section 7. The Extended R1CS instance for the simulation is generated similar to [Set20]. The ratio of the sparsity of the matrices to the number of constraints is fixed to 2. We remark that since the AoK from Section 7, requires univariate commitments to the witness, such an integration is only possible with a dual polynomial commitment scheme like KZG-FFT-FOURIER. For constraints equal to 2^{17} Spartan-AIR trades off prover time to get 1.7x better verifier time and at least 2x improvement in proof size.

References

[ABC+22] Diego F. Aranha, Emil Madsen Bennedsen, Matteo Campanelli, Chaya Ganesh, Claudio Orlandi, and Akira Takahashi. ECLIPSE: Enhanced compiling method for pedersen-committed zkSNARK engines. pages 584–614, 2022.

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. pages 209–236, 2010.

[AST23] Arasu Arun, Srinath Setty, and Justin Thaler. Jolt: Snarks for virtual machines via lookups. Cryptology ePrint Archive, Paper 2023/1217, 2023. https://eprint.iacr.org/2023/1217.

[BCG+18] Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune K. Jakobsen, and Mary Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execution. pages 595–626, 2018.

[BCHO22] Jonathan Bootle, Alessandro Chiesa, Yuncong Hu, and Michele Orrù. Gemini: Elastic SNARKs for diverse environments. pages 427–457, 2022.

[CFF+21] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. pages 3–33, 2021.

[CFQ19] Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. pages 2075–2092, 2019.

[EFG22] Liam Eagen, Dario Fiore, and Ariel Gabizon. cq: Cached quotients for fast lookups. Cryptology ePrint Archive, Paper 2022/1763, 2022. https://eprint.iacr.org/2022/1763.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. pages 33–62, 2018.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. pages 186–194, 1987.

[git24] Dual_PCS. https://github.com/arithmic/Dual_PCS.git, 2024.

[GKM+18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. pages 698–728, 2018.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. pages 113–122, 2008.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). pages 291–304, 1985.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. pages 305–326, 2016.

[GW20] Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Paper 2020/315, 2020. https://eprint.iacr.org/2020/315.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Paper 2019/953, 2019. https://eprint.iacr.org/2019/953.

[KST22] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. pages 359–388, 2022.

[KT23] Tohru Kohrita and Patrick Towa. Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments. Cryptology ePrint Archive, Paper 2023/917, 2023. https://eprint.iacr.org/2023/917.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. pages 177–194, 2010.

[Lee21] Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. pages 1–34, 2021.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. pages 2111–2128, 2019.

[PST13] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. pages 222–242, 2013.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. pages 704–737, 2020.

[SL20] Srinath Setty and Jonathan Lee. Quarks: Quadruple-efficient transparent zksnarks. Cryptology ePrint Archive, Paper 2020/1275, 2020. https://eprint.iacr.org/2020/1275.

[STW23] Srinath Setty, Justin Thaler, and Riad Wahby. Unlocking the lookup singularity with lasso. Cryptology ePrint Archive, Paper 2023/1216, 2023. https://eprint.iacr.org/2023/1216.

[Tha13] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. pages 71–89, 2013.

[Whi] Barry Whitehat. Lookup singularity.

[WTS+18] Riad S Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zksnarks without trusted setup. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 926–943. IEEE, 2018.

[ZBK+22] Arantxa Zapico, Vitalik Buterin, Dmitry Khovratovich, Mary Maller, Anca Nitulescu, and Mark Simkin. Caulk: Lookup arguments in sublinear time. Cryptology ePrint Archive, Paper 2022/621, 2022. `https://eprint.iacr.org/2022/621`.

[ZGK+22] Arantxa Zapico, Ariel Gabizon, Dmitry Khovratovich, Mary Maller, and Carla Ràfols. Baloo: Nearly optimal lookup arguments. Cryptology ePrint Archive, Paper 2022/1565, 2022. `https://eprint.iacr.org/2022/1565`.

A Univariate PCS KZG-FFT

A.1 Proof of Lemmas and Claims from Section 3

Lemma 4. *Let $h_{1,i}^{(d)}$ for $d \in [1,n]$ be as defined in Protocol 1, and Equation 1. Further, let f and $\mathbf{a}$ be as defined above. Then $C_f = \prod_{i\in[0,D-1]}(h_{1,i}^{(d)})^{a_i} = g_1^{f(r^{2^{n-d}})}$.*

Proof. Let $D = 2^d$ and f_i be the coefficient of Y^i in f, for $i \in [0, D-1]$, and $\mathbf{f} = (f_0, \ldots, f_{D-1})$. Then $M_{\omega_D} \cdot \mathbf{f} = f(\mathbf{H}_D) = \mathbf{a}$, and hence, from Fact 2

$$\mathbf{f} = M_{\omega_D}^{-1} \cdot \mathbf{a} = (\frac{1}{D} M_{\omega_D^{-1}}) \cdot \mathbf{a} \tag{22}$$

Let $\alpha^{(\mathbf{d})}$ be such that $g_1^{\alpha_i^{(d)}} = h_{1,i}^{(d)}$ for $i \in [0, D-1]$, and $\mathbf{r}^{(d)} \in \mathbb{F}^D$ be such that $r_i^{(d)} = r^{i\cdot 2^{n-d}}$ for $i \in [0, D-1]$. To prove the lemma it is sufficient to show that $f(r^{2^{n-d}}) = \langle \alpha^{(d)}, \mathbf{a}\rangle$. We use the following claim to complete the proof (proof below).

Claim A.1. $(\frac{1}{D} M_{\omega_D^{-1}}) \cdot \mathbf{r}^{(d)} = \alpha^{(d)}$

From the above claim, we have

$$\begin{aligned} f(r^{2^{n-d}}) &= (\mathbf{r}^{(d)})^T \cdot \mathbf{f} \\ &= (\mathbf{r}^{(d)})^T \cdot (\frac{1}{D} M_{\omega_D^{-1}}) \cdot \mathbf{a} && \text{from Equation 22} \\ &= (\alpha^{(d)})^T \cdot \mathbf{a} && \text{from Fact 2} \end{aligned}$$

□

Proof of Claim A.1. For $d = n$, we have from the definition of $\alpha_i^{(n)}$ in Algorithm 1 (α_i is identified with $\alpha_i^{(n)}$), for all $i \in [0, N-1]$

$$\alpha_i^{(n)} = \frac{1}{N} \prod_{j\in[0,n-1]} (1 + (\omega_N^{-i} r)^{2^j}) = \sum_{j\in[0,N-1]} (\omega_N^{-i} r)^j$$

Hence, we have $\alpha^{(\mathbf{n})} = \frac{1}{N} M_{\omega_N^{-1}} \cdot \mathbf{r}^{(n)}$ To prove this for any $d \in [1, n-1]$, we show $\alpha^{(\mathbf{d})} = \frac{1}{D} M_{\omega_D^{-1}} \cdot \mathbf{r}^{(d)}$ assuming $\alpha^{(\mathbf{d+1})} = \frac{1}{2^{d+1}} M_{\omega_{2D}^{-1}} \cdot \mathbf{r}^{(d+1)}$. In particular, we show that for all $i \in [0, D-1]$,

$$\alpha_i^{(d)} = \frac{1}{D} \prod_{j\in[0,d-1]} (1 + (\omega_D^{-i} r^{2^{n-d}})^{2^j})$$

assuming for all $i \in [0, 2D-1]$,

$$\alpha_i^{(d+1)} = \frac{1}{2D} \prod_{j\in[0,d]} (1 + (\omega_{2D}^{-i} r^{2^{n-d-1}})^{2^j})$$

By definition of $h_i^{(d)}$, we have for all $i \in [0, D-1]$,

$$\begin{aligned}
\alpha_i^{(d)} &= \alpha_i^{(d+1)} + \alpha_{i+D}^{(d+1)} \\
&= \frac{1}{2D}\big(\big(\prod_{j\in[0,d]} (1+\omega_{2D}^{-i} r^{2^{n-d-1}})^j\big) \;+ \\
&\qquad \big(\prod_{j\in[0,d]} (1+\omega_{2D}^{-i-2^d} r^{2^{n-d-1}})^j\big)\big) \\
&= \frac{1}{2D}\big(\big(\prod_{j\in[0,d]} (1+\omega_{2D}^{-i} r^{2^{n-d-1}})^j\big) \;+ \\
&\qquad \big(\prod_{j\in[0,d]} (1-\omega_{2D}^{-i} r^{2^{n-d-1}})^j\big)\big)
\end{aligned}$$

$$\begin{aligned}
\alpha_i^{(d)} &= \frac{1}{2D} \cdot \big(\sum_{j=2\ell,\ell\in[0,D-1]} (\omega_{2D}^{-i} \cdot r^{2^{n-d-1}})^j \;+ \\
&\qquad \sum_{j=2\ell+1,\ell\in[0,D-1]} -(\omega_{2D}^{-i} \cdot r^{2^{n-d-1}})^j\big) \\
&= \frac{1}{D} \cdot \sum_{j\in[0,D-1]} (\omega_{2D}^{-2i} \cdot r^{2^{n-d}})^j \\
&= \frac{1}{D} \prod_{j\in[0,d-1]} (1+\omega_D^{-i} r^{2^{n-d}})^j
\end{aligned}$$

The last equality in the above sequence of equations follows from Fact 1. □

A.2 Proof of Theorem 1

Completeness follows from the discussion in Section 3.2. We first prove KZG-FFT.commit satisfies commitment binding, and then prove knowledge soundness.

Commitment Binding: Suppose KZG-FFT.commit does not satisfy commitment binding in the bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow_R \mathcal{G}(1^\lambda)$. Then there exists an adversary $\mathcal{A}$ that on input $(\mathsf{srs}\} \leftarrow_R$ KZG-FFT.Setup(λ), outputs $(f_0(\mathbf{H}_{D_0}), f_1(\mathbf{H}_{D_1}))$ with a probability non-negligible in λ, satisfying:

1. $f_0 \in \mathbb{F}_{<D_0}[Y], f_1 \in \mathbb{F}_{<D_1}[Y]$ and $f_0(Y) \neq f_1(Y)$,
2. $C =$ KZG-FFT.commit $(\mathsf{srs}_\mathcal{P}, D_0, f_0(\mathbf{H}_{D_0})) =$ KZG-FFT.commit $(\mathsf{srs}_\mathcal{P}, D_1, f_1(\mathbf{H}_{D_1}))$.

We show that in this case an adversary $\mathcal{A}'$ can be constructed that when given input a) $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow_R \mathcal{G}(1^\lambda)$ and b) $\mathsf{srs}' = \{(\mathsf{srs}'_\mathcal{P}, \mathsf{srs}_\mathcal{V})\}$, where $\mathsf{srs}'_\mathcal{P} = \{g_1^{r^i}\}_{i\in[0,N-1]}$ for some $r \in_R \mathbb{F}$, and $\mathsf{srs}_\mathcal{V} = \{g_2^{r^{2^j}}\}_{j\in[0,n]}$. $\mathcal{A}'$ outputs the secret random value r with a probability non-negligible in λ, thereby breaking N-DLOG assumption. Here, we remark that $\mathsf{srs}'_\mathcal{P}$ is different from $\mathsf{srs}_\mathcal{P}$, and is as in the N-DLOG challenge. $\mathcal{A}'$ first computes $\mathsf{srs}_\mathcal{P}$ (where $\mathsf{srs}_\mathcal{P}$ is as defined in Algorithm 1). From Claim 3.1, we have that $\alpha = \frac{1}{N} M_{\omega_N^{-1}} \cdot \mathbf{r}$. Hence, the i-th element of $\mathsf{srs}_\mathcal{P}$ can be computed by taking MSME with group base elements being $\mathsf{srs}'_\mathcal{P}$ and the scalars being the i-th row of $\frac{1}{N} M_{\omega_N^{-1}}$. In particular, $\mathsf{srs}_\mathcal{P}$ can be computed from $\mathsf{srs}'_\mathcal{P}$ using at most N^2 exponentiations and additions over the group $\mathbb{G}_1$. $\mathcal{A}'$ lets $\mathsf{srs} = \{(\mathsf{srs}_\mathcal{P}, \mathsf{srs}_\mathcal{V})\}$, and calls $\mathcal{A}$ on input $\{(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{srs}\}$. $\mathcal{A}$ outputs $(f_0(\mathbf{H}_{D_0}), f_1(\mathbf{H}_{D_2}))$ satisfying the two properties stated above.

Let $D = \max(D_0, D_1)$. By interpolating and evaluating the lower degree polynomial, $\mathcal{A}'$ can compute $f_0(\mathbf{H}_D), f_1(\mathbf{H}_D)$. Let $d = \log D$, and let $\mathbf{a}_0$ and $\mathbf{a}_1$ agree with $f_0(\mathbf{H}_D)$ and $f_1(\mathbf{H}_D)$. Since $f_0(Y) \neq f_1(Y)$, we have $\mathbf{a}_0 \neq \mathbf{a}_1$, and $\mathbf{a}_0 - \mathbf{a}_1 \neq \mathbf{0}$, where $\mathbf{0}$ is the zero vector in $\mathbb{F}^D$. Let $\mathbf{b} = \mathbf{a}_0 - \mathbf{a}_1$, $\mathbf{f}' = (\frac{1}{D} M_{\omega_D^{-1}}) \cdot \mathbf{b}$, and $f' \in \mathbb{F}_{<D}[Y]$ be the polynomial whose coefficient corresponding to Y^i is f'_i for $i \in [0, D-1]$. We show in Lemma 5 below that $r^{2^{n-d}}$ is a root of $f'(Y)$.

Lemma 5. *Let $f' \in \mathbb{F}_{<D}[Y]$ be as defined above, and $r \in \mathbb{F}$ be the random point used to generate srs'. Then $f'(r^{2^{n-d}}) = 0$.*

Proof. From the definition of KZG-FFT.commit we have

$$\prod_{i \in [0, D-1]} (h_{1,i}^{(d)})^{a_{0,i}} = \prod_{i \in [0, D-1]} (h_{1,i}^{(d)})^{a_{1,i}}$$

Multiplying by the inverse (over $\mathbb{G}_1$) of the RHS in the above equation on both sides we have

$$\prod_{i \in [0, D-1]} (h_{1,i}^{(d)})^{b_i} = \prod_{i \in [0, dD-1]} (h_{1,i}^{(d)})^{0} \tag{23}$$

Let $\alpha^{(\mathbf{d})}$ be such that $g_1^{\alpha_i^{(d)}} = h_{1,i}^{(d)}$ for $i \in [0, D-1]$, and $\mathbf{r}^{(d)} \in \mathbb{F}^D$ be such that $r_i^{(d)} = r^{i \cdot 2^{n-d}}$ for $i \in [0, D-1]$. From Equation 23, $\langle \alpha^{(\mathbf{d})}, \mathbf{b} \rangle = 0$. Further from Claim A.1, $(\mathbf{r}^{(d)})^T \cdot (\frac{1}{D} M_{\omega_D^{-1}}) = \alpha^{(d)}$. This implies

$$(\mathbf{r}^{(d)})^T \cdot (\frac{1}{D} M_{\omega_D^{-1}}) \cdot \mathbf{b} = 0 \tag{24}$$

From the definition of $\mathbf{f}'$, we have

$$\mathbf{f}' = (\frac{1}{D} M_{\omega_D^{-1}}) \cdot \mathbf{b}$$

Since the coefficient of Y^i in $f'(Y)$ is f'_i for $i \in [0, D-1]$, from Equation 24, it follows that $r^{2^{n-d}}$ is a root of f'. □

It is well-known how to efficiently compute r, given a polynomial $f'(Y) \in \mathbb{F}_{<D}[Y]$ such that $r^{2^{n-d}}$ is a root of $f'(Y)$. We note this as a claim below, as we reuse it in knowledge soundness argument.

Claim A.2. *There is a $D^{O(1)}$ time algorithm that takes as input the coefficients of a polynomial $f'(Y) \in \mathbb{F}_{<D}[Y]$ such that $r^{2^{n-d}}$ is a root of $f'(Y)$, and outputs r.*

Proof. The algorithm employs root finding algorithm over $\mathbb{F}$ to find $r^{2^{n-d}}$, and then employs square-root finding algorithm over $\mathbb{F}$ recursively to find r. □

Knowledge Soundness: We show that KZG-FFT.eval$(\mathsf{srs}, C_f, D, u, v;\ f(\mathbf{H}_D))$ is knowledge sound for the relation

$$\{(\mathsf{srs}, (C_f, D, u, v)); f(\mathbf{H}_D) \mid\ f \in \mathbb{F}_{<D}[Y], f(u) = v, \\ \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_{\mathcal{P}}, D, f(\mathbf{H}_D)) = C_f\}$$

where the srs is updatable by the adversary. Let $\tilde{P}_{\mathsf{eval}}$ be the algebraic adversary that has access to an oracle as stated in Definition 3.3 [MBKM19]. Given input $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathsf{Gen}(1^\lambda)$ and an initial structured reference string $\perp$ (corresponding to degree N polynomials), let $\tilde{P}_{\mathsf{eval}}$ output (srs, C_f, D), and an $\mathbf{a} \in \mathbb{F}^N$ satisfying a) $C_f = \prod_{i \in [0, N-1]} (h_{1,i}^{(n)})^{a_i}$, b) In $\langle P_{\mathsf{eval}}, V_{\mathsf{eval}} \rangle (\mathsf{srs}, C_f, d, u, v; \mathbf{a}_f)$, where $u \in_R \mathbb{F}$, V_{eval} accepts with probability ϵ non-negligible in λ.

Here we note that $\perp$ is the trivial string corresponding to $r = 0$, and $\tilde{P}_{\mathsf{eval}}$ uses the oracle to update it to srs, and V_{eval} rejects if $\mathsf{srs} = \perp$. The oracle ensures that srs is well-formed, that is, there is an $r \in \mathbb{F}$ such that

$\mathsf{srs}_{\mathcal{P}} = \{g_1^{\alpha_i} \mid \alpha_i = N^{-1} \cdot \prod_{j\in[0,n-1]}(1 + (\omega_N^{-i} \cdot r)^{2^j}) \text{ for } i \in [0, N-1]\}$, and $\mathsf{srs}_{\mathcal{V}} = \{g_2^{r^{2^j}} \mid j \in [0,n]\}$.

Let $\mathcal{E}$ be the extractor with oracle access to an algebraic prover $\tilde{P}_{\mathsf{eval}}$. $\mathcal{E}$ invokes $\tilde{P}_{\mathsf{eval}}$ to receive $(\mathsf{srs}, C_f, D, u, v)$, and a representation $\mathbf{a} \in \mathbb{F}^N$ satisfying the above two conditions, where $u \in_R \mathbb{F}$. The extractor $\mathcal{E}$ queries $\tilde{P}_{\mathsf{eval}}$ at $(\mathsf{srs}, C_f, D, u, v)$ and with probability ϵ obtains C_q and a representation $\mathbf{b} \in \mathbb{F}^N$ such that a) $C_q = \prod_{i\in[0,N-1]}(h_{1,i}^{(n)})^{b_i}$, and b) $e(C_f \cdot g_1^{-v}, g_2) = e(C_q, h_2^{(d)} \cdot g_2^{-u})$. We prove the following claim which helps us in proving degree bound on f and q.

Claim A.3. *Let $\mathbf{a} \in \mathbb{F}^N$ and $C = \prod_{i\in[0,N-1]}(h_{1,i}^{(n)})^{a_i}$. There exists $k \in [0,n]$ such that $a_i = a_{i+j\cdot 2^k}$ for $i \in [0, 2^k - 1]$ and $j \in [0, 2^{n-k} - 1]$ if and only if there exists $f'_{<2^k}[Y]$ such that $C = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_{\mathcal{P}}, k, f(\mathbf{H}_{2^k}))$.*

Proof. Its follows from the definition of $\mathsf{KZG\text{-}FFT.commit}$, that

$$C = \prod_{i\in[0,2^k-1]} \Big(\prod_{j\in[0,2^{n-k}-1]} (h_{1,i+j\cdot 2^k}) \Big)^{a_i} = \prod_{i\in[0,2^k-1]} (h_{1,i}^{(k)})^{a_i}$$

In the above equality, we have used that $h_{1,i}^{(k)} = \prod_{j\in[0,2^{n-k}-1]} h_{1,i+j\cdot 2^k}$. Let $f(\mathbf{H}_{2^k})$ be such that $f(\mathbf{H}_{2^k})_i = a_i$ for $i \in [0, 2^k - 1]$. It follows that $f' \in \mathbb{F}_{<2^k}[Y]$ is such that $C = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_{\mathcal{P}}, k, f(\mathbf{H}_{2^k}))$. □

From the above claim and Lemma 4, it follows that there exists $k_1, k_2 \in [0,n]$ such that $f' \in \mathbb{F}_{<2^{k_1}}[Y]$, $q' \in \mathbb{F}_{<2^{k_2}}[Y]$ $C_f = g_1^{f'(r^{2^{n-k_1}})}$ and $C_q = g_1^{q'(r^{2^{n-k_2}})}$. Knowledge soundness of the protocol would follow from the following lemma.

Lemma 6. *Suppose $k_1, k_2 \in [0,n]$ such that $f' \in \mathbb{F}_{<2^{k_1}}[Y]$, $q' \in \mathbb{F}_{<2^{k_2}}[Y]$, $C_f = g_1^{f'(r^{2^{n-k_1}})}$ and $C_q = g_1^{q'(r^{2^{n-k_2}})}$. Then only one of the following holds:*

1. *$2^{k_1} \leq D$ and there exists $q'' \in \mathbb{F}_{<D}[Y]$ such that $f'(Y^{2^{d-k_1}}) - v = (Y-u)q''(Y)$*
2. *$N \geq 2^{k_1} > D$ and $Y^{2^{k_1-d}} - u$ divides $f'(Y) - v$.*

Proof. We prove the lemma using a case analysis. In each of the case either $f'(Y)$ and $q'(Y)$ satisfy the statement of the lemma or r is a root of a polynomial known to $\tilde{P}_{\mathsf{eval}}$. From Claim A.2, it follows that if r is a root of a polynomial known to $\tilde{P}_{\mathsf{eval}}$ then $\tilde{P}_{\mathsf{eval}}$ can compute r efficiently, and hence break the N-DLOG assumption. Hence, in our analysis we assume r is not the root of the polynomial that arises in each of the cases.

Case a: $2^{k_1} \leq D, 2^{k_2} \leq D$. This implies $n - k_1 \geq n - d$, and $n - k_2 \geq n - d$. Assume $n - k_1 \geq n - k_2$; the other case can be handled similarly. In this case it must be that either i) $f'(Y^{2^{d-k_1}}) - v = (Y-u)\cdot q'(Y^{2^{d-k_2}})$ or ii) r is a root of $f'(Y^{2^{d-k_1}}) - v - (Y-u)\cdot q'(Y^{2^{d-k_2}})$. Letting $q''(Y) = q'(Y^{2^{d-k_2}})$, we have $f'(Y^{2^{d-k_1}}) - v = (Y-u)q''(Y)$.

Case b: $2^{k_1} \leq D, 2^{k_2} > D$. In this case it must be that either i) $f'(Y^{2^{k_2-k_1}}) - v = (Y^{2^{k_2-d}} - u) \cdot q'(Y)$ or ii) r is a root of $f'(Y^{2^{k_2-k_1}}) - v - (Y^{2^{k_2-d}} - u) \cdot q'(Y)$. If

$$f'(Y^{2^{k_2-k_1}}) - v = (Y^{2^{k_2-d}} - u) \cdot q'(Y) \tag{25}$$

then we show that there exists $q''(Y)$ such that $q'(Y) = q''(Y^{2^{k_2-d}})$. Let $\gamma = k_2 - k_1$, and $\delta = k_2 - d$, and note that $\gamma - \delta = d - k_1 \geq 0$. Using polynomial division, we know there exists $r \in \mathbb{F}$ and $q' \in \mathbb{F}_{<N}[Y]$ such that

$$f'(Y^{2^{\gamma-\delta}}) = (Y-u)q''(Y) + r \tag{26}$$

Since $r \in \mathbb{F}$, substituting Y as Y^{2^δ} in the above equation, we have

$$f'(Y^{2^\gamma}) = (Y^{2^\delta} - u)q''(Y^{2^\delta}) + r \tag{27}$$

From Equations 25, and 27, we have $(Y^{2^\delta} - u)q'(Y) = (Y^{2^\delta} - u)q''(Y^{2^\delta}) + r$. This implies $Y^{2^\delta} - u$ divides r, and as $r = 0$, $q'(Y) = q''(Y^{2^\delta})$. Hence, Equation 25, can be rewritten as

$$f'(Y^{2^{\gamma-\delta}}) - v = (Y - u) \cdot q''(Y)$$
$$f'(Y^{2^{d-k_1}}) - v = (Y - u) \cdot q''(Y)$$

Case c: $2^{k_1} > D, 2^{k_1} \geq 2^{k_2}$. In this case it must be that either i) $f'(Y) - v = (Y^{2^{k_1-d}} - u) \cdot q'(Y^{2^{k_1-k_2}})$ or ii) r is a root of $f'(Y) - v = (Y^{2^{k_1-d}} - u) \cdot q'(Y^{2^{k_1-k_2}})$. This implies $Y^{2^{k_1-d}} - u$ divides $f'(Y) - v$.

Case d: $2^{k_1} > D, 2^{k_1} < 2^{k_2}$. In this case it must be that either i) $f'(Y^{2^{k_2-k_1}}) - v = (Y^{2^{k_2-d}} - u) \cdot q'(Y)$ or ii) r is a root of $f'(Y^{2^{k_2-k_1}}) - v - (Y^{2^{k_2-d}} - u) \cdot q'(Y)$. If

$$f'(Y^{2^{k_2-k_1}}) - v = (Y^{2^{k_2-d}} - u) \cdot q'(Y) \tag{28}$$

then we show that there exists $q''(Y)$ such that $q'(Y) = q''(2^{k_2-k_1})$. Let $\gamma = k_2 - k_1$, and $\delta = k_2 - d$, and note that, $\delta - \gamma = k_1 - d > 0$. Using polynomial division, we know there exists $r \in \mathbb{F}_{<2^{\delta-\gamma}}$ and $q'' \in \mathbb{F}_{<N}[Y]$ such that

$$f'(Y) = (Y^{2^\delta - \gamma} - u)q''(Y) + r(Y)$$

Substituting Y as Y^{2^γ} in the above equation, we have

$$f'(Y^{2^\gamma}) = (Y^{2^\delta} - u)q''(Y^{2^\gamma}) + r(Y^{2^\gamma}) \tag{29}$$

From Equations 28, and 29,

$$(Y^{2^\delta} - u) \cdot q'(Y) = (Y^{2^\delta} - u)q''(Y^{2^\gamma}) + r(Y^{2^\gamma})$$

This implies $Y^{2^\delta} - u$ divides $r(Y^{2^\gamma})$. But $r \in \mathbb{F}_{<2^{\delta-\gamma}}$, and hence degree of $r(Y^{2^\gamma})$ is less than 2^δ. Hence, $r(Y^{2^\gamma}) = 0$ implying $r(Y) = 0$, and $q'(Y) = q''(Y^{2^{k_2-k_1}})$. Hence, Equation 28, can be rewritten as

$$f'(Y^{2^{k_2-k_1}}) - v = (Y^{2^{k_2-d}} - u) \cdot q''(Y^{2^{k_2-k_1}})$$

Hence, we have $f'(Y) = Y^{2^{k_1-d}} \cdot q''(Y)$. This implies $Y^{2^{k_1-d}} - u$ divides $f'(Y) - v$. □

In Lemma 6, if (a) holds then let $f(Y) = f'(Y^{2^{d-k_1}}) \in \mathbb{F}_{<D}[Y]$. This implies $f(Y) - v = (Y - u) \cdot q'(Y)$, and hence $f(u) = v$. In this case from Claim A.3, it follows that $a_i = a_{i+j\cdot 2^{k_1}}$ for $i \in [0, 2^{k_1} - 1]$ and $j \in [0, 2^{n-k_1} - 1]$. Further $\mathbf{a}_{[:2^{k_1}]}$ agrees with $f'(\mathbf{H}_{2^{k_1}})$ over the FFT domain of size 2^{k_1}. Here $\mathbf{a}_{[:2^{k_1}]}$ denotes the first 2^{k_1} components of $\mathbf{a}$. Hence, $\mathcal{E}$ computes the coefficient representation of $f'(Y)$ from $f'(\mathbf{H}_{2^{k_1}})$ using FFT. Since $f(Y) = f'(Y^{2^{d-k_1}})$, $\mathcal{E}$ computes the coefficient representation of f from the coefficients of f', from which it computes $f(\mathbf{H}_D)$ again using FFT. Now we show that case (b) in Lemma 6 holds with probability $\frac{N}{2|\mathbb{F}|}$ over the choice of $u \in_R \mathbb{F}$. Let $f' \in \mathbb{F}_{<2^{k_1}}$ be as in Case (b) of Lemma 6, and $S = \{(u', v')) \mid Y^{2^{k_1-d}} - u' \text{ divides } f'(Y) - v'\}$. We argue in Lemma 7 (stated below) that $|S| \leq \frac{2^{k_1}}{2^{k_1-d}} = 2^d$. Hence, given C_f, such that $C_f = \mathsf{KZG\text{-}FFT.commit}(\mathsf{srs}_\mathcal{P}, 2^{k_1}, f'(\mathbf{H}_{2^{k_1}}))$, and u is sampled uniformly and independently at random from $\mathbb{F}$ then with probability at least $1 - \frac{D}{|\mathbb{F}|}$ case b does not hold. Hence, it follows that $\mathcal{E}$ is able to compute a valid witness with probability $\epsilon(1 - \frac{D}{|F|})$.

Lemma 7. *Let $2^{k_1} > D$, $f' \in \mathbb{F}_{<2^{k_1}}$, and $S = \{(u', v')) \mid Y^{2^{k_1-d}} - u'$ divides $f'(Y) - v'\}$. Then $|S| \leq \frac{2^{k_1}}{2^{k_1-d}} = 2^d$.*

Proof. First, we argue that for each $u' \in \mathbb{F}$ there exists at most one $v' \in \mathbb{F}$ such that $(u', v') \in S$. Suppose there exists $(u', v'_1) \in S$ and $(u', v'_2) \in S$. This implies there exists $q'_1 \in \mathbb{F}_{<D}[Y]$ and $q'_2 \in \mathbb{F}_{<D}[Y]$ such that

$$(Y^{2^{k_1-d}} - u') \cdot q'_1(Y) + v'_1 = (Y^{2^{k_1-d}} - u') \cdot q'_2(Y) + v'_2$$
$$(Y^{2^{k_1-d}} - u') \cdot q'_1(Y) = (Y^{2^{k_1-d}} - u') \cdot q'_2(Y) + v'_2 - v'_1$$

We subtract v'_1 from both sides to obtain the above equation. Since $(Y^{2^{k_1-d}} - u')$ divides LHS of the above equation, we have $v'_1 = v'_2$. Next, we argue for every $(u'_1, v'_1) \in S$ and $(u'_2, v'_2) \in S$, $v'_1 = v'_2$. If $(u'_1, v'_1) \in S$ and $(u'_2, v'_2) \in S$ then there exists $q'_1 \in \mathbb{F}_{<D}[Y]$ and $q'_2 \in \mathbb{F}_{<D}[Y]$ such that

$$(Y^{2^{k_1-d}} - u'_1) \cdot q'_1(Y) + v'_1 = (Y^{2^{k_1-d}} - u'_2) \cdot q'_2(Y) + v'_2 \tag{30}$$

If $k_1 - d > d$ then by observing the degree of the polynomials on LHS and RHS in the above equation we have $q'_1(Y) = q'_2(Y)$, $u'_1 = u'_2$ and $v'_1 = v'_2$. In fact in this case $|S| = 1$. Suppose $k_1 - d \leq d$. Then from polynomial division there exists $q''_2(Y) \in \mathbb{F}_{2^{d-(k_1-d)}}[Y]$ and $r(Y) \in \mathbb{F}_{<2^{k_1-d}}[Y]$ such that $q'_2(Y) = (Y^{2^{k_1-d}} - u'_1) \cdot q''_2(Y) + r(y)$. Substituting $q'_2(Y)$ in Equation 30

$$(Y^{2^{k_1-d}} - u'_1) \cdot (q'_1(Y) - (Y^{2^{k_1-d}} - u'_2) \cdot q''_2(Y)) + v'_1 = (Y^{2^{k_1-d}} - u'_2) \cdot r(Y) + v'_2$$

Since degree $r(Y)$ is less than 2^{k_1-d}, again observing the degree of the polynomials on LHS and RHS in the above equation we have either a) $r(Y) = 0$, $q'_1(Y) = (Y^{2^{k_1-d}} - u'_2) \cdot q''_2(Y)$, and $v'_1 = v'_2$, or b) $r(Y) = 0$, $q'_1(Y) = (Y^{2^{k_1-d}} - u'_2) \cdot q''_2(Y)$, $u'_1 = u'_2$ and $v'_1 = v'_2$. The argument above shows that all tuples in S have the same second element. Let $S = \{(u_j, v)\}_{j \in |S|}$. Then for all $j \in [1, |S|]$, $Y^{2^{d_1-k}} - u_j$ divides $f'(Y) - v$. Since degree of $f'(Y)$ is at most 2^k_1, $|S| \leq \frac{2^{k_1}}{2^{k_1-d}} = 2^d$. □

A.3 Updatability of the SRS in KZG-FFT

In this section, we show that the setup of KZG-FFT is updatable as defined in Definition 3.3, [MBKM19]. To this end we state the update_setup, and verify_setup in Algorithms 4, and 5 respectively. KZG-FFT.update_setup in Algorithm 4, takes as input the bilinear group $\{(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ and srs, $\{\pi_i\}_{i \in [0,\ell]}\}$, and outputs a new structured reference string srs′, and appends a proof $\pi_{\ell+1}$ to the list of existing proofs $\{\pi_i\}_{i \in [0,\ell]}$. KZG-FFT.verify_setup in Algorithm 5, takes as input $\{(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, srs, $\{\pi_i\}_{i \in [0,\ell]}\}$ and outputs either accept or reject depending on whether the srs is correctly formed. We prove the correctness of the update_setup, and verify_setup algorithms in Lemmas 8, and 9 respectively.

Lemma 8. *Let the inputs to* update_setup *be as in Algorithm 4. Let* $\mathsf{srs}_{\mathcal{P}} = \{h^{(n)}_{1,i}\}_{i \in [0,n-1]}$, $\mathsf{srs}_{\mathcal{V}} = \{h_{2,j}\}_{j \in [0,n]}$, *and* $\pi_i = (u_{1,i}, u_{2,i})$ *for* $i \in [0, \ell]$. *Suppose there is an* $r \in \mathbb{F}$ *such that a)* $h^{(n)}_{1,i} = g_1^{\alpha_i}$, *where* $\alpha_i = N^{-1} \cdot \prod_{j \in [0,n-1]} (1 + (\omega_N^{-i} \cdot r)^{2^j})$ *for* $i \in [0, N-1]$, *b)* $h_{2,j} = g_2^{r^{2^j}}$ *for* $j \in [0, n]$, *and c)* $u_{2,\ell} = g_2^r$. *Then Algorithm 4, outputs* srs′ *and* $\pi_{\ell+1}$ *such that a)* $h'_{1,i} = g_1^{\alpha'_i}$, *where* $\alpha'_i = N^{-1} \cdot \prod_{j \in [0,n-1]} (1 + (\omega_N^{-i} \cdot r \cdot r_{\ell+1})^{2^j})$ *for* $i \in [0, N-1]$, *b)* $h_{2,j} = g_2^{(r \cdot r_{\ell+1})^{2^j}}$ *for* $j \in [0, n]$, *and c)* $u_{2,\ell} = g_2^{r \cdot r_{\ell+1}}$.

Proof. At Step 3, the i-th component of $\widehat{\mathbf{h}}_1$, denoted $\widehat{h}_{1,i}$, is equal to $\prod_{j=[0,N-1]} (h^{(n)}_{1,i})^{\omega_N^{i \cdot j}}$. From the supposition, Fact 2, and Claim 3.1, it follows that there exists $r \in \mathbb{F}$ such that $\widetilde{h}_{1,i} = g_1^{(r \cdot r_{\ell+1})^i}$ for $i \in [0, N-1]$. Hence, at Step 9, $h'_{1,i} = g_1^{\alpha'_i}$, where $\alpha'_i = N^{-1} \cdot \prod_{j \in [0,n-1]} (1 + (\omega_N^{-i} \cdot r \cdot r_{\ell+1})^{2^j})$ for $i \in [0, N-1]$. Points (b) and (c) follow easily from the construction. □

We note that since M_{ω_N} is the FFT matrix of size N, the recursive decomposition of the FFT matrix can be exploited to compute $\widehat{\mathbf{h}}_1$ using $N \log N$ group additions and $N \log N$ group exponentiations. Similarly $\mathbf{h}'_1$ can be computed at Step 9 using $N \log N$ group additions and $N \log N + N$ group exponentiations.

Algorithm 4 KZG-FFT.update_setup: Updating Setup for KZG-FFT

Input: $\{(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)\mathsf{srs}, \{\pi_i\}_{i\in[0,r]}\}$
Output: $\{\mathsf{srs}', \{\pi_i\}_{i\in[0,\ell+1]}\}$

Read π_i as $(u_{1,i}, u_{2,i}) \in (\mathbb{G}_1, \mathbb{G}_2)$ for $i \in [0, \ell]$, $\mathsf{srs}_{\mathcal{P}} = \{h_{1,i}^{(n)} \mid i \in [0, N-1]\}$, and $\mathsf{srs}_{\mathcal{V}} = \{h_{2,j} \mid j \in [0, n]\}$.
Let $\mathbf{h}_1 \in \mathbb{G}_1^N$ be such that the i-th component of $\mathbf{h}_1$ is $h_{1,i}^{(n)}$.
Compute $\widehat{\mathbf{h}}_1 = M_{\omega_N} \cdot \mathbf{h}_1$.
Sample $r_{\ell+1} \in \mathbb{F}$, uniformly at random.
for $i \in [0, N-1]$ **do**
    Let $\widetilde{h}_{1,i} = (\widehat{h}_{1,i})^{r_{\ell+1}}$
end for
Let $\widetilde{\mathbf{h}}_1 \in \mathbb{G}_1^N$ be such that the i-th component of $\widetilde{\mathbf{h}}_1$ is $\widetilde{h}_{1,i}$.
Let $\mathbf{h}'_1 = \frac{1}{N} M_{\omega_N^{-1}} \cdot \widetilde{\mathbf{h}}_1$, and $\mathsf{srs}'_{\mathcal{P}} = \{h'_{1,i} \mid i \in [0, N-1]\}$. Here $h'_{1,i}$ is the i-th component of $\mathbf{h}'_1$.
Let $\mathsf{srs}'_{\mathcal{V}} = \{h_{2,j}^{r_{\ell+1}^{2^j}} \in \mathbb{G}_2 \mid j \in [0, n]\}$.
$\pi_{\ell+1} = (g_1^{r_{\ell+1}}, u_{2,\ell}^{r_{\ell+1}})$
Output $\{\mathsf{srs}' = (\mathsf{srs}'_{\mathcal{P}}, \mathsf{srs}'_{\mathcal{V}}), \{\pi_i\}_{i\in[0,\ell+1]}\}$

Algorithm 5 KZG-FFT.verify_setup: Verify Setup for KZG-FFT

Input: $\{(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{srs}, \{\pi_i\}_{i\in[0,r]}\}$
Output: {accept/reject}

Read π_i as $(u_{1,i}, u_{2,i}) \in (\mathbb{G}_1, \mathbb{G}_2)$ for $i \in [0, \ell]$.
if $e(u_{1,0}, g_2) \neq e(g_1, u_{2,0})$ **then**
    Output reject
end if
for $i \in [1, \ell]$ **do**
    if $e(u_{1,i}, u_{2,i-1}) \neq e(g_1, u_{2,i})$ **then**
        Output reject
    end if
end for
Let $\mathbf{h}_1 \in \mathbb{G}_1^N$ be such that the i-th component of $\mathbf{h}_1$ is $h_{1,i}^{(n)}$. Let $\mathbf{h}'_1 = M_{\omega_N} \cdot \mathbf{h}_1$.
Check $h'_{1,0} = g_1$.
for $i \in [1, N-1]$ **do**
    if $e(h'_{1,i}, g_2) \neq e(h'_{1,i-1}, u_{2,\ell})$ **then**
        Output reject
    end if
end for
if $h_{2,0} \neq u_{2,\ell}$ **then**
    Output reject
end if
for $j \in [0, n-1]$ **do**
    if $e(h'_{1,2^j}, h_{2,j}) \neq e(g_1, h_{2,j+1})$ **then**
        Output reject
    end if
end for
Output accept

Lemma 9. *Let the inputs to* verify_setup *be as in Algorithm 5. Let* $\mathsf{srs}_{\mathcal{P}} = \{h_{1,i}^{(n)}\}_{i\in[0,n-1]}$, $\mathsf{srs}_{\mathcal{V}} = \{h_{2,j}\}_{j\in[0,n]}$, *and* $\pi_i = (u_{1,i}, u_{2,i})$ *for* $i \in [0,\ell]$. *Then if* verify_setup *outputs accept then there is an* $r \in \mathbb{F}$ *such that a)* $h_{1,i}^{(n)} = g_1^{\alpha_i}$, *where* $\alpha_i = N^{-1} \cdot \prod_{j\in[0,n-1]}(1 + (\omega_N^{-i} \cdot r)^{2^j})$ *for* $i \in [0, N-1]$, *b)* $h_{2,j} = g_2^{r^{2^j}}$ *for* $j \in [0,n]$, *and c)* $u_{2,\ell} = g_2^r$.

Proof. At Step 3, if verify_setup accepts there is an $r_0 \in \mathbb{F}_p$ such that $u_{1,0} = g_1^{r_0}$ and $u_{2,0} = g_2^{r_0}$. At Step 6, if verify_setup accepts there is an $r_i \in \mathbb{F}_p$ such that $u_{1,i} = g_1^{r_i}$ and $u_{2,i} = g_2^{\prod_{j=0}^{i} r_j}$, for $i \in [1,\ell]$. In particular, if verify_setup accepts at Step 6 for all $i \in [1,\ell]$. then $u_{2,\ell} = g_2^{\prod_{j=0}^{\ell} r_j}$. Let $r = \prod_{j=0}^{\ell} r_i$, $u_2 = g_2^r$. At Step 13, if verify_setup accepts then for all $i \in [0, N-1]$, $h'_{1,i} = g_1^r$. From Fact 2, and Claim 3.1, this implies $h_{1,i}^{(n)} = g_1^{\alpha_i}$, where $\alpha_i = N^{-1} \cdot \prod_{j\in[0,n-1]}(1 + (\omega_N^{-i} \cdot r)^{2^j})$ for $i \in [0, N-1]$. Similarly, if verify_setup accepts at Step 21, then $h_{2,j} = g_2^{r^{2^j}}$ for $j \in [0,n]$. □

B Multilinear PCS KZG-FOURIER

B.1 Proofs of Claims and Lemma from Section 4.1

Claim B.1. $U_i^{(n)} = \prod_{j\in[0,n-1]}(1 + (\omega_N^{-i} \cdot Y)^{2^j})\frac{1}{N}$

Proof. The i-th component of $\frac{1}{N} \cdot M_{\omega_N^{-1}} \cdot Y$ is equal to

$$\sum_{\ell=0}^{N-1} \frac{1}{N} \cdot (\omega_N^{-i} \cdot Y)^{\ell} = \prod_{j\in[0,n-1]} (1 + (\omega_N^{-i} \cdot Y)^{2^j})\frac{1}{N}$$

□

Claim B.2. $\mathcal{U}_n(1) = 1$

Proof. Consider the summation $\sum_{i\in[0,N-1]} L_i(X_0, \ldots, X_{n-1}) = 1$. Consequently, we have

$$\mathcal{U}_n(1) = \mathcal{U}_n(\sum_{i\in[0,N-1]} L_i(X_0, \ldots, X_{n-1}))$$

By leveraging linearity and the definition of $\mathcal{U}_n$, we establish $\mathcal{U}_n(1) = \sum_{i\in[0,N-1]} U_i$. Moving forward with the definition of $\mathbf{U}^{(n)}$, it ensues that:

$$\sum_{i\in[0,N-1]} U_i^{(n)} = \mathbf{1}^T \cdot (\frac{1}{N} M_{\omega_N^{-1}} \cdot \mathbf{Y})$$

Here, $\mathbf{1}$ denotes the vector of all ones. The claim is substantiated by observing:

$$\mathbf{1}^T \cdot (\frac{1}{N} M_{\omega_N^{-1}}) = (1, 0, \ldots, 0)$$

□

Claim B.3. $\mathcal{U}_n(L_i^{(d)}) = \mathcal{U}_n(L_i^{(d+1)}) + \mathcal{U}_n(L_{i+D}^{(d+1)})$ *for* $d \in [1, n-1]$.[4]

Proof. The proof relies on the linearity of $\mathcal{U}_n$, and the observation that $L_i^{(d)} = L_i^{(d+1)} + L_{i+D}^{(d+1)}$. □

[4]It is important to note that $\mathcal{U}_n(L_i^{(d)})$ differs from $\mathcal{U}_d(L_i^{(d)})$.

Lemma 10. *$\mathcal{U}_n(L_i^{(d)}) = U_i^{(d)}(Y^{2^{n-d}})$, holds for $d \in [1,n]$. Moreover, for $f \in \mathbb{F}_{\leq 1}[X_0,\ldots,X_{d-1}]$, $\{f_{L_i^{(d)}} \in \mathbb{F}\}_{i\in[0,D-1]}$ represents the Fourier coefficients of f if and only if there exists a $g \in \mathbb{F}_{<D}[Y]$ such that $\mathcal{U}_n(f) = g(Y^{2^{n-d}})$ and $g(\omega_D^i) = f_{L_i^{(d)}}$.*

Proof. For $d = n$, , the relationship follows directly from the definition of $\mathcal{U}_n$. To establish $\mathcal{U}_n(L_i^{(d)}) = U_i^{(d)}(Y^{2^{n-d}})$ we proceed by assuming $\mathcal{U}_n(L_i^{(d+1)}) = U_i^{(d+1)}(Y^{2^{n-d-1}})$, for $d \in [1, n-1]$. It follows from Claim B.3 that $\mathcal{U}_n(L_i^{(d)}) = \mathcal{U}_n(L_i^{(d+1)}) + \mathcal{U}_n(L_{i+D}^{(d+1)})$ for $i \in [0, D-1]$. Moreover, the definition of $U_i^{(d+1)}$ imply Equations 31 and 32 for $i \in [0, D-1]$

$$\mathcal{U}_n(L_i^{(d+1)}) = \frac{1}{2D}\Big(\sum_{j=0}^{2D-1} \omega_{2D}^{-i} \cdot Y^{2^{n-d-1}}\Big) \tag{31}$$

$$\begin{aligned}
\mathcal{U}_n(L_{i+D}^{(d+1)}) &= \frac{1}{2D}\Big(\sum_{j=0}^{2D-1} \omega_{2D}^{-i-D} \cdot Y^{2^{n-d-1}}\Big) && (32)\\
&= \frac{1}{2D}\Big(\sum_{j=0}^{2D-1} -\omega_{2D}^{-i} \cdot Y^{2^{n-d-1}}\Big) \\
&= \frac{1}{2D}\Big(\sum_{j=2k,k\in[0,D-1]} (\omega_{2D}^{-i} \cdot Y^{2^{n-d-1}}) && (33)\\
&+ \sum_{j=2k+1,k\in[0,D-1]} -(\omega_{2D}^{-i} \cdot Y^{2^{n-d-1}})\Big)
\end{aligned}$$

In the above equality we have used $\omega_{2D}^{-D} = -1$. The equation below can be inferred by adding Equations 31, and 33

$$\begin{aligned}
&\mathcal{U}_n(L_i^{(d+1)}) + \mathcal{U}_n(L_{i+D}^{(d+1)}) \\
&\qquad = \frac{1}{D}\sum_{k\in[0,D-1]} (\omega_{2D}^{-2i} Y^{2^{n-d}})^k \\
&\qquad = \frac{1}{D}\sum_{k\in[0,D-1]} (\omega_{D}^{-i} Y^{2^{n-d}})^k
\end{aligned}$$

This proves $\mathcal{U}_n(L_i^{(d)}) = U_i^{(d)}(Y^{2^{n-d}})$, for $d \in [1,n]$.

Suppose $f(X_0,\ldots,X_{d-1}) = \sum_{i\in[0,D-1]} f_{L_i^{(d)}} \cdot L_i^{(d)}$. By the linearity of $\mathcal{U}_n$ we can express $\mathcal{U}_n(f) = \sum_{i\in[0,D-1]} f_{L_i^{(d)}} \cdot U_i^{(d)}(Y^{2^{n-d}})$. Let $g(Y) = \sum_{i\in[0,D-1]} f_{L_i^{(d)}} \cdot U^{(d)}(Y)$. Then $\mathcal{U}_n(f) = g(Y^{2^{n-d}})$, where $g(Y) \in \mathbb{F}_{<D}[Y]$. Additionally, considering Claim B.1, and recognizing that M_{ω_D} and $\frac{1}{D} \cdot M_{\omega_D^{-1}}$ are matrix inverses of each other, we deduce that $g(\omega_D^i) = f_{L_i^{(d)}}$ for $i \in [0, D-1]$. Conversely, given any $g(Y) \in \mathbb{F}_{<D}[Y]$, we can define an $f \in \mathbb{F}[X_0,\ldots,X_{d-1}]$ such that its D Fourier coefficients are specified as follows: $f_{L_i^{(d)}} = g(\omega_D^i)$. Since g is uniquely determined by its evaluations $g(\omega_D^i)_{i\in[0,D-1]}$, we conclude using similar arguments as above that $\mathcal{U}_n(f) = g(Y^{2^{n-d}})$. □

Lemma 11. *For $d \in [1, n-1]$, $f \in \mathbb{F}_{\leq 1}[X_0,\ldots,X_{d-1}]$ if and only if there exists $\psi_f \in \mathbb{F}_{<D}[Y]$ such that*

$$\mathcal{U}_n(X_d \cdot f) = \psi_f(Y^{2^{n-d-1}}) \cdot \prod_{j\in[0,D-1]} (Y^{2^{n-d-1}} - \omega_{2D}^j)$$

$$\mathcal{U}_n((1-X_d)\cdot f) = \psi_f(-Y^{2^{n-d-1}}) \cdot \prod_{j\in[0,D-1]} (Y^{2^{n-d-1}} + \omega_{2D}^j)$$

Proof. Let $f_{L_i^{(d)}}$, $i \in [0, D-1]$ represent the Fourier coefficients of f. Then the Fourier coefficient of $X_d \cdot f$ corresponding to the basis $L_i^{(d+1)}$ is 0 for $i \in [0, D-1]$, and $f_{L_i^{(d)}}$ for $i \in [D, 2D-1]$. As established in Lemma 10, there exists $\psi_1(Y) \in \mathbb{F}_{<2D}[Y]$ such that $\mathcal{U}_n(X_d \cdot f) = \psi_1(Y^{2^{n-d-1}})$. Moreover, $\psi_1(\omega_{2D}^i) = 0$ for $i \in [0, D-1]$, and $\psi_1(\omega_{2D}^i) = f_{L_i^{(d)}}$ for $i \in [D, 2D-1]$. This implies the existence of $\psi_f \in \mathbb{F}_{<D}[Y]$ such that

$$\psi_1(Y) = \psi_f(Y) \cdot \prod_{j\in[0,D-1]} (Y^{2^{n-d-1}} - \omega_{2D}^j)$$

Similarly, the Fourier coefficient of $(1-X_d)\cdot f$ corresponding to the basis $L_i^{(d+1)}$ is $f_{L_i^{(d)}}$ for $i \in [0, D-1]$, and 0 for $i \in [D, 2D-1]$. Hence, again as established in Lemma 10, there exists $\psi_2(Y) \in \mathbb{F}_{<2D}[Y]$ such that $\mathcal{U}_n((1-X_d)\cdot f) = \psi_2(Y^{2^{n-d-1}})$. Further, $\psi_2(\omega_{2D}^i) = f_{L_i^{(d)}}$ for $i \in [0, D-1]$, and $\psi_2(\omega_{2D}^i) = 0$ for $i \in [D, 2D-1]$. This implies the existence of $\psi'_f \in \mathbb{F}_{<D}[Y]$ such that

$$\psi_2(Y) = \psi'_f(Y) \cdot \prod_{j\in[0,D-1]} (Y^{2^{n-d-1}} - \omega_{2D}^j)$$

Finally, we prove that $\psi'_f(Y) = \psi_f(-Y)$, which will complete the proof. Observe that $\psi_1(\omega_{2D}^{i+2^k}) = \psi_1(-\omega_{2D}^i) = \psi_2(\omega_{2D}^i)$, for $i \in [D, 2D-1]$. Since $d \geq 1$, we have $\psi'_f(\omega_{2D}^i) = \psi_f(-\omega_{2D}^i)$ for $i \in [D, 2D-1]$. As ψ'_f and ψ_f are degree at most D univariate polynomials, it follows that $\psi'_f(Y) = \psi_f(-Y)$. The above argument also shows that corresponding to any $\psi \in \mathbb{F}_{<D}[Y]$, if $f \in \mathbb{F}_{\leq 1}[X_0, \ldots, X_{d-1}]$ is defined such that $f_{L_i^{(d)}} = \frac{\psi(-\omega_{2D}^i)}{\prod_{j\in[0,D-1]}(-\omega_{2D}^i - \omega_{2D}^j)}$ for $i \in [0, D-1]$ then

$$\mathcal{U}_n(X_d \cdot f) = \psi(Y^{2^{n-d-1}}) \cdot \prod_{j\in[0,2^d-1]} (Y^{2^{n-d-1}} - \omega_{2D}^j)$$

$$\mathcal{U}_n((1-X_d)\cdot f) = \psi(-Y^{2^{n-d-1}}) \cdot \prod_{j\in[0,D-1]} (Y^{2^{n-d-1}} + \omega_{2D}^j)$$

□

Claim B.4. *Let $\psi \in \mathbb{F}_{<D}[Y]$. Then there exists $\psi_o, \psi_e \in \mathbb{F}_{<2^{d-1}}[Y]$ such that $\psi(Y) = \psi_e(Y^2) + Y \cdot \psi_o(Y^2)$, and $\psi(-Y) = \psi_e(Y^2) - Y \cdot \psi_e(Y^2)$. Further,*

$$\psi_e(\omega_{2^{d-1}}^i) = \frac{\psi(\omega_D^i) + \psi(-\omega_D^i)}{2} \qquad \forall i \in [0, 2^{d-1}-1]$$

$$\psi_o(\omega_{2^{k-1}}^i) = \frac{\psi(\omega_D^i) + \psi(-\omega_D^i)}{2\omega_{2^k}} \qquad \forall i \in [0, 2^{d-1}-1]$$

Proof. It is easy to see that there exists $\psi_o, \psi_e \in \mathbb{F}_{<2^{d-1}}[Y]$ such that $\psi(Y) = \psi_e(Y^2) + Y \cdot \psi_o(Y^2)$, and $\psi(-Y) = \psi_e(Y^2) - Y \cdot \psi_e(Y^2)$. Hence, for $i \in [0, 2^{d-1}-1]$, we have

$$\psi(\omega_D^i) = \psi_e(\omega_{2^{d-1}}) + \omega_D^i \cdot \psi_o(\omega_{2^{d-1}})$$

$$\psi(-\omega_D^i) = \psi_e(\omega_{2^{d-1}}) - \omega_D^i \cdot \psi_o(\omega_{2^{d-1}})$$

In the above equations, we have used that $\omega_D^{2i} = \omega_{2^{d-1}}^i$. Adding the above equations and dividing by 2, we have the expression for ψ_e. Similarly, subtracting the second equation from the first equation and dividing the expression by 2, we have the expression for ψ_o. □

We make an additional claim here that notes down the computation required to compute the commits at Step 3a of Protocol 2.

Claim B.5. *Let $f \in \mathbb{F}_{\leq 1}[X_0, \ldots, X_{d-1}]$, $\psi_f(Y)$ corresponds to f as in Lemma 11, and $\psi_{f,e}(Y)$ and $\psi_{f,o}(Y)$ correspond to $\psi_f(Y)$ as in Claim B.4. There is an algorithm that takes as input the D Fourier coefficients of f, and the evaluations of $\prod_{i\in[0,D-1]}(Y - \omega_{2D}^i)$ over the offset FFT domain of size D, and outputs the evaluations of $\psi_{f,e}(Y)$ and $\psi_{f,o}(Y)$ over the FFT domain of size D. The algorithm performs $O(D)$ $\mathbb{F}$-divisions and additions and 6-FFT's of the following domain sizes: 1 of $2D$, 1 of D, 2 of $D/2$, and 2 of N.*

Proof. Recall, from the proof of Lemma 11, $\psi_1 \in \mathbb{F}_{<D}[Y]$ such that $\psi_1(\omega_{2D}^i) = 0$ for $i \in [0, D-1]$, and $\psi_1(\omega_{2D}^i) = f_{L_i^{(d)}}$ for $i \in [D, 2D-1]$. Hence, the D Fourier coefficients of f give the evaluations ψ_1 over $\mathbf{H}_{2D}$. First, the algorithm computes the evaluations of ψ_1 over the offset of FFT domain of size D. This can be done using two FFTs: one over size $2D$, and the other over size D. Next, the algorithm uses the evaluations of $\prod_{i\in[0,D-1]}(Y - \omega_{2D}^i)$ to compute the evaluations of ψ_f over the offset of FFT domain of size D. This step requires D field divisions. Here, from Fact 3, it follows that all evaluations of $\prod_{i\in[0,D-1]}(Y - \omega_{2D}^i)$ over the offset of FFT domain of size K are non-zero. The evaluations of ψ_f are used to compute evaluations of $\psi_{f,e}$ and $\psi_{f,0}$ over the offset of FFT domain of size $D/2$. This can be done using ideas similar to Claim B.4. This step requires $O(D)$ field additions and divisions. Finally, the evaluations of $\psi_{f,e}$ and $\psi_{f,0}$ are used to compute their evaluations over $\mathbf{H}_N$. This step requires two FFTs over domain of size $D/2$, and two FFTs over domain of size N. □

B.2 Proof of Theorem 2

Completeness stems from the discussion outlined in Section 4.3. Moreover, commitment binding is guaranteed by the proof of commitment binding presented in Theorem 1 (refer to Appendix A.2). Next, we proceed to establish knowledge soundness. Let $\tilde{P}_{\mathsf{eval}}$ denote the algebraic adversary with access to an oracle, as described in Definition 3.3 of [MBKM19]. Given inputs $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathsf{Gen}(1^\lambda)$ and an initial structured reference string $\perp$ (corresponding to degree N polynomials), $\tilde{P}_{\mathsf{eval}}$ outputs $(\mathsf{srs}, C_f, D, \mathbf{x}^{(d)}, y)$ and a vector $\mathbf{a} \in \mathbb{F}^N$ satisfying the following conditions: b) In $\langle P_{\mathsf{eval}}, V_{\mathsf{eval}}\rangle(\mathsf{srs}, C_f, D, (\mathbf{x}^{(d)}), y; \mathbf{a})$, where V_{eval} accepts with probability ϵ, which is non-negligible in λ. Here, $\perp$ denotes the trivial string corresponding to $r = 0$, and $\tilde{P}_{\mathsf{eval}}$ employs the oracle to update it to srs. V_{eval} rejects if $\mathsf{srs} = \perp$. The oracle ensures that srs is well-formed, meaning there exists an $r \in \mathbb{F}$ such that $\mathsf{srs}_{\mathcal{P}} = \{g_1^{\alpha_i} \mid \alpha_i = N^{-1} \cdot \prod_{j\in[0,n-1]}(1 + (\omega_N^{-i} \cdot r)^{2^j}) \text{ for } i \in [0, N-1]\}$, and $\mathsf{srs}_{\mathcal{V}} = \{g_2^{r^{2^j}} \mid j \in [0, n]\}$.

The extractor $\mathcal{E}$ rewinds and executes $\tilde{P}_{\mathsf{eval}}$, generating an acceptance transcript for KZG-FOURIER.eval with probability ϵ. Below, we provide a brief description of such a transcript. Recall, we denote $\mathcal{U}_d(f)$ as $w_f(Y)$, and

$$C_f = C_{w_f} = \prod_{i\in[0,N-1]} (h_{1,i}^{(n)})^{a_i}$$

is the claimed commitment to f. We omit f from the subscript of w_f in the proof below.

1. At Steps 3a and 3b, let $C_{\psi_{q_{k,e}}}$, $C_{\psi_{q_{k,o}}} \in \mathbb{G}_1$ be the commitments claimed by $\tilde{P}_{\mathsf{eval}}$ corresponding to $\psi_{q_{k,e}}(Y)$, $\psi_{q_{k,o}}(Y)$ for $k \in [1, d-1]$. $\tilde{P}_{\mathsf{eval}}$ also returns $q_0 \in \mathbb{F}$, and $\mathbf{b}^{(k,e)}$, $\mathbf{b}^{(k,o)}$, $\in \mathbb{F}^N$, such that

$$C_{\psi_{q_{k,e}}} = \prod_{i\in[0,N-1]} (h_{1,i}^{(n)})^{b_i^{(k,e)}},$$

$$C_{\psi_{q_{k,o}}} = \prod_{i\in[0,N-1]} (h_{1,i}^{(n)})^{b_i^{(k,o)}},$$

2. $\mathcal{E}$ samples $z \in_r \mathbb{F}$, corresponding to which $\tilde{P}_{\mathsf{eval}}$ at Step (3d) returns claimed evaluations of $w(Y)$ at z, $\psi_{q_{k,e}}(Y)$ and $\psi_{q_{k,o}}(Y)$ at $z^{2^{d-k}}$, and $\phi_k(Y)$ at $z^{2^{d-k-1}}$ for $k \in [1, d-1]$. Let v be the claimed value of $w(Y)$ at z, $v_{k,e}$ and $v_{k,o}$ be the claimed values of $\psi_{q_{k,o}}(Y)$, $\phi_k(Y)$ at $z^{2^{d-k}}$ respectively, and u_k be the claimed value of $\psi_{q_{k,e}}(Y)$ at z^{d-k-1} for $k \in [1, d-1]$. Additionally, it satisfies the check made by V_{eval} at Step (3e) using Equation 34 below. In the following equations, the expressions for $m_{k,1}$, $m_{k,2}$, m_k, m'_k for $k \in [1, d-1]$ follow from Equations 8 and 9.

$$m_{k,1} = v_{k,e} + z^{2^{d-k-1}} \cdot v_{k,o}$$

$$m_{k,2} = v_{k,e} - z^{2^{d-k-1}} \cdot v_{k,o}$$

$$m_k = m_{k,1} \cdot u_k$$

$$m'_k = m_{k,1} \cdot u_k + m_{k,2} \cdot \frac{z^{2D \cdot 2^{d-k-1}} - 1}{u_k}$$

$$v - y = \Big(\sum_{k=1}^{d-1} (m_k - x_k \cdot m'_k) \Big) + (z^{2^{d-1}} - x_0) \cdot q_0) \tag{34}$$

3. Corresponding to γ_w, $\gamma_{k,e}, \gamma'_{k,e}, \gamma_{k,o}, \gamma'_{k,o}, \delta_k$ sampled uniformly, and independently at random by the extractor, $\tilde{P}_{\mathsf{eval}}$ returns the claimed commitment $C_\psi \in \mathbb{G}_1$ to $\psi(Y)$, and $\mathbf{b}_\psi \in \mathbb{F}^N$ such that

$$C_\psi = \prod_{i \in [0, N-1]} (h_{1,i}^{(n)})^{b_{\psi,i}}$$

4. Corresponding to s sampled uniformly, and independently at random from $\mathbb{F}$ by the extractor, $\tilde{P}_{\mathsf{eval}}$ returns claimed evaluations of $w(Y)$, $\psi_{q_{k,e}}(Y)$, $\psi_{q_{k,o}}(Y)$, $\phi_k(Y)$ for $k \in [1, d-1]$ at s. Let v' be the claimed value of $w(Y)$ at s, and $v'_{k,e}$, $v'_{k,o}$, u'_k be the claimed values of $\psi_{q_{k,e}}(Y)$, $\psi_{q_{k,o}}(Y)$, $\phi_k(Y)$ at s for $k \in [1, d-1]$. $\mathcal{E}$ samples β_w, $\beta_{k,e}, \beta_{k,o}, \kappa_k$ for $k \in [1, d-1]$ uniformly, and independently at random from $\mathbb{F}$. Using these values the claimed value of $\eta(Y)$ at s can be computed using Equations 11 and 12. Denote this computed claimed value as t. Finally, $\tilde{P}_{\mathsf{eval}}$ returns C_μ the claimed commitment to $\mu(Y)$, and $\mathbf{b}_\mu$ such that

$$C_\mu = \prod_{i \in [0, N-1]} (h_{1,i}^{(n)})^{b_{\mu,i}},$$

and the following check made by V_{eval} is satisfied at the last step

$$e(C_\eta \cdot g_1^{-t}, g_2) = e(C_\mu, h_2^{(d)} \cdot g_2^{-t}) \tag{35}$$

where C_η satisfies Equation 13.

From the proof of Theorem 1, it follows that if Equation 35 is satisfied then the $\mathcal{E}$ can compute $\eta \in \mathbb{F}_{<D}[Y]$ such that $\eta(s) = t$, and $C_\eta = g_1^{\eta(r^{2^{n-d}})}$. Moreover, from Claim A.3, there exists $\psi' \in \mathbb{F}_{<2^{\ell_\psi}}[Y]$, $w' \in \mathbb{F}_{2^{\ell_w}}[Y]$, $\psi'_{q_{k,e}} \in \mathbb{F}_{<2^{\ell_{k,e}}}[Y]$, $\psi'_{q_{k,o}} \in \mathbb{F}_{<2^{\ell_{k,o}}}[Y]$, for $k \in [1, d-1]$ such that

$$C_\psi = g_1^{\psi'(r^{2^{n-\ell_\psi}})}, \qquad C_w = g_1^{w'(r^{2^{n-\ell_w}})},$$

$$C_{\psi_{q_{k,e}}} = g_1^{\psi'_{q_{k,e}}(r^{2^{n-\ell_{k,e}}})}, \qquad C_{\psi_{q_{k,o}}} = g_1^{\psi'_{q_{k,o}}(r^{2^{n-\ell_{k,o}}})}$$

Note that the $\mathcal{E}$ can efficiently compute the evaluations of such polynomials over appropriate FFT domains using $\mathbf{a}$, $\mathbf{b}_\psi$, $\mathbf{b}_{k,e}$, $\mathbf{b}_{k,o}$ for $k \in [1, d-1]$ respectively. Also from construction of the setup in Section 4.2, we have

$$C_{\phi_k} = g_1^{\phi_k(r^{2^{n-k-1}})}$$

The construction of C_η, Claim A.2, and the N-DLOG assumption implies

$$\eta(Y^{2^{n-d}}) = \psi'(Y^{2^{n-\ell_\psi}}) + \beta_w \cdot w'(Y^{2^{n-\ell_w}}) + \sum_{k\in[1,d-1]} \Bigg(\beta_{k,e} \cdot \psi_{q_k,e}(Y^{2^{n-\ell_{k,e}}}) + \beta_{k,o} \cdot \psi_{q_k,o}(Y^{2^{n-\ell_{k,o}}}) \kappa_k \cdot \phi_k(Y^{2^{n-k-1}})\Bigg) \tag{36}$$

In the following claim, we bound the degrees of ψ', w', $\psi'_{k,e}$, $\psi'_{k,o}$ for $k \in [1, d-1]$.

Claim B.6. *Let η, ψ', w', $\psi'_{k,e}$, $\psi'_{k,o}$ for $k \in [1, d-1]$ be as defined above. Then the degree of all these polynomials is at most D with probability at least $1 - \frac{1}{|F|}$ over the random choices of β_w, and $\beta_{k,e}$, $\beta_{k,o}$, and κ_k, for $k \in [1, d-1]$.*

Proof. Since the degree of $\eta(Y)$ is at most $D = 2^d$, the LHS of Equation 36 has monomials of the form $Y^{i\cdot 2^{n-d}}$ for $i \in [0, D-1]$. Hence, by Schwartz-Zippel with probability at least $\epsilon(1 - \frac{1}{|F|})$, the monomials in each of the polynomials $\psi'(Y^{2^{n-\ell_\psi}})$, $w'(Y^{2^{n-\ell_w}})$, $\psi_{q_k,e}(Y^{2^{n-\ell_{k,e}}})$, $\psi_{q_k,o}(Y^{2^{n-\ell_{k,o}}})$, $\phi_k(Y^{2^{n-k-1}})$ for $k \in [1, d-1]$ is of the form $Y^{i\cdot 2^{n-d}}$ for $i \in [0, D-1]$. Now suppose $2^{\ell_\psi} \geq D$. Then this implies there exists $j \neq j'$ such that $j, j' \in [0, 2^{\ell_\psi} - 1]$ such that $2^{n-d} \cdot i = 2^{n-\ell_\psi} \cdot j$, and $2^{n-d} \cdot i = 2^{n-\ell_\psi} \cdot j'$. But this implies $j = j'$, a contradiction. Hence, $2^{\ell_\psi} < D$. A similar argument can be used to bound the degrees of all the remaining polynomials. □

From Claim B.6, we conclude that $\psi'(Y)$, $w'(Y)$, $\psi'_{q_k,e}$, $\psi'_{q_k,o}$ for $k \in [1, d-1]$ are at most degree D polynomials. Hence, there exists $\psi(Y)$, $w'(Y) \in \mathbb{F}_{<D}[Y]$, $\psi'_{q_k,e}$, $\psi'_{q_k,o}\mathbb{F}_{<D}[Y]$ for $k \in [1, d-1]$ such that $\psi(Y^{2^{n-d}}) = \psi'(Y^{2^{n-\ell_\psi}})$, $w(Y^{2^{n-d}}) = w'(Y^{2^{n-\ell_w}})$, $\psi_{q_k,e}(Y^{2^{n-d}}) = \psi_{q_k,e}(Y^{2^{n-\ell_{q_k,e}}})$, $\psi_{q_k,o}(Y^{2^{n-d}}) = \psi_{q_k,o}(Y^{2^{n-\ell_{q_k,o}}})$. For example, if $\ell_w = d$ then let $\psi(Y) = \psi'(Y)$, and if $\ell_w = d-1$ then let $\psi(Y) = \psi(Y^2)$ and so on. Hence, we may rewrite Equation 36 as follows

$$\eta(Y) = \psi(Y) + \beta_w \cdot w(Y) + \sum_{k\in[1,d-1]} \Bigg(\beta_{k,e} \cdot \psi_{q_k,e}(Y) + \beta_{k,o} \cdot \psi_{q_k,o}(Y) \kappa_k \cdot \phi_k(Y^{2^{d-k-1}})\Bigg) \tag{37}$$

Let E_1 be the event that the above equation holds. Conditioned on that E_1 holds, and since $\eta(s) = t$ and t is computed using Equations 11 and 12 from the claimed values of $w(Y)$, $\psi_{q_k,e}$, $\psi_{q_k,o}$, and ϕ_k at s, we have by Schwartz-Zippel with probability $1 - \frac{D+1}{|F|}$ over the random choices of s, β_w, $\beta_{k,e}$, $\beta_{k,o}$, and κ_k for $k \in [1, d-1]$,

$$w(s) = v', \quad \psi_{q_k,e}(s) = v'_{k,e}, \qquad \text{and}$$
$$\psi_{q_k,o}(s) = v'_{k,o}, \quad \psi_k(s^{2^{d-k-1}}) = u'_k$$

and, $\psi(Y)$ satisfies Equation 11.

Let E_2 be the event that Equation 11 holds. Conditioned on E_2, with probability $1 - \frac{1}{\mathbb{F}}$, over the random choices of γ, $\gamma_{k,e}$, $\gamma_{k,o}$, and δ_k for $k \in [1, d-1]$.

$$w(z) = v, \quad \psi_{q_k,e}(z^{2^{d-k}}) = v_{k,e}, \quad \psi_{q_k,o}(z^{2^{d-k}}) = v_{k,o}$$
$$\phi_k^{z^{2^{d-k-1}}} = u_k, \quad \zeta_{q_k,e}(z^{2^{d-k}}) = z^{2^{2d-k}-2^d} \cdot v_{k,e},$$
$$\zeta_{q_k,o}(z^{2^{d-k}}) = z^{2^{2d-k}-2^d} \cdot v_{k,o}$$

Let E_3 be the event that the above relations hold. Conditioned on E_3, and since the check using Equation 34 passes, with probability $1 - \frac{2D}{|\mathbb{F}|}$ the following hold:

1. $\zeta_{q_k,e}(Y) = Y^{2^d-2^{k-1}} \cdot \psi_{k,e}(Y)$, and $\zeta_{q_k,o}(Y) = Y^{2^d-2^{k-1}} \cdot \psi_{k,o}(Y)$.
2. Let Let $\psi_{q_k}(Y) = \psi_{q_k,e}(Y^2) + Y \cdot \psi_{q_k,o}(Y^2)$. Then

$$\begin{aligned} w(Y) - v = \sum_{k\in[1,d-1]} \Big(& \psi_{q_k}(Y^{2^{d-k-1}}) \cdot \phi_k(Y^{2^{d-k-1}}) \\ & - x_k \cdot \big((\psi_{q_k}(Y^{2^{d-k-1}}) \cdot \phi_k(Y^{2^{d-k-1}}) \\ & + \psi_{q_k}(-Y^{2^{d-k-1}}) \cdot \frac{Y^{2^{d+1}\cdot 2^{d-k-1}}}{\phi_k(Y^{2^{d-k-1}})} \big) \Big) \\ & + (Y^{2^{d-1}} - x_0) q_0 \end{aligned} \tag{38}$$

Since degree of $\psi(Y)$ is at most D, from Equation 11 it follows that degrees of $\zeta_{q_k,e}(Y)$ and $\zeta_{q_k,o}(Y)$ are at most D. This implies degree of $\psi_{k,e}$, and $\psi_{k,o}$ is at most $K/2$, and hence degree of $\psi_{q_k}(Y)$ is at most K. From Lemma 10 there exists $f \in \mathbb{F}_{\leq 1}[\mathbf{X}^{(d)}]$, and $q_k \in \mathbb{F}_{\leq 1}[\mathbf{X}^{(k)}]$ for $k \in [1, d-1]$ such that $\mathcal{U}_d(f) = w(Y)$, and

$$\mathcal{U}_d(X_k \cdot q_k) = \psi_{q_k}(Y^{2^{d-k-1}}) \cdot \phi_k(Y^{2^{d-k-1}})$$

$$\mathcal{U}_d(q_k) = \psi_{q_k}(Y^{2^{d-k-1}}) \cdot \phi_k(Y^{2^{d-k-1}}) + \psi_{q_k}(-Y^{2^{d-k-1}}) \cdot \frac{Y^{2^{d+1}\cdot 2^{d-k-1}}}{\phi_k(Y^{2^{d-k-1}})}$$

Further, it is easy to construct f from w using the evaluations of w over $\mathbf{H}_D$. Hence, we have from Equation 38,

$$\mathcal{U}_d(f) - v = \sum_{k\in[0,d-1]} \mathcal{U}_d(X_k \cdot q_k) - \mathcal{U}_d(x_k \cdot q_k)$$

Since $\mathcal{U}_d$ is a linear isomorphism, we have

$$f(X_0, \ldots, X_{d-1}) - v = \sum_{k\in[0,d-1]} (X_k - x_k) \cdot q_k(X_0, \ldots, X_{k-1})$$

This implies if $\mathcal{E}$ can construct an compute an accepting transcript using $\tilde{P}_{\mathsf{eval}}$ with probability ϵ, then it can compute the D Fourier coefficients of an $f \in \mathbb{F}_{\leq 1}[X_0, \ldots, X_{d-1}]$ such that $f(x_0, \ldots, x_{d-1}) = v$ with probability at least $\epsilon(1 - \frac{D}{|\mathbb{F}|})^4$, which is non-negligible in λ assuming $|\mathbb{F}| \gg \lambda$.

C Proof of Theorem 3

Completeness: First we argue that if $M_{\omega_D} \cdot \mathbf{f} = \mathbf{a}$ then

$$\prod_{i\in[0,D-1]} e(\tau_i^{(1,d)}, g_2^{a_i}) = \prod_{j\in[0,D-1]} e(m_{D,j}, g_2^{f_j}),$$

where $m_{D,j}$ is as computed in Step 4 of Algorithm 3. Since $M_{\omega_D} \cdot \mathbf{f} = \mathbf{a}$, for all $i \in [0, D-1]$

$$a_i = \sum_{j\in[0,D-1]} \omega_D^{i\cdot j} \cdot f_j \tag{39}$$

Now we have the following sequence of equations.

$$\begin{aligned}
\prod_{j\in[0,D-1]} e(m_{D,j}, g_2^{f_j}) &= \prod_{j\in[0,D-1]} e(m_{D,j}^{f_j}, g_2) && (40)\\
&= \prod_{j\in[0,D-1]} e(\prod_{i\in[0,D-1]} ((\tau_i^{(1,d)})^{\omega_D^{i\cdot j}})^{f_j}, g_2)\\
&= \prod_{j\in[0,D-1]} e(\prod_{i\in[0,D-1]} (\tau_i^{(1,d)})^{f_j\cdot\omega_D^{i\cdot j}}, g_2)\\
&= \prod_{i\in[0,D-1]}\prod_{j\in[0,D-1]} e((\tau_i^{(1,d)})^{f_j\cdot\omega_D^{i\cdot j}}, g_2)\\
&= \prod_{i\in[0,D-1]} e(\prod_{j\in[0,D-1]} (\tau_i^{(1,d)})^{f_j\cdot\omega_D^{i\cdot j}}, g_2)\\
&= \prod_{i\in[0,D-1]} e((\tau_i^{(1,d)})^{\sum_{j\in[0,D-1]} f_j\cdot\omega_D^{i\cdot j}}, g_2)\\
&= \prod_{i\in[0,D-1]} e((\tau_i^{(1,d)})^{a_i}, g_2) && (41)\\
&= \prod_{i\in[0,D-1]} e((\tau_i^{(1,d)}), g_2^{a_i}) && (42)
\end{aligned}$$

Equations 40 and 42 above follows from bi-linearity of e, and Equation 41 follows from Equation 39. Hence, it follows from the completeness of dory that in this case V_{dory} always accepts at Steps 2-4, and hence V_{lin} accepts at Step 5.

Knowledge Soundness: Suppose $\tilde{P}_{\mathsf{lin}}$ outputs C_f, $C_{\mathbf{a}}$ such that V_{lin} accepts with non-negligible probability in λ. This implies V_{dory} accepts at Steps 2-4. By knowledge soundness of Dory, there exists an extractor that can rewind $\tilde{P}_{\mathsf{lin}}$ and compute $\mathbf{f} \in \mathbb{F}^D$, and $\mathbf{a} \in \mathbb{F}^D$ such that the following hold:

$$\begin{aligned}
C_{\mathbf{f}} &= \prod_{i=0}^{D-1} e(\tau_i^{(1,d)}, g_2^{f_i})\\
C_{\mathbf{a}} &= \prod_{i=0}^{n-1} e(\tau_i^{(1,d)}), g_2^{a_i})\\
\prod_{j\in[0,D-1]} e(m_{D,j}, g_2^{f_j}) &= \prod_{i\in[0,D-1]} e(\tau_i^{(1,d)}), g_2^{a_i}) && (43)
\end{aligned}$$

Hence, it suffices to argue that $M_{\omega_D}\cdot\mathbf{f} = \mathbf{a}$ in this case. Suppose $M_{\omega_D}\cdot\mathbf{f} \neq \mathbf{a}$, and $M_{\omega_D}\cdot\mathbf{f} = \mathbf{b}$. In the above sequence of Equations (from Equation 40 to Equation 42) replacing $\mathbf{a}$ with $\mathbf{b}$, it follows that

$$\prod_{j\in[0,D-1]} e(m_{D,j}, g_2^{f_j}) = \prod_{i\in[0,D-1]} e(\tau_i^{(1,d)}), g_2^{b_i})$$

Hence, from Equation 43, we have

$$\prod_{i\in[0,D-1]} e(\tau_i^{(1,d)}), g_2^{a_i}) = \prod_{i\in[0,D-1]} e(\tau_i^{(1,d)}), g_2^{b_i})$$

Since $\mathbf{a} \neq \mathbf{b}$, this implies $\mathcal{E}$ can efficiently compute two distinct vectors with the same AFG commitment under commitment key $\tau^{(1,d)}$. This contradicts SXDH with respect to bilinear group generator $\mathcal{G}$.

D Succinct Argument of Knowledge for Inner-Pairing Products

In this section we revisit the Dory argument system from [Lee21] that allows the prover to give a succinct proof corresponding to an inner pairing product. In particular, the prover proves there exists a vector $\mathbf{h} \in \mathbb{G}_1^D$, a vector $\mathbf{s} \in \mathbb{G}_2^D$ such that $C = \prod_{i\in[0,D-1]} e(h_i, s_i)$. The Dory proof system enables the prover to iteratively prove the knowledge of $\mathbf{h}$ and $\mathbf{s}$ such that $C = \prod_{i\in[n]} e(h_i, s_i)$, $C_{\mathbf{h}} = \prod_{i\in[0,D-1]} e(h_i, \tau_{2,i})$, and $C_{\mathbf{s}} = \prod_{i\in[0,D-1]} e(\tau_{1,i}, s_i)$, where τ_1, and τ_2 are publicly known elements. Additionally, as a special case the Dory proof system can also be used to succinctly prove the knowledge of opening for an AFG commitment.

In Section D.1, we begin by specifying the required public parameters for the Dory proof system, and in Section D.2, we state the Dory argument system. Finally, in Section D.3, we give state how to use Dory argument to prove evaluations for an univariate or a multilinear polynomial.

D.1 Public Parameters

Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, be a bilinear group. Let $n = \log N$, and $\tau^{(1,j)} \in \mathbb{G}_1^{N/2^{n-j}}$ and $\tau^{(2,j)} \in \mathbb{G}_2^{N/2^{n-j}}$ for $j \in [0, n]$ be publicly known vectors chosen independently and uniformly at random from $\mathbb{G}_1$, and $\mathbb{G}_2$. Let $C_{\tau_j} = \langle \tau^{(1,j)}, \tau^{(2,j)} \rangle$, for $j \in [0, n]$, and $\tau_L^{(1,j)} \in \mathbb{G}_1^{N/2^{n-(j+1)}}$, $\tau_R^{(1,j)} \in \mathbb{G}_1^{N/2^{n-(j+1)}}$ denote the left and the right halves of the vector $\tau^{(1,j)}$, for $j \in [0, n-1]$. Similarly, let $\tau_L^{(2,j)} \in \mathbb{G}_2^{N/2^{n-(j+1)}}$, $\tau_R^{(2,j)} \in \mathbb{G}_2^{N/2^{n-(j+1)}}$ denote the left and the right halves of the vector $\tau^{(2,j)}$, for $j \in [0, n-1]$. Further, let $\Delta_L^{(1,j)} = \langle \tau_L^{(1,j)}, \tau^{(2,j+1)} \rangle$, $\Delta_R^{(1,j)} = \langle \tau_R^{(1,j)}, \tau^{(2,j+1)} \rangle$, $\Delta_L^{(2,j)} = \langle \tau^{(1,j+1)}, \tau_L^{(2,j)} \rangle$, $\Delta_R^{(1,j)} = \langle \tau^{(1,j+1)}, \tau_R^{(2,j)} \rangle$. For $j \in [0, n-1]$, the quantities $\Delta_L^{(1,j)}, \Delta_R^{(1,j)}, \Delta_L^{(2,j)}, \Delta_R^{(2,j)}$ are pre-computed and are part of the public parameters.
Let $\mathrm{pp} \leftarrow_R \mathsf{dory.Setup}(1^\lambda, N)$, where λ is security parameter, and N is the vector size-bound on $\mathbf{h}$ and $\mathbf{s}$. Here the public parameters $\mathrm{pp} = (\mathrm{pp}_{\mathcal{P}}, \mathrm{pp}_{\mathcal{V}})$, $\mathrm{pp}_{\mathcal{P}} = \{\tau^{(1,j)}, \tau^{(2,j)} \mid j \in [0, n])\}$, and $\mathrm{pp}_{\mathcal{V}} = \{C_{\tau_j}, \Delta_L^{(1,j)}, \Delta_R^{(1,j)}$ $\Delta_L^{(2,j)}$, $\Delta_R^{(2,j)} \mid j \in [0, n-1])\}$.

D.2 Dory Protocol

The Dory argument system is given in Protocol 7. pp is generated as described in Section D.1. It is an argument of knowledge for the following relation

$$\begin{aligned}\{C, C_{\mathbf{h}}, C_{\mathbf{s}}, D \leq N \mid \exists \mathbf{h} \in \mathbb{G}_1^D, \mathbf{s} \in \mathbb{G}_2, \text{ such that} \\ C = \prod_{i\in[0,D-1]} e(h_i, s_i), \quad C_{\mathbf{h}} = \prod_{i\in[0,D-1]} e(h_i, \tau_i^{(2,d)}), \\ C_{\mathbf{s}} = \prod_{i\in[0,D-1]} e(\tau_i^{(1,d)}, s_i)\end{aligned}$$

The Dory protocol proceeds in d rounds (for loop at Step 2 corresponds to the d rounds), and at each round the size of the instance is reduced by half. At the end of d rounds the size of instance is one and can be checked trivially by the verifier (Steps 10-11). We describe one round of the argument system (that is Steps 2-9), where the instance size is reduced by half.

Round j: At the beginning of round j the verifier has knowledge of $C_1^{(j+1)}, C_2^{(j+1)}, C_3^{(j+1)}$ defined as follows:

$$C_1^{(j+1)} = \langle \mathbf{h}^{(j+1)}, \mathbf{s}^{(j+1)} \rangle, \; C_2^{(j+1)} = \langle \mathbf{h}^{(j+1)}, \tau^{(2,j+1)} \rangle,$$
$$C_3^{(j+1)} = \langle \tau^{(1,j+1)}, \mathbf{s}_{(2,j+1)} \rangle$$

Here, for $j \in [0, d-1]$, $\tau^{(1,j)}, \tau^{(2,j)}$ are as defined in Section D.1, and $\mathbf{h}^{(j)} \in \mathbb{G}_1^{D/2^{d-j}}, \mathbf{s}^{(j)} \in \mathbb{G}_2^{D/2^{d-j}}$. Additionally, for round $j = d-1$: $\mathbf{h}^{(d)} = \mathbf{h}$, $\mathbf{s}^{(d)} = \mathbf{s}$, and hence, $C_1^{(d)} = C, C_2^{(d)} = C_{\mathbf{h}}, C_3^{(d)} = C_{\mathbf{s}}$.

In round j, the prover and verifier exchange messages to enable the verifier to compute $C_1^{(j)}, C_2^{(j)}, C_3^{(j)}$ corresponding to vectors $\mathbf{h}^{(j)}, \mathbf{s}^{(j)}, \tau^{(1,j)}, \tau^{(2,j)}$ using the received messages such that if the prover knows $\mathbf{h}^{(j+1)}, \mathbf{s}^{(j+1)}, \tau^{(1,j+1)}, \tau^{(2,j+1)}$ then it can (efficiently) compute $\mathbf{h}^{(j)}, \mathbf{s}^{(j)}$ such that

$$C_1^{(j)} = \langle \mathbf{h}^{(j)}, \mathbf{s}^{(j)} \rangle,\ C_2^{(j)} = \langle \mathbf{h}^{(j)}, \tau^{(2,j)} \rangle,\ C_3^{(j)} = \langle \tau^{(1,j)}, \mathbf{s}_{(2,j)} \rangle$$

Ate Step 3, P_{dory} computes $D_{1,L}, D_{1,R}, D_{2,L}, D_{2,R}$ as follows:

$$\begin{aligned} D_{1,L} &= \langle \mathbf{h}_L^{(j+1)}, \tau^{(2,j)} \rangle \\ D_{1,R} &= \langle \mathbf{h}_R^{(j+1)}, \tau^{(2,j)} \rangle \\ D_{2,L} &= \langle \tau^{(1,j)}, \mathbf{s}_L^{(j+1)} \rangle \\ D_{2,R} &= \langle \tau^{(1,j)}, \mathbf{s}_R^{(j+1)} \rangle \end{aligned} \tag{44}$$

Let $\mathbf{w}_1 = \mathbf{h}^{(j+1)} + \alpha_j \cdot \tau^{(1,j+1)}$, $\mathbf{w}_2 = \mathbf{s}^{(j+1)} + \alpha_j^{-1} \cdot \tau^{(2,j+1)}$. At Step 5, P_{dory} computes s_L, and s_R as follows:

$$\begin{aligned} s_L &= \langle \mathbf{w}_{1,L}, \mathbf{w}_{2,R} \rangle \\ s_R &= \langle \mathbf{w}_{1,R}, \mathbf{w}_{2,L} \rangle \end{aligned} \tag{45}$$

At Step 7, P_{dory} computes $\mathbf{h}^{(j)}$ and $\mathbf{s}^{(j)}$ as follows:

$$\begin{aligned} \mathbf{h}^{(j)} &= \mathbf{w}_{1,L}^{\beta_j} \cdot \mathbf{w}_{1,R}^{\beta_j^{-1}} \\ \mathbf{s}^{(j)} &= \mathbf{w}_{2,L}^{\beta_j^{-1}} \cdot \mathbf{w}_{2,R}^{\beta_j} \end{aligned} \tag{46}$$

To compute $C_1^{(j)}, C_2^{(j)}, C_3^{(j)}$ at Step 8, V_{dory} first computes $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$

$$\langle \mathbf{w}_1, \mathbf{w}_2 \rangle = C_1^{(j+1)} \cdot (C_3^{(j+1)})^{\alpha_j} \cdot (C_2^{(j+1)})^{\alpha_j^{-1}} \cdot (\langle \tau^{(1,j+1)}, \tau^{(2,j+1)} \rangle)$$

Finally at Step 8, $C_1^{(j)}, C_2^{(j)}, C_3^{(j)}$ is computed as follows:

$$\begin{aligned} C_1^{(j)} &= (\langle \mathbf{w}_1, \mathbf{w}_2 \rangle) \cdot s_L^{\beta_j^2} \cdot s_R^{\beta_j^{-2}} \\ C_2^{(j)} &= D_{1,L}^{\beta_j} \cdot (\Delta_{1,L}^{(j)})^{\beta_j \alpha_j} \cdot D_{1,R}^{\beta_j^{-1}} \cdot (\Delta_{1,R}^{(j)})^{\beta_j^{-1} \alpha_j} \\ C_3^{(j)} &= D_{1,L}^{\beta_j^{-1}} \cdot (\Delta_{2,L}^{(j)})^{\beta_j^{-1} \alpha_j^{-1}} \cdot D_{1,R}^{\beta_j} \cdot (\Delta_{2,R}^{(j)})^{\beta_j \alpha_j^{-1}} \end{aligned} \tag{47}$$

It can be easily verified that $\mathbf{h}^{(j)}$, and $\mathbf{s}^{(j)}$ as defined at Step 5, and $C_1^{(j)}, C_2^{(j)}, C_3^{(j)}$ as defined at Step 6 satisfy

$$C_1^{(j)} = \langle \mathbf{h}^{(j)}, \mathbf{s}^{(j)} \rangle,\ C_2^{(j)} = \langle \mathbf{h}^{(j)}, \tau^{(2,j)} \rangle,\ C_3^{(j)} = \langle \tau^{(1,j)}, \mathbf{s}_{(2,j)} \rangle$$

Protocol 7: Dory Argument System

$\mathsf{pp} \leftarrow_R \mathsf{dory.Setup}(1^\lambda, N)$
$\text{accept/reject} \leftarrow \langle P_{\mathsf{dory}}, V_{\mathsf{dory}} \rangle (\mathsf{pp}, C, C_{\mathbf{h}}, C_{\mathbf{s}}, D; \mathbf{h}, \mathbf{s})$

1: Let $C_1^{(d)} = C$, $C_2^{(d)} = C_{\mathbf{h}}$, $C_3^{(d)} = C_{\mathbf{s}}$, $\mathbf{h}^{(d)} = \mathbf{h}$, $\mathbf{s}^{(d)} = \mathbf{s}$
2: **for** $j = d-1$ to $j = 0$; j- - **do**
3: $P_{\mathsf{dory}} \to V_{\mathsf{dory}}$: Compute $D_{1,L}, D_{1,R}, D_{2,L}, D_{2,R}$ as given in Equation 44 and send it to V_{dory}.
4: $V_{\mathsf{dory}} \to P_{\mathsf{dory}}$: Sample $\alpha_j \in \mathbb{F}$ and send to P_{dory}.

5: $P_{\mathsf{dory}} \to V_{\mathsf{dory}}$: Compute s_L, and s_R as in Equation 45 and send it to V_{dory}
6: $V_{\mathsf{dory}} \to P_{\mathsf{dory}}$: Sample $\beta_j \in \mathbb{F}_p$ and send to P_{dory}.
7: P_{dory}: Compute $\mathbf{h}^{(j)}$ and $\mathbf{s}^{(j)}$ as given in Equation 46.
8: V_{dory}: Compute $C_1^{(j)}, C_2^{(j)}, C_3^{(j)}$ as given in Equation 47.
9: **end for**
10: $P_{\mathsf{dory}} \to V_{\mathsf{dory}}$: $\mathbf{h}^{(0)} \in \mathbb{G}_1$, $\mathbf{s}^{(0)} \in \mathbb{G}_2$
11: V_{dory}: Accept if the following are true

$$C_1^{(0)} = e(\mathbf{h}^{(0)}, \mathbf{s}^{(0)}), C_2^{(0)} = e(\mathbf{h}^{(0)}, \tau^{(2,0)})$$

$$C_3^{(0)} = e(\tau^{(1,0)}, \mathbf{s}^{(0)})$$

D.3 Evaluation Proofs using Dory

For completeness, in this section, we also state how Dory can be used to evaluate univariate (or multilinear) polynomial at a given point in Protocol 8. The (Fourier) coefficients of the (resp. multilinear) univariate polynomial are committed to using the AFG commitment. In particular, Protocol 8 can be used to prove the following relations for the univariate and multilinear evaluations respectively:

$$\{C_f \in \mathbb{G}_T, D \leq N, u \in \mathbb{F}, v \in \mathbb{F} \mid \exists f \in \mathbb{F}_{<D}[Y] \text{ such that } C_f = \prod_{i=0}^{D-1} e(\tau_i^{(1,d)}, g_2^{f_i}), \text{ and } \langle f(u) = v\}$$

$$\{C_{\mathbf{a}} \in \mathbb{G}_T, d \leq n, \mathbf{x} \in \mathbb{F}^d, y \in \mathbb{F} \mid \exists \mathbf{a} \in \mathbb{F}^D \text{ such that } C_{\mathbf{a}} = \prod_{i=0}^{D-1} e(\tau_i^{(1,d)}, g_2^{a_i}), \text{ and } \langle \tilde{a}(\mathbf{x}) = y\}$$

We explain the protocol for univariate commitment scheme, and the same can be adapted for multilinear.

Protocol 8: Evaluation Proof using Dory

pp $\leftarrow_R$ dory.Setup$(1^\lambda, N)$
accept/reject $\leftarrow \langle P_{\mathsf{dory\_eval}}, V_{\mathsf{dory\_eval}} \rangle (\mathsf{pp}, C_f, D, u, v; \mathbf{f})$

1: Let $C_1^{(d)} = C_f$, $C_2^{(d)} = \langle \tau^{1,d}, \tau^{2,d} \rangle$, $C_3^{(d)} = \langle \tau^{1,d}, \tau^{2,d} \rangle$, $\mathbf{h}^{(d)} = \tau^{(1,d)}$, $\mathbf{s}^{(d)} = g_2^{\mathbf{f}}$
2: P_{dory}: Compute $\mathbf{z}^{(d)} = \otimes_{j \in [0,d-1]}(1, u^{2^j})$
3: V_{dory}: Compute $q^{(d)} = g_T^y$
4: **for** $j = d-1$ to $j = 0$; j- - **do**
5: $P_{\mathsf{dory}} \to V_{\mathsf{dory}}$: Compute $\theta_L, \theta_R, \delta_L, \delta_R$ as given in Equations 48-49, and send it to V_{dory}.
6: $P_{\mathsf{dory}} \to V_{\mathsf{dory}}$: Compute $D_{1,L}, D_{1,R}, D_{2,L}, D_{2,R}$ as given in Equation 44 and send it to V_{dory}.
7: $V_{\mathsf{dory}} \to P_{\mathsf{dory}}$: Sample $\alpha_j \in \mathbb{F}$ and send to P_{dory}.
8: $P_{\mathsf{dory}} \to V_{\mathsf{dory}}$: Compute s_L, and s_R as in Equation 45 and send it to V_{dory}
9: $V_{\mathsf{dory}} \to P_{\mathsf{dory}}$: Sample $\beta_j \in \mathbb{F}_p$ and send to P_{dory}.
10: P_{dory}: Compute $\mathbf{h}^{(j)}$ and $\mathbf{s}^{(j)}$ as given in Equation 46.
11: V_{dory}: Compute $C_1^{(j)}, C_2^{(j)}, C_3^{(j)}$ as given in Equation 47.
12: P_{dory}: Compute $\mathbf{z}^{(j)} = \beta_j \cdot \mathbf{z}_L^{(j+1)} + \beta_j^{-1} \cdot \mathbf{z}_R^{(j+1)}$
13: V_{dory}: Compute $q^{(j)}$ as in Equation 51.

14: **end for**
15: $P_{\mathsf{dory}} \to V_{\mathsf{dory}}$: $\mathbf{h}^{(0)} \in \mathbb{G}_1$, $\mathbf{s}^{(0)} \in \mathbb{G}_2$
16: V_{dory}: Compute $\mathbf{z}^{(0)} = \prod_{i\in[0,d-1]}(\beta_{d-1-j} + \beta^{-1}_{d-1-j} \cdot u^{2^j})$
17: V_{dory}: Accept if the following are true

$$C_1^{(0)} = e(\mathbf{h}^{(0)}, \mathbf{s}^{(0)}), C_2^{(0)} = e(\mathbf{h}^{(0)}, \tau^{(2,0)})$$

$$C_3^{(0)} = e(\tau^{(1,0)}, \mathbf{s}^{(0)})$$

$$e(g_1^{\mathbf{z}^{(0)}}, \mathbf{s}^{(0)}) = q^{(0)}$$

Protocol 8 is similar to Protocol 7, in the sense that it is internally running $\langle P_{\mathsf{dory}}, V_{\mathsf{dory}}\rangle$ for $C = C_{\mathbf{f}}$, $C_{\mathbf{h}} = \langle \tau^{1,d}, \tau^{2,d}\rangle$, $C_{\mathbf{s}} = C_f$, where $\mathbf{h} = \tau^{(1,d)}$, $\mathbf{s} = g_2^{\mathbf{f}}$, to open the commitment to $\mathbf{f}$. Hence, $C_1^{(0)}$, $C_2^{(0)}$, $C_3^{(0)}$ is set accordingly at Step 1. Most of the steps in Protocol 8 are similar to Protocol 7 and we refer to Section D.2 for its description. The protocol involves additional steps to check $f(u) = v$, which is equivalent to checking $\langle \mathbf{z}^{(d)}, \mathbf{f}\rangle = v$, where $\mathbf{z}^{(d)} = \otimes_{j\in[0,d-1]}(1, u^{2^j})$ and $\otimes$ denote the tensor product. We explain the additional Steps 2-3, 5, 12-13, 16, and the final check at Step 17. At Step 5, P_{dory} computes θ_L, θ_R, δ_L, δ_R as follows:

$$\theta_L = \langle g_1^{\mathbf{z}_L^{(i)}}, \mathbf{s}_R^{(0)}\rangle \tag{48}$$

$$\theta_R = \langle g_1^{\mathbf{z}_R^{(i)}}, \mathbf{s}_L^{(0)}\rangle \tag{49}$$

$$\delta_L = \langle g_1^{\mathbf{z}_L^{(i)}}, \tau_R^{(2,d-j)}\rangle \tag{50}$$

$$\delta_R = \langle g_1^{\mathbf{z}_R^{(i)}}, \tau_L^{(2,d-j)}\rangle \tag{51}$$

At Step 13, V_{dory} computes $\mathbf{q}^{(j)}$ as follows

$$\mathbf{q}^{(j)} = q^{(j+1)} \cdot \theta_L^{\beta_j^2} \cdot \theta_R^{\beta_j^{-2}} \cdot \delta_L^{\alpha_j^{-1}\beta_j^{-2}} \cdot \delta_R^{\alpha_j^{-1}\beta_j^2} \tag{52}$$

It is easy to check that for θ_L, θ_R, δ_L, δ_R, and $\mathbf{q}^{(j)}$ as defined for an honest prover we have $\langle g_1^{\mathbf{z}^{(j)}}, \mathbf{s}^{(j)}\rangle = q^{(j)}$, for $j \in [0, d-1]$. Also, the tensor structure of the evaluation point is exploited by V_{dory} to succinctly compute $\mathbf{z}^{(0)} = \prod_{i\in[0,d-1]}(\beta_{d-1-j} + \beta^{-1}_{d-1-j} \cdot u^{2^j})$ at Step 16. Hence, the verifier is succinctly able to check $e(g_1^{\mathbf{z}^{(0)}}, \mathbf{s}^{(0)}) = q^{(0)}$ at Step 17.