$\mathfrak{cq}_{lin}$: Efficient linear operations on KZG commitments with cached quotients

Liam Eagen
Blockstream Research

Ariel Gabizon
Zeta Function Technologies

October 16, 2023

Abstract

Given two KZG-committed polynomials $f(X), g(X) \in \mathbb{F}_{<n}[X]$, a matrix $M \in \mathbb{F}^{n \times n}$, and subgroup $H \subset \mathbb{F}^*$ of order n, we present a protocol for checking that $f|_{\mathbb{H}} \cdot M = g|_{\mathbb{H}}$. After preprocessing, the prover makes $O(n)$ $\mathbb{F}$-operations and $\mathbb{G}_1$-operations. This is a significant improvement for dense matrices compared to the lincheck protocols in [CHM+19, COS19], where the prover's run-time (also after preprocessing) was quasilinear in the number of non-zeroes of M, which could be n^2.

1 Introduction

The landscape of pairing-based zk-SNARKs is mostly populated by members of two families:

1. zk-SNARKs for QAP/R1CS following the approach originating in [GGPR13]; having their most popular and optimized form in [Gro16].
2. Universal zk-SNARKs based on the polynomial commitment scheme of [KZG10], introduced in [MBKM19] with later popular members being Marlin [CHM+19] and $\mathcal{P}$lon$\mathcal{K}$ [GWC19].

The second group has the big advantage of only requiring a trusted setup once per circuit size bound, rather than once per circuit. This practical advantage makes it a frequent choice for real-world use.

The first group, however, has the following attractive feature the second usually lacks: As often put, it can do "addition gates for free".[1]

[1] The one caveat to this informal statement is that when starting with an input to an arithmetic circuit, **P** must compute and store an extended witness that contains the values of all intermediate circuit gates to feed into the R1CS corresponding to the circuit. This extension of the witness obviously depends on the number and fan-in of the addition gates.

Concretely, suppose we consider circuits with fan-in two multiplication gates and unlimited fan-in addition gates; and define somewhat oddly the *size* of a circuit as the number of *multiplication gates*. Say C is a circuit of size n under this definition. The approach of [GGPR13] allows us to prove knowledge of a satisfying assignment for C in $O(n \log n)$ operations, regardless of the number and structure of the addition gates!

A natural question is whether the advantages of both groups - cheap linear constraints and universal setup - can be combined?

In fact, the work of Groth, Kohlweiss, Maller, Meiklejohn, and Miers [GKM$^+$18] provides a positive answer to this question, with some caveats.

- For circuits of size n, [GKM$^+$18] requires a structured reference string (SRS) of size $O(n^2)$.

- Given a circuit C of size n, the process of "specializing" the SRS to C, enabling a prover to later run in time $O(n \log n)$, requires time $O(n^3)$.[2]

- The universality of the SRS is with respect to all circuits of size n rather than size *at most* n. Thus, to use an SRS generated for size n on circuits of size $m < n$, one must either pad the circuit and obtain a running time proportional to n, or run a setup for all circuit sizes that are powers of two up to n, resulting in an SRS of size $O(n^2 \log n)$.

In this work, we present a protocol called $\mathfrak{cq}\mathit{lin}$, that can improve upon the above caveats. Specifically, using $\mathfrak{cq}\mathit{lin}$ one can obtain a universal SNARK where

- For circuits of size *at most* n, we require a *single* trusted setup generating an SRS of size $O(n^2)$.

- Given a specific circuit of size $m \leq n$, the process of "specializing" the SRS to the circuit, enabling a prover to run in time $O(m \log m)$, requires time $O(m^2 \cdot \log m)$.

$\mathfrak{cq}\mathit{lin}$ in itself is not a full SNARK, but rather a protocol for a central component of SNARKs, called *lincheck*. This is exactly the component responsible for the caveats in [GKM$^+$18] mentioned above.

1.1 The lincheck problem and our results

We describe the problem formally in the context of KZG-commitments [KZG10]. For a subgroup $\mathbb{H} = \left\{\omega, \omega^2, \omega^n = 1\right\} \subset \mathbb{F}^*$ of order n and polynomial $f \in \mathbb{F}[X]$, define $f|_{\mathbb{H}}$ as[3] the vector $v \in \mathbb{F}^n$ with $v_i = f(\omega^i)$.

Fix a public matrix $M \in \mathbb{F}^{n \times n}$. Suppose a prover $\mathbf{P}$ has polynomials $f, g \in \mathbb{F}[X]$, and a verifier $\mathbf{V}$ has the commitments $\mathsf{cm}_1, \mathsf{cm}_2$ of f, g respectively. $\mathbf{P}$ now wants to convince $\mathbf{V}$ that $f|_{\mathbb{H}} \cdot M = g|_{\mathbb{H}}$.

[2] [GKM$^+$18] only says the cost of the specialization step is polynomial time; but this is what our calculations show.

[3] We assume some canonical way to choose a generator ω of $\mathbb{H}$, so that the definition of $f|_{\mathbb{H}}$ indeed depends only on f and $\mathbb{H}$.

This problem, in a more general setting of arbitrary polynomial commitment schemes, was dubbed *lincheck* in [BCR+19], following which [CHM+19, COS19] provided lincheck protocols with succinct verification. These papers gave protocols for lincheck where the number of prover operations was quasilinear in the number of non-zero entries in M, which can grow up to n^2.

For the special case of KZG commitments, $\mathfrak{cq}_{\mathit{lin}}$ requires only $O(n)$ $\mathbb{F}$-operations and $\mathbb{G}_1$-operations for $\mathbf{P}$, regardless of the structure of M.

1.2 Comparison to [GKM+18]

Though [GKM+18] does not explicitly discuss the lincheck problem, it implicitly contains a lincheck solution also requiring only $O(n)$ prover operations![4] The lincheck protocol derived from [GKM+18] compares to $\mathfrak{cq}_{\mathit{lin}}$ similarly to what was mentioned earlier regarding the resultant SNARKs:

- For lincheck on $n \times n$ matrices, [GKM+18] requires a trusted setup with SRS size $O(n^2)$ for each value of n, while $\mathfrak{cq}_{\mathit{lin}}$ requires *one* trusted setup of such SRS size for all values *up to* a given n.

- Given a matrix $M \in \mathbb{F}^{n \times n}$, the preprocessing step in $\mathfrak{cq}_{\mathit{lin}}$ requires $O(n^2 \cdot \log n)$ operations, while [GKM+18] requires $O(n^3)$ operations.

1.3 Usefulness of result and future work

The setup time makes $\mathfrak{cq}_{\mathit{lin}}$ less attractive for sparse matrices. Such matrices arise for example, when making an R1CS instance out of a circuit with constant fan-in addition gates, resulting in an $n \times n$ matrix with $O(n)$ non-zeroes. Dense matrices, as might arise in zkML, are a better fit for $\mathfrak{cq}_{\mathit{lin}}$.

Ideally, we would want a protocol merging the good qualities of [CHM+19, COS19] with those of $\mathfrak{cq}_{\mathit{lin}}$ and [GKM+18]. Namely, a lincheck protocol with setup time quasilinear in the matrix sparsity and prover time $O(n)$. This is an interesting direction for future research, and would imply a fully satisfying solution to combining the good features of both families of pairing-based SNARKs.

We now give a semi-formal overview of our protocol.

2 Technical Overview

Our starting point is observing that working with univariate polynomials of degree n^2, we can "simulate" matrix-vector multiplication with a combination of polynomial multiplication and modular reductions:

[4] It takes considerable work to extract the lincheck protocol. One step, for example, is reducing the case of checking $M \cdot f|_{\mathbb{H}} = g|_{\mathbb{H}}$ to checking $M' \cdot v|_{\mathbb{H}} = 0$ where M' is the matrix $(M, -I)$ and v's values on a subgroup of size $2n$ are the concatenation $(f|_{\mathbb{H}}, g|_{\mathbb{H}})$.

We represent the vector $a = (a_1, \ldots, a_n) \in \mathbb{F}^n$ by the polynomial $A(X) := \sum_{i\in[n]} a_i L_i(X^n)$, where $\{L_i\}$ is a Lagrange base of the subgroup $\mathbb{H}$ of order n.

Fix a matrix $M \in \mathbb{F}^{n\times n}$. For each $i \in [n]$ let $\mathsf{R}_i(X) := \sum_{j\in[n]} M_{i,j} L_j(X)$ be the polynomial representing the i'th row of M. Finally, represent M by the polynomial

$$M(X) := \sum_{i\in[n]} L_i(X^n)\mathsf{R}_i(X).$$

Observe that for $g \in \mathbb{F}_{<n}[X]$, $g|_{\mathbb{H}} = a \cdot M$ if and only $g(X) = \sum a_i \mathsf{R}_i(X)$. Thus, our task given commitments to A, g, M, is to show that g is indeed of this form.

As we'll see in Section 6, we have that the product $A(X)M(X)$ taken modulu $Z(X) := X^{n^2} - 1$ is equal to

$$R(X) = \sum a_i L_i(X^n)\mathsf{R}_i(X).$$

Thus, the polynomial $g(X)$ representing the matrix-vector product can be obtained (roughly) by modding out $R(X)$ by X^n.

A natural path to a protocol from these calculations, is having the prover $\mathbf{P}$ send a commitment r to R and show R satisfies the desired modular relations with A, M and g. To show such relations hold, $\mathbf{P}$ usually needs to compute quotient polynomials corresponding to the modular relations. However, these quotients have degree close to n^2 in our case, while we want to maintain an $O(n)$ prover. This is where cached quotients come into the picture.

The cached commitments and cached quotients techniques The cached quotients technique was introduced in [EFG22]. In fact, this technique is a special case of what we could call the "cached commitments method", that is implicitly used in many pairing-based protocols e.g. [ZBK$^+$22, GGPR13, GKM$^+$18].

The idea is simple: Suppose we have a polynomial-IOP where the prover sends a polynomial $f(X)$ of high degree but *low sparsity* in some pre-determined basis. Moreover, suppose f is involved only in verifier equations of degree at most two. Then we can precompute the KZG commitments to the basis elements - these precomputed commitments are what we call the *cached commitments.* Later, during protocol run time, we can compute the *commitment* to f from the cached commitments in a number of operations depending only on the sparsity. Now, the question is if having only the commitment to f is sufficient for our protocol. Then answer is yes - since f is involved only in degree at most two equations, they can be verified directly from commitments using pairings.

The cached quotients method from [EFG22] focuses on the following special case of this framework. We have predetermined polynomials $M(X), Z(X)$. $\mathbf{P}$ wishes, for a committed witness polynomial $A(X) \in \mathbb{F}[X]$, to prove correctness of the polynomial $R(X) = A(X) \cdot M(X) \bmod Z(X)$. The natural way to prove correctness of R is checking $\deg(R) < \deg(Z)$, and in addition supplying the quotient polynomial $Q(X)$ such that

$$A(X) \cdot M(X) = Q(X)Z(X) + R(X).$$

Note that $\deg(Q) = \deg(A) + \deg(M) - \deg(Z)$; which can be larger than the degree of polynomials we want to explicitly work with. For example, when setting A, M, Z as in the above setting of matrix-vector multiplication, $\deg(Q) = n^2 - n \sim n^2$.

Here an important observation is that if A is k-sparse in a basis of polynomials $\{B_i(X)\}$, i.e. $A(X) = \sum_{j\in[k]} c_{i_j} B_{i_j}(X)$ for some $\left\{c_{i_j}\right\}_{j\in[k]}$; then Q is k-sparse in the basis $\{Q_i\}$ of the polynomials $Q_i(X)$ such that

$$B_i(X) \cdot M(X) = Q_i(X)Z(X) + R_i(X),$$

for some R_i of degree smaller than $\deg(Z)$. Quite simply, we have $Q(X) = \sum_{j\in[k]} c_{i_j} Q_{i_j}(X)$. Thus, we can precompute the commitments of the $\{Q_i\}$, and during proving compute the commitment q of Q in only k group operations! Moreover, we can use this commitment to check correctness of r via

$$e(\mathsf{a}, \mathsf{m}) \stackrel{?}{=} e(\mathsf{q}, \mathbf{z})e(\mathsf{r}, [1]_2)$$

where $\mathsf{a} = [A(x)]_1, \mathsf{m} = [M(x)]_2, \mathbf{z} = [Z(x)]_2$. Note also that, similarly, $R(X) = \sum_{j\in[k]} c_{i_j} R_{i_j}(X)$. Thus r can also be computed in k operations if we “cache” the remainders $\{[R_i(x)]_1\}$.

Coming back to our lincheck protocol, we see that we have a good fit for the cached quotients method: Our witness polynomial A has degree close to n^2, but is n-sparse in the basis of functions $\{L_i(X^n)\}_{i\in[n]}$. Similarly, the polynomial R inherits A’s sparseness from the first modular reduction; and thus we can use the method again for proving the correctness of R’s reduction by X^n.

For the full details, see the subsequent sections, with the main protocol being presented in Section 8.

3 Preliminaries

3.1 Terminology and Conventions

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

Whenever we use the term *efficient*, we mean an algorithm running in time $\mathsf{poly}(\lambda)$. Furthermore, we assume an *object generator* $\mathcal{O}$ that is run with input λ before all protocols, and returns all fields and groups used. Specifically, in our protocol $\mathcal{O}(\lambda) = (\mathbb{F}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e, g_1, g_2, g_t)$ where

- $\mathbb{F}$ is a prime field of super-polynomial size $r = \lambda^{\omega(1)}$.
- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ are all groups of size r, and e is an efficiently computable non-degenerate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$.
- g_1, g_2 are uniformly chosen generators such that $e(g_1, g_2) = g_t$.

We usually let the λ parameter be implicit, i.e. write $\mathbb{F}$ instead of $\mathbb{F}(\lambda)$. We write $\mathbb{G}_1$ and $\mathbb{G}_2$ additively. We use the notations $[x]_1 := x \cdot g_1$ and $[x]_2 := x \cdot g_2$.

We often denote by $[n]$ the integers $\{1, \ldots, n\}$. We assume throughout the paper that n is a power of two. We use the acronym e.w.p for "except with probability"; i.e. e.w.p γ means with probability *at least* $1 - \gamma$.

universal SRS-based public-coin protocols We describe public-coin (meaning the verifier messages are uniformly chosen) interactive protocols between a prover and verifier; when deriving results for non-interactive protocols, we implicitly assume we can get a proof length equal to the total communication of the prover, using the Fiat-Shamir transform/a random oracle. Using this reduction between interactive and non-interactive protocols, we can refer to the "proof length" of an interactive protocol.

We allow our protocols to have access to a structured reference string (SRS) that can be derived in deterministic $\mathsf{poly}(\lambda)$-time from an "SRS of monomials" of the form $\left\{[x^i]_1\right\}_{a \le i \le b}$, $\left\{[x^i]_2\right\}_{c \le i \le d}$, for uniform $x \in \mathbb{F}$, and some integers a, b, c, d with absolute value bounded by $\mathsf{poly}(\lambda)$. It then follows from Bowe et al. [BGM17] that the required SRS can be derived in a universal and updatable setup requiring only one honest participant; in the sense that an adversary controlling all but one of the participants in the setup does not gain more than a $\mathsf{negl}(\lambda)$ advantage in its probability of producing a proof of any statement.

For notational simplicity, we sometimes use the SRS srs as an implicit parameter in protocols, and do not explicitly write it.

3.2 The algebraic group model

We introduce some terminology from [GWC19] to capture analysis in the Algebraic Group Model of Fuchsbauer, Kiltz and Loss[FKL18].

In our protocols, by an *algebraic adversary* $\mathcal{A}$ in an SRS-based protocol we mean a $\mathsf{poly}(\lambda)$-time algorithm which satisfies the following.

- For $i \in \{1, 2\}$, whenever $\mathcal{A}$ outputs an element $A \in \mathbb{G}_i$, it also outputs a vector v over $\mathbb{F}$ such that $A =< v, \mathsf{srs_i} >$.

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

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

By a "real pairing check" we mean a check of the form

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

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

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

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

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

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

$$[1]_1, [x]_1, \ldots, \left[x^Q\right]_1, [1]_2, [x]_2, \ldots, \left[x^Q\right]_2$$

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

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

See [GWC19] for the proof.

4 Preliminaries about polynomials

Let $\mathbb{H} \subset \mathbb{F}$ be a multiplicative subgroup of order n. Let $L_1(X), \ldots, L_n(X)$ be a Lagrange basis for $\mathbb{H}$. Namely, for a fixed generator $\omega \in \mathbb{H}$; $L_i(X)$ is the unique element of $\mathbb{F}_{<n}[X]$ with $L_i(\omega^i) = 1$ and $L_i(\omega^j) = 0$ for $i \neq j \in [n]$. Denote $Z(X) := X^{n^2} - 1$, and $Z_H(X) := X^n - 1$.

Operations on polynomials For polynomials $f(X), g(X) \in \mathbb{F}[X]$ where $d := \deg(g(X)) > 0$, we denote by $f(x) \bmod g(X)$ the unique polynomial $R(X) \in \mathbb{F}_{<d}[X]$ such that for some $q(X) \in \mathbb{F}[X]$,

$$f(X) = q(X) \cdot g(X) + R(X).$$

4.1 Power polynomials

We review some facts we will use for polynomials that are n'th powers.

Composed Lagranges We work with Lagrange polynomials composed internally with raising to the n'th power; i.e. polynomials of the form $L'_i(X) := L_i(X^n)$. We use the following claim.

Claim 4.1. *We have*

1. For any $i \in [n]$

$$L_i^2(X^n) = L_i(X^n) \bmod Z(X).$$

2. For any $i \neq j \in [n]$, we have

$$L_i(X^n) \cdot L_j(X^n) = 0 \bmod Z(X).$$

Proof. The claim follows from the similar more commony used one, without the composition with the n'th power, i.e. that

1. For any $i \in [n]$
$$L_i^2(X) = L_i(X) \bmod Z_H(X).$$

2. For any $i \neq j \in [n]$, we have
$$L_i(X) \cdot L_j(X) = 0 \bmod Z_H(X).$$

For example, note that the first item implies for each $i \in [n]$, that for some $Q(X) \in \mathbb{F}[X]$,

$$L_i^2(X) = Q(X) \cdot Z_H(X) + L_i(X)$$

Which implies

$$L_i^2(X^n) = Q(X^n) \cdot Z_H(X^n) + L_i(X^n)$$

Since $Z_H(X^n) = Z(X)$ and $\deg(L_i(X^n)) < n^2$, this indeed implies

$$L_i^2(X^n) = L_i(X^n) \bmod Z(X).$$

□

We'll use the following easy claim.

Claim 4.2. *Any polynomial $A \in \mathbb{F}_{<n^2}[X]$ that is of the form $A(X) = f(X^n)$ for $f \in \mathbb{F}[X]$ can be written as*

$$A(X) = \sum_{i \in [n]} a_i \cdot L_i(X^n)$$

for some $a_1, \ldots, a_n \in \mathbb{F}$.

Proof. We know that f can be written as

$$f(X) = \sum_{i \in [n]} a_i L_i(X),$$

for some $a_1, \ldots, a_n \in \mathbb{F}$ so we have

$$A(X) = f(X^n) = \sum_{i \in [n]} a_i L_i(X^n).$$

□

KZG for n'th powers: The following claim is easy to verify.

Claim 4.3. *Fix any $r, z \in \mathbb{F}$ and polynomial $P(X) \in \mathbb{F}[X]$.*

1. *If $(X^n - r^n)$ divides $P(X) - z$, then $P(r) = z$.*
2. *If $P(X) = A(X^n)$ for some $A \in \mathbb{F}[X]$, and $P(r) = z$, then $(X^n - r^n)$ divides $P(X)$.*

4.2 FFT "in the exponent"

We state without proof a few facts following from standard FFT techniques. They are based on FFT operations being linear, and thus possible to do on encoded elements $[a]_1$ without knowing the underlying scalar a. In the lemma we denote for $B \in \mathbb{F}[X]$, by $(B)_{[i]}$ the coefficient of X^i in $B(X)$.

Lemma 4.4. *For $B \in \mathbb{F}_{<n}[X]$, let $S_B := \left\{ \left[(B)_{[i]} \right]_1 \right\}_{i \in \{0,\ldots,n-1\}}$.*

1. *Given S_B we can compute in $O(n \log n)$ $\mathbb{F}$-operations and $\mathbb{G}_1$-operations the set $\{[B(a)]_1\}_{a \in \mathbb{H}}$.*
2. *Given $\{[B(a)]_1\}_{a \in \mathbb{H}}$ we can compute S_B in $O(n \log n)$ $\mathbb{F}$-operations and $\mathbb{G}_1$-operations.*

5 Extensions of the FK scheme

Feist and Khovratovich [FK, Tom] present an efficient method to compute all KZG opening proofs for a polynomial $P(X) \in \mathbb{F}[X]$ over a subgroup $\mathbb{H}$. Their result has become an important component in recent pairing-based schemes [ZBK+22, PK22, ZGK+22, EFG22]. We present a simple extension of their result that is important for the precomputations presented in the next section.

Lemma 5.1. *Fix a power of two n, and polynomials $A, P \in \mathbb{F}_{<n}[X]$. Let $\mathbb{H} \subset \mathbb{F}$ be a multiplicative subgroup of order n. Given the vector $\left(\left[A(x) \cdot x^{ni} \right]_1 \right)_{i \in \{0,\ldots,n-1\}}$ of $\mathbb{G}_1$ elements, we can compute in $O(n \log n)$ $\mathbb{G}_1$-operations the vector $\left(\left[\frac{A(x)(P(x^n) - P(a))}{x^n - a} \right]_1 \right)_{a \in \mathbb{H}}$.*

Proof. [FK] provide an algorithm computing the KZG proofs $\left\{ \left[\frac{P(x) - P(a)}{x - a} \right]_1 \right\}_{a \in \mathbb{H}}$ from the vector $\left(\left[x^i \right]_1 \right)_{i \in \{0,\ldots,n-1\}}$ in $O(n \log n)$ $\mathbb{G}_1$-operations. Since their algorithm only performs $\mathbb{F}$-linear operations, it follows that substituting $\left(\left[A(x) \cdot x^{ni} \right]_1 \right)_{i \in \{0,\ldots,n-1\}}$ as the input, and noticing that $\left[A(x) \cdot x^{ni} \right]_1 = A(x) \cdot \left[s^i \right]_1$ for $s := x^n$, gives the desired output. □

Next, we present a further extension of [FK] in a batched setting, where we want to compute a set of combinations of k opening proofs at each point $a \in \mathbb{H}$. "Combination" here is in the sense of Lemma 5.1 that the opening proof is multiplied by $A(x)$ for some polynomial $A(X)$. Invoking Lemma 5.1 independently k times and combining the results would give an algorithm requiring $O(kn \log n)$ field and group operations. We improve on this by giving an algorithm requiring only $O(kn)$ $\mathbb{G}_1$-operations after a one time $O(kn \log n)$ $\mathbb{G}_1$-operations computation independent of the polynomials being opened.

Lemma 5.2. *Fix integers k, n, and polynomials $A_1, \ldots, A_k \in \mathbb{F}_{<n}[X]$. Given* $\mathsf{srs} = \left\{ \left[x^i\right]_1 \right\}_{i \in \{0,\ldots,2n-1\}}$, *there is*

- *An algorithm $\mathscr{A}_0$ taking* $\mathsf{srs}, A_1, \ldots, A_k$ *as input that performs $O(kn \log n)$ $\mathbb{F}$-operations and $\mathbb{G}_1$-operations producing a vector* srs^* *of $\mathbb{G}_1$ elements.*
- *An algorithm $\mathscr{A}_1$ taking* srs^* *and any $P_1, \ldots, P_k \in \mathbb{F}_{<n}[X]$ as input, performing $O(kn \log n)$ $\mathbb{F}$-operations and $O(kn)$ $\mathbb{G}_1$-operations, and producing the $\mathbb{G}_1$-elements $\{\pi_a\}_{a \in \mathbb{H}}$, where*

$$\pi_a := \left[\sum_{j \in [k]} A_j(x) \cdot \frac{P_j(x) - P_j(a)}{x - a} \right]_1 .$$

Moreover, if $\mathsf{srs} = \left\{ \left[x^i\right]_1 \right\}_{i \in \{0,\ldots,n^2-1\}}$, *the same claim holds when $\mathscr{A}_1$ is computing instead the modified elements*

$$\pi'_a := \left[\sum_{j \in [k]} A_j(x) \cdot \frac{P_j(x^n) - P_j(a)}{x^n - a} \right]_1 .$$

Proof. Let $\mathbb{V} \subset \mathbb{F}$ be a subgroup of order $2n$. We begin by describing $\mathscr{A}_0$. Given srs, $\mathscr{A}_0$ first computes in $O(k \cdot n)$ operations the elements $\left\{ \left[A_j(x) x^i\right]_1 \right\}_{j \in [k], i \in \{0,\ldots,n-1\}}$

Now, before proceeding with $\mathscr{A}_0$, define the polynomial

$$T_x(Y) := \frac{x^n - Y^n}{x - Y} = \sum_{i=0}^{n-1} x^{n-1-i} Y^i .$$

We use Y instead of X as the variable to emphasize the srs secret x is simply a constant on which the coefficients of T_x depend. $\mathscr{A}_0$ computes in $O(kn \log n)$ operations the set $\mathsf{srs}^* := \left\{ [A_j(x) T_x(a)]_1 \right\}_{j \in [k], a \in \mathbb{V}}$

We move to $\mathscr{A}_1$. For deductic purposes, assume for a moment we simply wanted to compute the KZG proofs of some $D \in \mathbb{F}_{<n}[X]$ over $\mathbb{H}$. That is, the set of elements $\mathsf{KZG}_{D,\mathbb{H}} := \left\{ \left[\frac{D(x) - D(a)}{x - a} \right]_1 \right\}_{a \in \mathbb{H}}$. We define a polynomial $h_D(Y)$ related to D as

$$h_D(Y) := \frac{D(x) - D(Y)}{x - Y} .$$

The important point is that for any $a \in \mathbb{F}, [h_D(a)]_1$ is precisely the KZG opening proof of D at a! And so $\mathsf{KZG}_{D,\mathbb{H}} = \{[h_D(a)]_1\}_{a \in \mathbb{H}}$. Thus, we need to compute the encodings of h_D's values on $\mathbb{H}$. Moreover, we need to do this without direct access to h_D's coefficients as they depend on x.

For this purpose, we multiply the equation defining h_D by $(x^n - Y^n)$ and obtain

$$h_D(Y)(x^n - Y^n) = \frac{D(x) - D(Y)}{x - Y} \cdot (x^n - Y^n) = (D(x) - D(Y)) \cdot T_x(Y).$$

Shifting terms we get

$$h_D(Y)Y^n = D(Y)T_x(Y) - D(x)T_x(Y) + h_D(Y)x^n.$$

Note that on the RHS, all monomials of degree at least n in Y come from the term $D(Y)T_x(Y)$. It follows that the last $n-1$ coefficients of $D \cdot T_x$ are the coefficients of h_D; i.e. $(h_D)_{[i]} = (D \cdot T_x)_{[n+i]}$ for $i = 0, \ldots, n-2$. Thus, given the values $\{[D(a)T_x(a)]_1\}_{a \in \mathbb{V}}$, we can apply Lemma 4.4 to obtain $\left\{ \left[(h_D)_{[i]}\right]_1 \right\}_{i \in \{0,\ldots,n-2\}}$, and apply it again to obtain $\{[h_D(a)]_1\}_{a \in \mathbb{H}}$.

To summarize - to compute $\mathsf{KZG}_{D,\mathbb{H}}$ for $D \in \mathbb{F}_{<n}[X]$ it suffices to obtain $\{[D(a)T_x(a)]_1\}_{a \in \mathbb{V}}$.

Coming back to what $\mathscr{A}_1$ needs to compute, define $c_j := A_j(x)$ for $j \in [k]$, and $D \in \mathbb{F}_{<n}[X]$ as

$$D(Y) := \sum_{j \in [k]} c_j P_j(Y).$$

So we have

$$h_D(Y) = \sum_{j \in [k]} c_j \cdot \frac{P_j(x) - P_j(Y)}{x - Y}$$

Note that $\pi_a = [h_D(a)]_1$ for $a \in \mathbb{H}$ so $\mathscr{A}_1$ in fact needs to compute $\mathsf{KZG}_{D,\mathbb{H}}$. As explained above, for this purpose it suffices to compute $\{[D(a)T_x(a)]_1\}_{a \in \mathbb{V}}$. Hence, we need to show how this set can be computed in $O(kn)$ $\mathbb{G}_1$-operations and $O(kn \log n)$ $\mathbb{F}$-operations from srs^*. This is done as follows.

1. For each $j \in [k]$, compute in $O(n \log n)$ $\mathbb{F}$-operations the set of field elements $\{P_j(a)\}_{a \in \mathbb{V}}$.

2. For each $j \in [k], a \in \mathbb{V}$, compute the element $[c_j P_j(a) T_x(a)]_1$, as a product of the element $P_j(a)$ computed in the previous step, and the element $[c_j T_x(a)]_1$ from srs^*.

3. Compute for each $a \in \mathbb{V}$, $[D(a)T_x(a)]_1$ as a sum of k elements from the previous step:
$$[D(a)T_x(a)]_1 = \sum_{j \in [k]} [c_j P_j(a) T_x(a)]_1 .$$

Finally, the "moreover" part of the lemma is derived by having $\mathscr{A}_0$ initially compute the elements $\left\{A_j(x)x^{ni}\right\}$ rather than $\left\{A_j(x)x^i\right\}$. Since all operations are linear, if $\mathscr{A}_0$ and $\mathscr{A}_1$ proceed in the same way, we end up with $\{\pi'_a\}$ rather than $\{\pi_a\}$.

□

6 Representing matrix-vector multiplication with univariate polynomials

Given a matrix $M \in \mathbb{F}^{n\times n}$ we identify M with the univariate polynomial $M \in \mathbb{F}_{<n^2}[X]$ defined as

$$M(X) := \sum_{i\in[n]} L_i(X^n)\mathsf{R}_i(X),$$

where for $i \in [n]$ we define

$$\mathsf{R}_i(X) := \sum_{j\in[n]} M_{i,j}L_j(X).$$

When "simulating" a matrix-vector mutiplication via polynomial multiplication, we use a polynomial containing only n'th powers. Specifically, to represent the vector $a \in \mathbb{F}^n$ we use the polynomial

$$A(X) := \sum_{i\in[n]} a_i \cdot L_i(X^n).$$

The following claim follows from Claim 4.1.

Claim 6.1. *For any $M(X), A(X) \in \mathbb{F}[X]$ defined as above we have*

$$A(X) \cdot M(X) = \sum_{i\in[n]} a_i \cdot L_i(X^n) \cdot \mathsf{R}_i(X) \bmod Z(X).$$

Proof. One thing to notice is that indeed $\deg(L_i(X^n)\mathsf{R}_i(X)) = n \cdot (n-1) + (n-1) = n^2 - 1 < n^2$. □

7 Precomputing cached quotients

We define the cached quotients, and also some "cached remainders", we use in our protocol.

1. For $i \in [n]$, let $Q_i(X), R_i(X) = L_i(X^n)\mathsf{R}_i(X) \in \mathbb{F}_{<n^2}[X]$ be the unique polynomials such that
$$L_i(X^n) \cdot M(X) = Q_i(X)Z(X) + R_i(X).$$
2. For $i \in [n]$, define $S_i(X) \in \mathbb{F}_{<n^2}[X]$ to be the unique polynomial such that
$$(L_i(X^n) - 1/n) \cdot \mathsf{R}_i(X) = X^n S_i(X).$$

Lemma 7.1. *Suppose $\mathsf{srs} \in \mathbb{G}_1^{n^2}$ is of the form $\mathsf{srs} = \left\{\left[x^i\right]_1\right\}_{i\in[0..n^2-1]}$ for some $x \in \mathbb{F}$. There is an algorithm that given srs performs $O(n^2 \log n)$ $\mathbb{F}$- and $\mathbb{G}_1$-operations and produces a vector srs^* of $\mathbb{G}_1$-elements such that the following holds.*

Given srs^* and any $M \in \mathbb{F}^{n\times n}$, there is an algorithm computing the elements

$$\{[Q_i(x)]_1, [R_i(x)]_1, [S_i(x)]_1\}_{i\in[n]}$$

in $O(n^2)$ $\mathbb{G}_1$-operations and $O(n^2 \log n)$ $\mathbb{F}$-operations.

Proof. srs^* consists of the elements

- For each $i, j \in [n]$, $[U_{i,j}(x)]_1 := [L_i(x^n)L_j(x)]_1$.
- For each $j \in [n]$, $[V_j(x)]_1 := \left[x^{n^2-n}L_j(x)/n\right]_1$,
- When setting $k := n$, and $A_j(X) := L_j(X)$ for $j \in [n]$, the elements computed by $\mathscr{A}_0$ in the "Moreover" part of Lemma 5.2.

These can all be computed in $O(n^2 \log n)$ group operations independently of the matrix M. Note $W_{i,2j}(x) = U_{i,j}(x)$ and $W_{i,2j+1}(x) = U_{i,j}(\omega_{2n}^{-1}x)$.

$\underline{S_i(x)}$: To compute the $[S_i(x)]_1$ commitments, the setup algorithm will exploit an identity also used in [EFG22]:

$$L_i(X) - 1/n = x(\omega^{-i}L_i(X) - X^{n-1}/n).$$

composing internally with X^n, and multiplying with $L_j(X)$ we get

$$(L_i(X^n)-1/n)\cdot L_j(X) = X^n(\omega^{-i}L_i(X^n)-X^{n^2-n}/n)\cdot L_j(X) = \omega^{-i}X^n(U_{i,j}(X)-V_j(X)).$$

Which implies for each $i \in [n]$ that

$$S_i(X) = \sum_{j\in[n]} M_{i,j} \cdot \omega^{-i}(U_{i,j}(X) - V_j(X)).$$

Letting N be the number of non-zero entries in M. Thus, given srs^* we can compute in $O(N) = O(n^2)$ group operations for all $i \in [n]$, the element

$$[S_i(x)]_1 = \sum_{j=0}^{n-1} M_{i,j} \cdot (\omega^{-i}\,[U_{i,j}(x)]_1 - [V_j(x)]_1).$$

$\underline{R_i(x)}$: We can easily compute $[R_i(x)]_1$ from srs^* as

$$[R_i(x)]_1 = \sum_{j\in[n]} M_{i,j}\,[L_i(x^n)\cdot L_j(x)]_1 = \sum_{j\in[n]} M_{i,j}U_{i,j}.$$

$\underline{Q_i(x)}$:

To compute the quotients $[Q_i(x)]_1$ we show this reduces to the setting of Lemma 5.2. First, we set $k = n$, and $A_j(X) = L_j(X)$ for $j \in [n]$. We write $M(X)$ as a combination of *column* polynomials:

$$M(X) = \sum_{j\in[n]} \mathsf{C}_j(X^n)L_j(X),$$

where $\mathsf{C}_j(X) := \sum_{i\in[n]} M_{i,j}L_i(X)$.

We will compute Q_i as a sum of "per-column" quotients. Specifically, for $i,j \in [n]$ define $Q_{i,j}(X)$ as the unique solution to

$$L_i(X^n)\left(\mathsf{C}_j(X^n)L_j(X)\right) = L_i(X^n)\left(M_{i,j}L_j(X)\right) + Z(X)Q_{i,j}(X).$$

Note that summing the above equation for fixed $i \in [n]$ over all $j \in [n]$ gives

$$L_i(X^n)M(X) = R_i(X) + Z(X) \cdot \sum_{j\in[n]} Q_{i,j}(X),$$

so that we indeed have $Q_i(X) = \sum_{j\in[n]} Q_{i,j}(X)$.

Rearranging terms in the equation defining $Q_{i,j}$, and using $\mathsf{C}_j(\omega^i) = M_{i,j}$, we get

$$Q_{i,j}(X) = \frac{L_j(X)L_i(X^n)(\mathsf{C}_j(X^n) - \mathsf{C}_j(\omega^i))}{Z(X)}.$$

We now use a technique from Lemma 3.1 of [EFG22], but adapted to n'th powers: Using the identity $L_i(X^n) = c_i \cdot \frac{Z(X)}{X^n - \omega^i}$ for the constant $c_i := (\omega^i/n)$, we can rewrite $Q_{i,j}$ as a modified form of the KZG opening of $\mathsf{C}_j(X)$ at ω^i.

$$Q_{i,j}(X) = c_i L_j(X) \cdot \frac{\mathsf{C}_j(X^n) - \mathsf{C}_j(\omega^i)}{X^n - \omega^i}.$$

And thus

$$Q_i(X) = c_i \cdot \sum_{j\in[n]} L_j(X) \cdot \frac{\mathsf{C}_j(X^n) - \mathsf{C}_j(\omega^i)}{X^n - \omega^i}.$$

At this point, we can see that computing the set $\{[Q_i(x)]_1\}_{i\in[n]}$, excluding multiplication by the scalars c_i which we can do separately, corresponds to the setting of Lemma 5.2: Set $k = n$, and for each $j \in [n]$ set $A_j(X) = L_j(X)$ and $P_j(X) = \mathsf{C}_j(X)$. □

8 Main Protocol

We give a natural definition of a lincheck protocol secure in the algebraic group model.

Definition 8.1. *A* lincheck protocol *is a pair* $\mathscr{P} = (\mathsf{gen}, \mathsf{lincheck})$ *such that*

- $\mathsf{gen}(n, M)$ *is a randomized algorithm receiving as input parameters integer* n *and* $M \in \mathbb{F}^{n\times n}$*. Given these inputs* gen *outputs a string* srs_M *of* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *elements.*
- $\mathsf{lincheck}(\mathsf{f}, \mathsf{g}, M, \mathsf{srs}_M, \mathbb{H}; f(X), g(X))$ *is an interactive public coin protocol between* $\mathbf{P}$ *and* $\mathbf{V}$ *where* $\mathbf{P}$ *has private input* $f, g \in \mathbb{F}_{<n}[X]$*, and both parties have access to* f, g *and* $\mathsf{srs}_M = \mathsf{gen}(n, M)$*; such that*
  - ***Completeness:*** *If* $\mathsf{f} = [f(x)]_1, \mathsf{g} = [g(x)]_1$ *and* $f|_{\mathbb{H}} \cdot M = g|_{\mathbb{H}}$ *then* $\mathbf{V}$ *outputs* acc *with probability one.*

- ***Knowledge soundness in the algebraic group model:*** *The probability of any efficient algebraic $\mathcal{A}$ winning the following game is $\mathsf{negl}(\lambda)$.*
  1. *$\mathcal{A}$ chooses parameter n and a matrix $M \in \mathbb{F}^{n \times n}$.*
  2. *We compute $\mathsf{srs}_M = \mathsf{gen}(n, M)$ and send it to $\mathcal{A}$.*
  3. *$\mathcal{A}$ sends messages f, g and $f, g \in \mathbb{F}_{<d}[X]$ such that $\mathsf{f} = [f(x)]_1, \mathsf{g} = [g(x)]_1$ where d is the smallest integer such that all $\mathbb{G}_1$ elements in srs_M are linear combinations of $\left\{ [x^i]_1 \right\}_{i \in \{0,\ldots,d-1\}}$.*
  4. *$\mathcal{A}$ and $\mathbf{V}$ engage in the protocol $\mathsf{lincheck}(\mathsf{f}, \mathsf{g}, M, \mathsf{srs}_M, \mathbb{H}; f(X), g(X))$, with $\mathcal{A}$ taking the role of $\mathbf{P}$.*
  5. *$\mathcal{A}$ wins if*
     - *$\mathbf{V}$ outputs acc, and*
     - *$f|_{\mathbb{H}} \cdot M \neq g|_{\mathbb{H}}$.*

We are now ready to present our main protocol.

8.1 The $\mathfrak{cq}\mathit{lin}$ protocol

$\underline{\mathsf{gen}(n, M)\text{:}}$

1. Choose random $x \in \mathbb{F}$ compute and output $\left\{ [x^i]_1 \right\}_{i \in \{0,\ldots,n^2-1\}}, \left\{ [x^i]_2 \right\}_{i \in \{0,\ldots,n^2\}}$.
2. Compute and output the elements $\left\{ \left[x^{n^2-n} \cdot L_i(x) \right]_1, [L_i(x^n)]_1 \right\}_{i \in [n]}, \left\{ [L_i(x^n) L_j(x)]_1 \right\}_{i,j \in [n]}$
3. Compute and output $\mathbf{z} := [Z(x)]_2$.
4. As explained in Section 6, identify the matrix M, with the polynomial
$$M(X) := \sum_{i \in [n]} L_i(X^n) \mathsf{R}_i(X),$$
where for $i \in [n]$ we define
$$\mathsf{R}_i(X) := \sum_{j \in [n]} M_{i,j} L_j(X)$$
Compute and output $\mathsf{m} := [M(x)]_2$.
5. For $i \in [n]$, using the method explaind in Lemma 7.1, compute and output:
   (a) $\mathsf{r}_i = [R_i(x)]_1$ where $R_i(X) := L_i(X^n) \mathsf{R}_i(X)$
   (b) $\mathsf{q}_i = [Q_i(x)]_1$ such that
$$L_i(X^n) \cdot M(X) = Q_i(X) Z(X) + R_i(X),$$

(c) $\mathsf{s}_i = [S_i(x)]_1$ such that

$$(L_i(X^n) - 1/n) \cdot \mathsf{R}_i(X) = X^n S_i(X)$$

<u>$\mathsf{lincheck}(\mathsf{srs}_M, \mathsf{f}, \mathsf{g}; f(X), g(X))$:</u>

Round 1: Checking matrix multiplication is correct and degree checking g

1. Let $A(X) := f(X^n) = \sum_{i \in [n]} a_i \cdot L_i(X^n)$. $\mathbf{P}$ computes and sends $\mathsf{a} := [A(x)]_1$.
2. $\mathbf{P}$ computes $\mathsf{r} := [R(x)]_1$, and $\mathsf{q} := [Q(x)]_1$, where $R, Q \in \mathbb{F}_{<n^2}[X]$ are such that
$$A(X) \cdot M(X) = Z(X) \cdot Q(X) + R(X).$$
3. $\mathbf{P}$ sends r, q.
4. $\mathbf{V}$ checks correctness of r via the pairing check
$$e(\mathsf{a}, \mathsf{m}) = e(\mathsf{q}, \mathbf{z}) \cdot e(\mathsf{r}, [1]_2).$$
5. $\mathbf{P}$ computes $\mathsf{s} := [S(x)]_1$ where $S \in \mathbb{F}_{<n^2}[X]$ is such that
$$R(X) - (1/n) \cdot g(X) = S(X) \cdot X^n.$$
6. $\mathbf{V}$ checks that $R(X) = (1/n) \cdot g(X) \bmod X^n$ via the pairing check
$$e(\mathsf{r} - (1/n) \cdot \mathsf{g}, [1]_2) = e(\mathsf{s}, [x^n]_2)$$
7. $\mathbf{P}$ computes $\mathsf{p} := [P(x)]_1$ where $P(X) := g(X) \cdot X^{n^2 - n}$.
8. $\mathbf{V}$ checks that $\deg(g) < n$ via the pairing check
$$e(\mathsf{g}, \left[x^{n^2-n}\right]_2) = e(\mathsf{p}, [1]_2)$$

Round 2: Checking that $A(X) = f(X^n)$ on a random challenge

1. $\mathbf{V}$ chooses and sends random $\gamma \in \mathbb{F}$.
2. Let $z := f(\gamma^n) = A(\gamma)$. $\mathbf{P}$ computes $\pi = [h(x)]_1$ where
$$h(X) := \frac{f(X) - z}{X - \gamma^n}$$

3. and $\pi_1 = [h_1(x)]_1$ where
$$h_1(X) := \frac{A(X) - z}{X^n - \gamma^n}$$

4. $\mathbf{V}$ checks that $A(\gamma) = f(\gamma^n)$ via the pairing checks:
$$e(\mathsf{f} - [z]_1 + \gamma^n \cdot \pi, [1]_2) = e(\pi, [x]_2),$$
$$e(\mathsf{a} - [z]_1 + \gamma^n \cdot \pi_1, [1]_2) = e(\pi_1, [x^n]_2).$$

5. $\mathbf{V}$ outputs rej if any of the pairing checks failed, and acc otherwise.

For the rest of the section we will prove

Theorem 8.2. $\mathfrak{cq}\mathit{lin} = (\mathsf{gen}, \mathsf{lincheck})$ *as described above, is a lincheck protocol where*

- *Running* $\mathsf{gen}(n, M)$ *for any matrix* M *requires* $O(n^2 \log n)$ $\mathbb{F}$*-operations and* $O(n^2)$ $\mathbb{G}_1$*-operations, after an initial computation of* $O(n^2 \log n)$ $\mathbb{F}$*-operations and* $\mathbb{G}_1$*-operations.*
- *In* $\mathsf{lincheck}$*,*
  1. $\mathbf{P}$ *requires* $O(n)$ $\mathbb{F}$*-operations and* $7n$ $\mathbb{G}_1$*-operations.*
  2. $\mathbf{V}$ *requires six pairings.*

The claim on the runtime of gen follows from Lemma 7.1. The claim that $\mathbf{V}$ only requires six pairings uses the standard method of randomly combining pairings with identical $\mathbb{G}_2$ arguments. The different $\mathbb{G}_2$ arguments are m,z,$[1]_2$,$[x^n]_2$,$\left[x^{n^2-n}\right]_2$,$[x]_2$.

Regarding prover runtime - $\mathbf{P}$ computes the seven commitments a,r,q,s,p,π,π_1, where all respective polynomials are n-sparse in bases for which we have precomputed the commitments. One thing to notice is that the coefficients of h_1 in the basis $\{L_i(X^n)\}$ are the same as the coefficients of h in the basis $\{L_i\}$; and the latter can be computed in $O(n)$ $\mathbb{F}$-operations. We proceed to show the knowledge soundness property holds.

Knowledge soundness proof: Let $\mathcal{A}$ be an efficient algebraic adversary participating in the Knowledge Soundness game from Definition 8.1.

We show its probability of winning the game is $\mathsf{negl}(\lambda)$. Let $f, g \in \mathbb{F}_{<n^2}[X]$ be the polynomials sent by $\mathcal{A}$ in the third step of the game such that $\mathsf{f} = [f(x)]_1$, $\mathsf{g} = [g(x)]_1$. As $\mathcal{A}$ is algebraic, when sending the commitments r,a,q,s,p,π,π_1 during protocol execution it also sends polynomials $R, A, Q, S, P, h, h_1 \in \mathbb{F}_{<n^2}[X]$ such that the former are their corresponding commitments. Let E' be the event that $\mathbf{V}$ outputs acc. Let E be the event that $\mathcal{A}$ wins the knowledge soundness game. Note that $E \subset E'$. We show that E is contained in the union of two events E_0, E_1 each of probability $\mathsf{negl}(\lambda)$.

E implies all pairing checks have passed. Let $E_0 \subset E$ be the subevent where one of the corresponding ideal pairing checks as defined in Section 3.2 didn't pass. According to Lemma 3.2, $\Pr(E_0 = \mathsf{negl}(\lambda))$. Let E_1 be the event that E_0 didn't happen and

$A(X) \neq f(X^n)$. Given that E_0 didn't occur, we have from the checks in step 4 of Round 2, that:

$$f(X) - z = h(X) \cdot (X - \gamma^n), A(X) - \mathsf{z} = h_1(X)(X^n - \gamma^n);$$

which implies according to Claim 4.3 that $f(\gamma^n) = z$ and $A(\gamma) = z$. If $A(X) \neq f(X^n)$ there are less than n^3 choices of $\gamma \in \mathbb{F}$ such that the above holds. Hence the probability of E_1 is at most $n^3/|\mathbb{F}| = \mathsf{negl}(\lambda)$. Now assume that both E_0 and E_1 didn't occur. Thus we have $A(X) = \sum_{i\in[n]} a_i L_i(X^n)$ where $\{a_i\}$ are such that $f(X) = \sum_{i\in[n]} a_i L_i(X)$. We have

- From Round 1, Step 4

  $$A(X)M(X) = Q(X)Z(X) + R(X)$$

  since $\deg(R) < n^2$ this means, by claims 4.1,6.1 that

  $$R(X) = A(X)M(X) \bmod Z(X) = \sum_{i\in[n]} a_i L_i(X^n) C_i(X).$$

- From Round 1, step 8, we have that $P(X) = g(X) \cdot X^{n^2-n}$, and so $\deg(g) < n$.

- From Round 1, step 6, we have that

  $$R(X) - (1/n) \cdot g(X) = S(X) \cdot X^n.$$

  Since $\deg(g) < n$, this means that

  $$(1/n)g(X) = R(X) \bmod X^n = \sum_{i\in[n]} a_i(1/n) \cdot C_i(X)$$

  And so $g(X) = \sum_{i\in[n]} a_i \cdot C_i(X)$, which means $g|_{\mathbb{H}} = f|_{\mathbb{H}} \cdot M$. In summary, we have shown the event that $\mathcal{A}$ wins the knowledge soundness game is contained in two negligible probability events E_0, E_1 and so knowledge soundness holds for $\mathfrak{cq}_{\mathit{lin}}$.

Acknowledgements

The second author thanks the Ethereum Foundation for a grant supporting this work. We thank Pratyush Mishra for bringing up the connection to [GKM+18]. We thank Josh Beal and Pun Waiwitlikhit for a corrections.

References

[BCR+19] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, pages 103–128, 2019.

[BGM17] S. Bowe, A. Gabizon, and I. Miers. Scalable multi-party computation for zk-snark parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017. `https://eprint.iacr.org/2017/1050`.

[CHM$^+$19] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zksnarks with universal and updatable SRS. *IACR Cryptology ePrint Archive*, 2019:1047, 2019.

[COS19] A. Chiesa, D. Ojha, and N. Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. *IACR Cryptology ePrint Archive*, 2019:1076, 2019.

[EFG22] L. Eagen, D. Fiore, and A. Gabizon. cq: Cached quotients for fast lookups. *IACR Cryptol. ePrint Arch.*, page 1763, 2022.

[FK] D. Feist and D. Khovratovich. Fast amortized kate proofs.

[FKL18] G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 33–62, 2018.

[GGPR13] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

[GKM$^+$18] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-snarks. *IACR Cryptology ePrint Archive*, 2018.

[Gro16] J. Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 305–326, 2016.

[GWC19] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptology ePrint Archive*, 2019:953, 2019.

[KZG10] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. pages 177–194, 2010.

[MBKM19] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. *IACR Cryptology ePrint Archive*, 2019:99, 2019.

[PK22] J. Posen and A. A. Kattis. Caulk+: Table-independent lookup arguments. 2022.

[Tom] A. Tomescu. Feist-khovratovich technique for computing kzg proofs fast.

[ZBK$^+$22] A. Zapico, V. Buterin, D. Khovratovich, M. Maller, A. Nitulescu, and M. Simkin. Caulk: Lookup arguments in sublinear time. *IACR Cryptol. ePrint Arch.*, page 621, 2022.

[ZGK$^+$22] A. Zapico, A. Gabizon, D. Khovratovich, M. Maller, and C. Ràfols. Baloo: Nearly optimal lookup arguments. *IACR Cryptol. ePrint Arch.*, page 1565, 2022.