$\mathfrak{cq}$:* Cached quotients for fast lookups

Liam Eagen
Blockstream

Dario Fiore
IMDEA software institute

Ariel Gabizon
Zeta Function Technologies

January 8, 2023

Abstract

We present a protocol for checking the values of a committed polynomial $f(X) \in \mathbb{F}_{<n}[X]$ over a multiplicative subgroup $\mathbb{H} \subset \mathbb{F}$ of size n are contained in a table $\mathsf{t} \in \mathbb{F}^N$. After an $O(N \log N)$ preprocessing step, the prover algorithm runs in time $O(n \log n)$. Thus, we continue to improve upon the recent breakthrough sequence of results[ZBK+22, PK22, GK22, ZGK+22] starting from Caulk [ZBK+22], which achieve sublinear complexity in the table size N. The two most recent works in this sequence [GK22, ZGK+22] achieved prover complexity $O(n \cdot \log^2 n)$.

Moreover, $\mathfrak{cq}$ has the following attractive features.

1. As in [ZBK+22, PK22, ZGK+22] our construction relies on homomorphic table commitments, which makes them amenable to vector lookups.
2. As opposed to [ZBK+22, PK22, GK22, ZGK+22] the $\mathfrak{cq}$ verifier doesn't involve pairings with prover defined $\mathbb{G}_2$ points, which makes recursive aggregation of proofs more convenient.

1 Introduction

The *lookup problem* is fundamental to the efficiency of modern zk-SNARKs. Somewhat informally, it asks for a protocol to prove the values of a committed polynomial $f(X) \in \mathbb{F}_{<n}[X]$ are contained in a table T of size N of predefined legal values. When the table T corresponds to an operation without an efficient low-degree arithmetization in $\mathbb{F}$, such a protocol produces significant savings in proof construction time for programs containing the operation. Building on previous work of [BCG+18], $\mathfrak{plookup}$ [GW20] was the first to explicitly describe a solution to this problem in the polynomial-IOP context. $\mathfrak{plookup}$ described a protocol with prover complexity quasilinear in both n and N. This left the intriguing question of whether the dependence on N could be made *sublinear*

*Pronounced "seek you".

after performing a preprocessing step for the table T. Caulk [ZBK+22] answered this question in the affirmative by leveraging bi-linear pairings, achieving a run time of $O(n^2 + n \log N)$. Caulk+ [PK22] improved this to $O(n^2)$ getting rid of the dependence on table size completely.[1]

Naturally, the quadratic dependence on n of these works made them impractical for a circuit with many lookup gates. This was resolved in two more recent protocols - ba*loo* [ZGK+22] and Flookup [GK22] achieving a runtime of $O(n \log^2 n)$. While Flookup has better concrete constants, ba*loo* preserved an attractive feature of Caulk - using a *homomorphic commitment* to the table. This means that given commitments $\mathsf{cm}_1, \mathsf{cm}_2$ to tables T_1, T_2 with elements $\{a_i\}$,$\{b_i\}$ respectively; we can check membership in the set of elements $\{a_i + \alpha b_i\}$ by running the protocol with $\mathsf{cm} := \mathsf{cm}_1 + \alpha \cdot \mathsf{cm}_2$ as the table commitment. This is crucial for vector lookups that have become popular in zk-SNARKs, as described in Section 4 of [GW20].

One drawback of all four recent constructions - Caulk,Caulk+,ba*loo*,Flookup; is that they require the verifier perform a pairing where both $\mathbb{G}_1$ and $\mathbb{G}_2$ pairing arguments are not fixed in the protocol, but prover defined. This makes it harder to recrusively aggregate multiple proofs via random combination, in the style described e.g. in Section 8 of [BCMS20].

1.1 Our results

In this paper, we present a protocol called cq - short for "cached quotients" which is a central technical component in the construction (and arguably in all four preceding works). cq

1. Improves asymptotic prover performance in field operations from $O(n \log^2 n)$ to $O(n \log n)$, and has smaller constants in group operations and proof size compared to ba*loo*.

2. Uses homomorphic table commitments similarly to Caulk,Caulk+ and ba*loo*, enabling convenient vector lookups.

3. Achieves for the first time in this line of work convenient aggregatability by having all verifier pairings use fixed protocol-defined $\mathbb{G}_2$ arguments.

Table 1: Scheme comparison. n = witness size, N = Table size, "Aggregatable"= All prover defined pairing arguments are in $\mathbb{G}_1$

Scheme	Preprocessing	Proof size	Prover Work	Verifier Work	Homomorphic?	Aggregatable?
Caulk [ZBK+22]	O($N \log N$) $\mathbb{F}$,$\mathbb{G}_1$	14 $\mathbb{G}_1$, 1 $\mathbb{G}_2$, 4 $\mathbb{F}$	$O(n^2 + n \cdot \log(N))$ $\mathbb{F}$,$\mathbb{G}_1$	$4P$	✓	✗
Caulk+ [PK22]	$O(N \log N)$ $\mathbb{F}$,$\mathbb{G}_1$	7 $\mathbb{G}_1$, 1 $\mathbb{G}_2$, 2 $\mathbb{F}$	$O(n^2)$ $\mathbb{F}$,$\mathbb{G}_1$	$3P$	✓	✗
Flookup [GK22]	$O(N \log^2 N)$ $\mathbb{F}$,$\mathbb{G}_1$	6 $\mathbb{G}_1$, 1 $\mathbb{G}_2$, 4 $\mathbb{F}$	$6n$ $\mathbb{G}_1$, n $\mathbb{G}_2$, $O(n \log^2 n)$ $\mathbb{F}$	$3P$	✗	✗
ba*loo* [ZGK+22]	$O(N \log N)$ $\mathbb{F}$,$\mathbb{G}_1$	12 $\mathbb{G}_1$, 1 $\mathbb{G}_2$, 4 $\mathbb{F}$	$13n$ $\mathbb{G}_1$ n $\mathbb{G}_2$, $O(n \log^2 n)$ $\mathbb{F}$	$5P$	✓	✗
cq (this work)	$O(N \log N)$ $\mathbb{F}$,$\mathbb{G}_1$	8 $\mathbb{G}_1$, 3 $\mathbb{F}$	$8n$ $\mathbb{G}_1$, $O(n \log n)$ $\mathbb{F}$	$5P$	✓	✓

[1]A nuance is that while the *number* of field and group operations are independent of table size, the field and group must be larger than the table in all these constructions, including this paper.

1.2 Technical Overview

We explain our protocol in the context of the line of work starting from [ZBK$^+$22].

The innovation of Caulk To restate the problem, we have an input polynomial $f(X)$, a table $\mathfrak{t}$ of size N encoded as the values of a polynomial $T(X) \in \mathbb{F}_{<N}[X]$ on a subgroup $\mathbb{V}$ of size N. We want to show f's values on a subgroup $\mathbb{H}$ of size n are contained in $\mathfrak{t}$; concisely that $f|_{\mathbb{H}} \subset \mathfrak{t}$. We think of the parameters as $n << N$. We want our prover $\mathbf{P}$ to perform a number of operations *sublinear* in N, or ideally, a number of operations depending only on n.

One natural approach - is to send the verifier $\mathbf{V}$ a polynomial T_f encoding the *n values from $\mathfrak{t}$ actually used in f*, and then run a lookup protocol using T_f. The challenging problem is then to prove T_f *actually encodes values from T*. Speaking impercisely, the "witness" to T_f's correctness is a quotient Q of degree $N-n$. It would defeat our purpose to actually compute Q - as that would require $O(N)$ operations.

The central innovation of Caulk [ZBK$^+$22] is the following observation: If we precompute commitments to certain quotient polynomials, we can compute in a number of operations depending only on n, the *commitment* to Q. Moreover, having only a commitment to Q suffices to check, via pairings, that T_f is valid.

This approach was a big step forward, enabling for the first time lookups sublinear in table size. However, it has the following disadvantage: "Extracting" the subtable of values used in f, is analogous to looking at restrictions of the original table polynomial to arbitrary sets - far from the nice subgroups we are used to in zk-SNARK world. Very roughly speaking, this is why all previous four works end up needing to work with interpolation and evaluation of polynomials on arbitrary sets. The corresponding algorithms for working on such sets have asymptotics of $O(n \cdot \log^2 n)$ rather than the $O(n \log n)$ we get for subgroups (of order 2^k for example).

Our approach The key difference between [ZBK$^+$22, PK22, GK22, ZGK$^+$22] and $\mathfrak{cq}$ is that we use the idea of succinct computation of quotient commitments, not to extract a subtable, *but to directly run an exisiting lookup protocol on the original large table more efficiently.* Specifically, we use as our starting point the "logarithmic derivative based lookup" of [Eag22, Hab22].

[Hab22] utilizes the following lemma (cf. Lemma 2.4 or Lemma 5 in [Hab22]): $f|_{\mathbb{H}} \subset \mathfrak{t}$ if and only if for some $m \in \mathbb{F}^N$

$$\sum_{i\in[N]} \frac{m_i}{X+\mathfrak{t}_i} = \sum_{i\in[n]} \frac{1}{X+f_i},$$

as rational functions. [Hab22] checks this identity on a random β, by sending commitments to polynomials A and B whose values correspond to the summands evaluated at β of the LHS and RHS respectively. Given commitments to A, B, we can check the above equality holds via various sumcheck techniques, e.g. as described in [BCR+19] (cf. Lemma 2.1). The RHS is not a problem because it is a sum of size n. Computing A's commitment is actually not a problem either, because the number of its non-zero values on $\mathbb{V}$ is at most n. So when precomputing the commitments to the Lagrange base of $\mathbb{V}$, we can compute A's commitment in n group operations.

The main challenge is to convince the verifier $\mathbf{V}$ that A is correctly formed. This is equivalent to the existence of a quotient polynomial $Q_A(X)$ such that

$$A(X)(T(X) + \beta) - m(X) = Q_A(X) \cdot Z_{\mathbb{V}}(X).$$

It can be seen that this is the same $Q_A(X)$ as when writing

$$A(X)T(X) = Q_A(X)Z_{\mathbb{V}}(X) + R(X),$$

for $R(X) \in \mathbb{F}_{<N}[X]$.

Here is where our central innovation, and the term "cached quotients" come from. We observe that while computing Q_A would take too long, we can compute the commitment $[Q_A(x)]_1$ to Q_A in $O(n)$ operations as follows. We precompute for each $L_i(X)$ in the Lagrange basis of $\mathbb{V}$ its quotient commitment when multiplying with $T(X)$, i.e. the commitment to $Q_i(X)$ such that for some remainder $R_i(X) \in \mathbb{F}_{<N}[X]$.

$$L_i(X)T(X) = Q_i(X) \cdot Z_{\mathbb{V}}(X) + R_i(X).$$

Given the commitments $[Q_i(x)]_1$, $[Q_A(x)]_1$ can be computed in $O(n)$ $\mathbb{G}_1$-operations via linear combination. Moreover, all the elements $[Q_i(x)]_1$ can be computed in an $O(N \log N)$ preprocessing phase leveraging the work of Feist and Khovratovich[FK]. See Section 3 for details on this.

2 Preliminaries

2.1 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 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.

The Aurora lemma Our sumcheck relies on the following lemma originally used in the Aurora construction ([BCR+19], Remark 5.6).

Lemma 2.1. *Let $H \subset \mathbb{F}$ be a multiplicative subgroup of size t. For $f \in \mathbb{F}_{<t}[X]$, we have*

$$\sum_{a \in H} f(a) = t \cdot f(0).$$

2.2 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 2.2. *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 2.3. *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.

The log-derivative method We crucially use the following lemma from [Hab22].

Lemma 2.4. *Given $f \in \mathbb{F}^n$, and $t \in \mathbb{F}^N$, we have $f \subset t$ as sets if and only if for some $m \in \mathbb{F}^N$ the following identity of rational functions holds*

$$\sum_{i \in [n]} \frac{1}{X + f_i} = \sum_{i \in [N]} \frac{m_i}{X + t_i}.$$

3 Cached quotients

Notation: In this section and the next we use the following conventions. $\mathbb{V} \subset \mathbb{F}$ denotes a mutliplicative subgroup of order N which is a power of two. We denote by $\mathbf{g}$ a generator of $\mathbb{V}$. Hence, $\mathbb{V} = \left\{\mathbf{g}, \mathbf{g}^2, \ldots, \mathbf{g}^N = 1\right\}$. Given $P \in \mathbb{F}[X]$ and integer $i \in [N]$, we denote $P_i := P(\mathbf{g}^i)$. For $i \in [N]$, we denote by $L_i \in \mathbb{F}_{<N}[X]$ the i'th Lagrange polynomial of $\mathbb{V}$. Thus, $(L_i)_i = 1$ and $(L_i)_j = 0$ for $i \neq j \in [N]$.

For a polynomial $A(X) \in \mathbb{F}_{<N}[X]$, we say it is *$n$-sparse* if $A_i \neq 0$ for at most n values $i \in [N]$. The *sparse representation* of such A consists of the (at most) n pairs (i, A_i) such that $A_i \neq 0$. We denote $\mathrm{supp}(A) := \{i \in [N] | A_i \neq 0\}$.

The main result of this section is a method to compute a commitment to a quotient polynomial - derived from a product with a preprocessed polynomial; in a number of operations depending only on the sparsity of the other polynomial in the product.

The result crucially relies on the following lemma based on a result of Feist and Khovratovich[FK].

Lemma 3.1. *Fix $T \in \mathbb{F}_{<N}[X]$, and a subgroup $\mathbb{V} \subset \mathbb{F}$ of size N. There is an algorithm that given the $\mathbb{G}_1$ elements $\left\{\left[x^i\right]_1\right\}_{i \in \{0,\ldots,N-1\}}$ computes for $i \in [N]$, the elements $q_i := [Q_i(x)]_1$ where $Q_i(X) \in \mathbb{F}[X]$ is such that*

$$L_i(X) \cdot T(X) = T_i \cdot L_i(X) + Z_{\mathbb{V}}(X) \cdot Q_i(X)$$

in $O(N \cdot \log N)$ $\mathbb{G}_1$ operations.

Proof. Recall the definition of the Lagrange polynomial

$$L_i(X) = \frac{Z_{\mathbb{V}}(X)}{Z'_{\mathbb{V}}(\mathbf{g}^i)(X - \mathbf{g}^i)}.$$

Substituting this definition, we can write the quotient $Q_i(X)$ as

$$Q_i(X) = \frac{T(X) - T_i}{Z'_{\mathbb{V}}(\mathbf{g}^i)(X - \mathbf{g}^i)} = Z'_{\mathbb{V}}(\mathbf{g}^i)^{-1} K_i(X),$$

for $K_i(X) := \frac{T(X) - T_i}{X - \mathbf{g}^i}$. Note that the values $\{[K_i(x)]_1\}_{i \in [N]}$ are exactly the KZG opening proofs of $T(X)$ at the elements of $\mathbb{V}$. Thus, the algorithm of Feist and Khovratovich [FK, Tom] can be used to compute commitments to all the proofs $[K_i(x)]_1$ in $O(N \log N)$ $\mathbb{G}_1$-operations. This works by writing the vector of $[K_i(x)]_1$ as a the product of a matrix with the vector of $\left[x^i\right]_1$. This matrix is a DFT matrix times a Toeplitz matrix, both of which have algorithms for evaluating matrix vector products in $O(N \log N)$ operations. Thus, all the KZG proofs can be computed in $O(N \log N)$ field operations and operations in $\mathbb{G}_1$.

Finally, the algorithm just needs to scale each $[K_i(x)]_1$ by $Z'_{\mathbb{V}}(\mathbf{g}^i)$ to compute $[Q_i(x)]_1$. Conveniently, these values admit a very simple description when $Z_{\mathbb{V}}(X) = X^N - 1$ is a group of roots of unity.

$$Z'_{\mathbb{V}}(X)^{-1} = (N X^{N-1})^{-1} \equiv X/N \bmod Z_{\mathbb{V}}(X)$$

In total, the prover computes the coefficients of $T(X)$ in $O(N \log N)$ field operations, computes the KZG proofs for $T(\mathbf{g}^i) = t_i$ in $O(N \log N)$ group operations, and then scales these proofs by $\mathbf{g}^i/n$ in $O(N)$ group operations. In total, this takes $O(N \log N)$ field and group operations in $\mathbb{G}_1$. $\square$

Theorem 3.2. *Fix integer parameters $0 \le n \le N$ such that n, N are powers of two. Fix $T \in \mathbb{F}_{<N}[X]$, and a subgroup $\mathbb{V} \subset \mathbb{F}$ of size N. Let* $\mathsf{srs} = \left\{ \left[x^i\right]_1 \right\}_{i\in[0,\ldots,N-1]}$ *for some $x \in \mathbb{F}$. There is an algorithm $\mathscr{A}$ that after a preprocessing step of $O(N \log N)$ $\mathbb{F}$- and $\mathbb{G}_1$-operations starting with* srs *does the following.*

Given input $A(X) \in \mathbb{F}_{<N}[X]$ that is n-sparse and given in sparse representation, $\mathscr{A}$ computes in $O(n)$ $\mathbb{F}$-operations and n $\mathbb{G}_1$-operations each of the elements $\mathsf{cm}_1 = [Q(x)]_1$, $\mathsf{cm}_2 = [R(x)]_1$ for $Q(X), R(X) \in \mathbb{F}_{<N}[X]$ such that

$$A(X) \cdot T(X) = Q(X) \cdot Z_{\mathbb{V}}(X) + R(X).$$

Proof. The preprocessing step consists of computing the quotient commitments $[Q_i(X)]_1$ in $O(N \log N)$ operations, as described in Lemma 3.1. As stated in the lemma, for each $i \in [N]$ we have

$$L_i(X) \cdot T(X) = T_i \cdot L_i(X) + Z_{\mathbb{V}}(X) \cdot Q_i(X).$$

By assumption, the polynomial $A(X)$ can be written as a linear combination of at most n summands in the Lagrange basis of $\mathbb{V}$.

$$A(X) = \sum_{i\in\mathrm{supp}(A)} A_i \cdot L_i(X)$$

Substituting this into the product with $T(X)$, and substituting each of the products $L_i(X)T(X)$ with the appropriate cached quotient $Q_i(X)$ we find

$$\begin{aligned} A(X)T(X) &= \sum_{i\in\mathrm{supp}(A)} A_i \cdot L_i(X)T(X) = \sum_{i\in\mathrm{supp}(A)} A_i \cdot T_i L_i(X) + A_i \cdot Z_{\mathbb{V}}(X) Q_i(X) \\ &= \sum_{i\in\mathrm{supp}(A)} A_i \cdot T_i L_i(X) + Z_{\mathbb{V}}(X) \cdot \sum_{i\in\mathrm{supp}(A)} A_i \cdot Q_i(X). \end{aligned}$$

Observing that the terms of the first sum are all of degree smaller than N, we get that

$$Q(X) = \sum_{i\in\mathrm{supp}(A)} A_i \cdot Q_i(X)$$
$$R(X) = \sum_{i\in\mathrm{supp}(A)} A_i T_i \cdot L_i(X)$$

Hence, commitments to both the quotient $Q(X)$ and remainder $R(X)$ can be computed in at most n group operations as

$$[Q(x)]_1 = \sum_{i\in\mathrm{supp}(A)} A_i \cdot [Q_i(x)]_1$$
$$[R(x)]_1 = \sum_{i\in\mathrm{supp}(A)} A_i T_i \cdot [L_i(x)]_1$$

$\square$

4 $\mathfrak{cq}$ - our main protocol

Before describing our protocol, we give a definition of a lookup protocol secure against algebraic adversaries.

Definition 4.1. *A* lookup protocol *is a pair* $\mathscr{P} = (\mathsf{gen}, \mathsf{IsInTable})$ *such that*

- $\mathsf{gen}(N, \mathsf{t})$ *is a randomized algorithm receiving as input parameters integer* N *and* $\mathsf{t} \in \mathbb{F}^N$*. Given these inputs* gen *outputs a string* srs *of* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *elements.*
- $\mathsf{IsInTable}(\mathsf{cm}, \mathsf{t}, \mathsf{srs}, \mathbb{H}; f)$ *is an interactive public coin protocol between* $\mathbf{P}$ *and* $\mathbf{V}$ *where* $\mathbf{P}$ *has private input* $f \in \mathbb{F}_{<n}[X]$*, and both parties have access to* t, cm *and* $\mathsf{srs} = \mathsf{gen}(N, \mathsf{t})$*; such that*
  - ***Completeness:*** *If* $\mathsf{cm} = [f(x)]_1$ *and* $f|_{\mathbb{H}} \subset \mathsf{t}$ *then* $\mathbf{V}$ *outputs* acc *with probability one.*
  - ***Knowledge soundness in the algebraic group model:*** *The probability of any efficient algebraic* $\mathcal{A}$ *to win the following game is* $\mathsf{negl}(\lambda)$*.*
    1. *$\mathcal{A}$ chooses integer parameters N, n and a table $\mathsf{t} \in \mathbb{F}^N$.*
    2. *We compute $\mathsf{srs} = \mathsf{gen}(\mathsf{t}, N)$.*
    3. *$\mathcal{A}$ sends a message cm and $f \in \mathbb{F}_{<d}[X]$ such that $\mathsf{cm} = [f(x)]_1$ where d is such that all $\mathbb{G}_1$ elements in srs 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 $\mathsf{IsInTable}(\mathsf{t}, \mathsf{cm}, \mathsf{srs}, \mathbb{H})$, where $\mathbb{H} \subset \mathbb{F}$ is a subgroup of order n, with $\mathcal{A}$ taking the role of $\mathbf{P}$.*
    5. *$\mathcal{A}$ wins if*
       * $\mathbf{V}$ *outputs* acc*, and*
       * $f|_{\mathbb{H}} \not\subset \mathsf{t}$.

We say a lookup protocol is homomorphic *if for any fixed parameter* N *and fixed randomness,* $\mathsf{gen}(\mathsf{t}, N)$ *can be written as* $(\mathsf{gen}_1, \mathsf{gen}_2(\mathsf{t}))$ *such that* gen_1 *is fixed, and* gen_2 *is an* $\mathbb{F}$*-linear function of* t*.*

4.1 The $\mathfrak{cq}$ protocol

$\mathsf{gen}(N, \mathsf{t})$:

1. Choose random $x \in \mathbb{F}$ compute and output $\left\{[x^i]_1\right\}_{i\in\{0,\ldots,N-1\}}, \left\{[x^i]_2\right\}_{i\in\{0,\ldots,N\}}$.
2. Compute and output $[Z_{\mathbb{V}}(x)]_2$.
3. Compute $T(X) = \sum_{i\in[N]} \mathsf{t}_i L_i(X)$. Compute and output $[T(x)]_2$.
4. For $i \in [N]$, compute and output:

(a) $q_i = [Q_i(x)]_1$ such that

$$L_i(X) \cdot T(X) = \mathsf{t}_i \cdot L_i(X) + Z_{\mathbb{V}}(X) \cdot Q_i(X).$$

(b) $[L_i(x)]_1$.

(c) $\left[\frac{L_i(x)-L_i(0)}{x}\right]_1$.

Before describing $\mathsf{IsInTable}$, we explain an optimization we use in Step 6 of Round 2. Since we know in advance we are going to open B at zero, it is more efficient to commit to the *the opening proof polynomial* $B_0(X) := \frac{B(X)-B(0)}{X}$ *of* B *at 0 instead of committing to* B. To evaluate B, $\mathbf{V}$ can use the relation $B(X) = B_0(X) \cdot X + b_0$.

We note that it's possible to make a similar optimization for A to further reduce proof size and prover time. However, this entails an additional verifier pairing for the check in Step 11 of Round 2.

$\underline{\mathsf{IsInTable}(\mathsf{cm}, \mathsf{t}, \mathsf{srs}, \mathbb{H}; f)}$:

Round 1: Committing to the multiplicities vector

1. $\mathbf{P}$ computes the polynomial $m(X) \in \mathbb{F}_{<N}[X]$ defined by setting m_i, for each $i \in [N]$, to the number of times t_i appears in $f|_{\mathbb{H}}$.
2. $\mathbf{P}$ sends $\mathsf{m} := [m(x)]_1$.

Round 2: Interpolating the rational identity at a random β; checking correctness of A's values + degree check for B using pairings

1. $\mathbf{V}$ chooses and sends random $\beta \in \mathbb{F}$.
2. $\mathbf{P}$ computes $A \in \mathbb{F}_{<N}[X]$ such that for $i \in [N]$, $A_i = m_i/(\mathsf{t}_i + \beta)$.
3. $\mathbf{P}$ computes and sends $\mathsf{a} := [A(x)]_1$.
4. $\mathbf{P}$ computes and sends $\mathsf{q_a} := [Q_A(x)]_1$ where $Q_A \in \mathbb{F}_{<N}[X]$ is such that
$$A(X)(T(X) + \beta) - m(X) = Q_A(X) \cdot Z_{\mathbb{V}}(X)$$
5. $\mathbf{P}$ computes $B(X) \in \mathbb{F}_{<n}[X]$ such that for $i \in [n]$, $B_i = 1/(f_i + \beta)$.
6. $\mathbf{P}$ computes $B_0(X) \in \mathbb{F}_{<n-1}[X]$ defined as $B_0(X) := \frac{B(X)-B(0)}{X}$.
7. $\mathbf{P}$ computes and sends $\mathsf{b}_0 := [B_0(x)]_1$.

8. $\mathbf{P}$ computes $Q_B(X)$ such that
$$B(X)(f(x)+\beta)-1=Q_B(X)\cdot Z_{\mathbb{H}}(X).$$

9. $\mathbf{P}$ computes and sends $\mathsf{q_b} := [Q_B(x)]_1$.

10. $\mathbf{P}$ computes and sends $\mathsf{p} = [P(x)]_1$ where
$$P(X) := B_0(X)\cdot X^{N-1-(n-2)}.$$

11. $\mathbf{V}$ checks that A encodes the correct values:
$$e(\mathsf{a},[T(x)]_2)=e(\mathsf{q_a},[Z_{\mathbb{V}}(x)]_2)\cdot e(\mathsf{m}-\beta\cdot\mathsf{a},[1]_2)$$

12. $\mathbf{V}$ checks that B_0 has the appropriate degree:
$$e\left(\mathsf{b}_0,\left[x^{N-1-(n-2)}\right]_2\right)=e(\mathsf{p},[1]_2).$$

Round 3: Checking correctness of B at random $\gamma\in\mathbb{F}$

1. $\mathbf{V}$ sends random $\gamma,\eta\in\mathbb{F}$.
2. $\mathbf{P}$ sends $b_{0,\gamma} := B_0(\gamma), f_\gamma := f(\gamma)$.
3. $\mathbf{P}$ computes and sends the value $a_0 := A(0)$.
4. $\mathbf{V}$ sets $b_0 := (N\cdot a_0)/n$.
5. As part of checking the correctness of B, $\mathbf{V}$ computes $Z_{\mathbb{H}}(\gamma)=\gamma^n-1$, $b_\gamma := b_{0,\gamma}\cdot\gamma+b_0$ and
$$Q_{b,\gamma} := \frac{b_\gamma\cdot(f_\gamma+\beta)-1}{Z_{\mathbb{H}}(\gamma)}.$$
6. To perform a batched KZG check for the correctness of the values $b_{0,\gamma}, f_\gamma, Q_{b,\gamma}$
   (a) $\mathbf{V}$ sends random $\eta\in\mathbb{F}$. $\mathbf{P}$ and $\mathbf{V}$ separately compute
   $$v := b_{0,\gamma}+\eta\cdot f_\gamma+\eta^2\cdot Q_{b,\gamma}.$$
   (b) $\mathbf{P}$ computes $\pi_\gamma := [h(x)]_1$ for
   $$h(X) := \frac{B_0(X)+\eta\cdot f(X)+\eta^2\cdot Q_B(X)-v}{X-\gamma}$$
   (c) $\mathbf{V}$ computes
   $$\mathsf{c} := \mathsf{b}_0+\eta\cdot\mathsf{cm}+\eta^2\cdot\mathsf{q_b}$$
   and checks that
   $$e(\mathsf{c}-[v]_1+\gamma\cdot\pi_\gamma,[1]_2)=e(\pi_\gamma,[x]_2).$$

7. To perform a KZG check for the correctness of a_0

   (a) **P** computes and sends $\mathsf{a}_0 := [A_0(x)]_1$ for

$$A_0(X) := \frac{A(X) - a_0}{X}$$

   (b) **V** checks that

$$e(\mathsf{a} - [a_0]_1, [1]_2) = e(\mathsf{a}_0, [x]_2).$$

Note that although the above description contains nine pairings, we can reduce to five pairings via the standard technique of randomly batching pairings that share the same $\mathbb{G}_2$ argument. It is easy to check that $\mathfrak{cq}$ is homomorphic according to Definition 4.1.

The main things to address are the efficiency of the gen algorithm used for preprocessing, the efficiency of **P** in $\mathsf{IsInTable}$, and the knowledge soundness of $\mathsf{IsInTable}$.

$\mathsf{Runtime\ of\ gen}$: We claim that gen requires $O(N \log N)$ $\mathbb{G}_1$- and $\mathbb{F}$-operations and $O(N)$ $\mathbb{G}_2$-operations. The claim regarding the $\mathbb{G}_2$ operations is obvious. The elements $\{q_i\}$ can be computing in $O(N \log N)$ operations according to Lemma 3.1. The elements $\{[L_i(x)]_1\}$ can be computed in $O(N \log N)$ via FFT as explained in Section 3.3 of [BGG17]. Given the element $[L_i(x)]_1$, the element $\left[\frac{L_i(x)-L_i(0)}{x}\right]_1$ can be computed as

$$\left[\frac{L_i(x) - L_i(0)}{x}\right]_1 = \mathbf{g}^{-i} \cdot [L_i(x)]_1 - (1/N) \cdot \left[x^{N-1}\right]_1.$$

$\mathsf{Runtime\ of}$ $\mathbf{P}$: Note first that the computation of m, a can be done in n $\mathbb{G}_1$-operations as $m(X)$and $A(X)$ are n-sparse. The main thing to address is the computation of $\mathsf{q_a}$; that can be done in n $\mathbb{G}_1$-operations given srs according to Theorem 3.2. The only step requiring $O(n \log n)$ $\mathbb{F}$-operations is the computation the quotient $Q_B(X)$ which involves FFT on $\mathbb{H}$. We also note that the commitment $\mathsf{a}_0 = \left[\frac{A(x)-A(0)}{x}\right]_1$ can be computed in n $\mathbb{G}_1$-operations as the linear combination

$$\left[\frac{A(x) - A(0)}{x}\right]_1 = \sum_{i \in \mathrm{supp}(A)} A_i \cdot \left[\frac{L_i(x) - L_i(0)}{x}\right]_1.$$

$\mathsf{Knowledge\ soundness\ proof}$: Let $\mathcal{A}$ be an efficient algebraic adversary participating in the Knowledge Soundness game from Definition 4.1. We show its probability of winning the game is $\mathsf{negl}(\lambda)$. Let $f \in \mathbb{F}_{<N}[X]$ be the polynomial sent by $\mathcal{A}$ in the third step of the game such that $\mathsf{cm} = [f(x)]_1$. As $\mathcal{A}$ is algebraic, when sending the commitments m,a,b_0,p,$\mathsf{q_a}$,$\mathsf{q_b}$,π_γ,a_0 during protocol execution it also sends polynomials $m(X), A(X), B_0(X),$ $P(X), Q_A(X), Q_B(X), h(X), A_0(X) \in \mathbb{F}_{<N}[X]$ such that the former are their corresponding commitments. Let E be the event that **V** outputs acc. Note that the event that $\mathcal{A}$ wins the knowledge soundness game is contained in E. E implies

all pairing checks have passed. Let $A \subset E$ be the event that one of the corresponding ideal pairing checks as defined in Section 2.2 didn't pass. According to Lemma 2.3, $\Pr(A = \mathsf{negl}(\lambda))$. Given that A didn't occur, we have

- From Round 2, Step 11
$$A(X)(T(X)+\beta) - M(X) = Q_A(X) \cdot Z_{\mathbb{V}}(X)$$
Which means that for all $i \in [N]$,
$$A_i = \frac{M_i}{T_i + \beta}$$
- From Round 2, Step 12
$$X^{N-1-(n-2)} B_0(X) = P(X),$$
which implies that $\deg(B_0) \leq n-2$. Note also that we know $\deg(A) < N$ simply from $\left[x^{N-1}\right]_1$ being the highest $\mathbb{G}_1$ power in srs.[2]
- Moving to Round 3, from the checks of steps 6c and 7b, e.w.p. $n/|\mathbb{F}|$ over $\eta, \zeta \in \mathbb{F}$ (see e.g. Section 3 of [GWC19] for an explanation of batched KZG [KZG10]), we have $b_{0,\gamma} = B_0(\gamma), Q_{b,\gamma} = Q_B(\gamma), f_\gamma = f(\gamma), a_0 = A(0)$.
- Define $B(X) := B_0(X) \cdot X + b_0$ for b_0 set as in step 4. Note that we have $\deg(B) < n$. Let ω by a generator of $\mathbb{H}$.
- By how $b_\gamma, Q_{b,\gamma}$ are set in step 5, the above implies that e.w.p. $(N+n)/|\mathbb{F}|$ over γ
$$B(X) \cdot (f(X) + \beta) = 1 + Q_B(X) Z_{\mathbb{H}}(X),$$
which implies for all $i \in [n]$ that $B(\omega^i) = \frac{1}{f(\omega^i)+\beta}$.
- We now have using Lemma 2.1 that
$$N \cdot a_0 = \sum_{i \in [N]} A_i = \sum_{i \in [N]} \frac{m_i}{T_i + \beta},$$
$$n \cdot b_0 = \sum_{i \in [n]} B(\omega^i) = \sum_{i \in [n]} \frac{1}{f(\omega^i) + \beta}.$$
Recall that b_0 was set such that $N \cdot a_0 = n \cdot b_0$. Thus e.w.p. $(n \cdot N)/|\mathbb{F}|$ over $\beta \in \mathbb{F}$, we have that
$$\sum_{i \in [N]} \frac{m_i}{T_i + X} = \sum_{i \in [n]} \frac{1}{f(\omega^i) + X},$$
which implies $f|_{\mathbb{H}} \subset \mathfrak{t}$ by Lemma 2.4.

In summary, we have shown the event that $\mathbf{V}$ outputs acc while $f|_{\mathbb{H}} \not\subset \mathfrak{t}$ is contained in a constant number of events with probability $\mathsf{negl}(\lambda)$; and so $\mathfrak{cq}$ satisfies the knowledge soundness property.

[2] An important point is that when using an SRS built with higher degrees in $\mathbb{G}_1$, A must also be degree checked via an additional pairing. In such a case, we must also change the power of X in Step 12 of Round 2 from $N-1-(n-2)$ to $d-(n-2)$ where d is the maximal SRS degree in $\mathbb{G}_1$.

Acknowledgements

We thank Kobi Gurkan and Aurel Nicolas for corrections.

References

[BCG+18] J. Bootle, A. Cerulli, J. Groth, S. K. Jakobsen, and M. Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*, volume 11272 of *Lecture Notes in Computer Science*, pages 595–626. Springer, 2018.

[BCMS20] B. Bünz, A. Chiesa, P. Mishra, and N. Spooner. Recursive proof composition from accumulation schemes. In *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2020.

[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.

[BGG17] S. Bowe, A. Gabizon, and M. D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-snark. *IACR Cryptology ePrint Archive*, 2017:602, 2017.

[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`.

[Eag22] Liam Eagen. Bulletproofs++. *IACR Cryptol. ePrint Arch.*, page 510, 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.

[GK22] A. Gabizon and D. Khovratovich. flookup: Fractional decomposition-based lookups in quasi-linear time independent of table size. *IACR Cryptol. ePrint Arch.*, page 1447, 2022.

[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.

[GW20] A. Gabizon and Z. J. Williamson. plookup: A simplified polynomial protocol for lookup tables. *IACR Cryptol. ePrint Arch.*, page 315, 2020.

[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.

[Hab22] U. Haböck. Multivariate lookups based on logarithmic derivatives. *IACR Cryptol. ePrint Arch.*, page 1530, 2022.

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

[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.