

# **cq**\* Cached quotients for fast lookups

Liam Eagen  
Blockstream

Dario Fiore  
IMDEA software institute

Ariel Gabizon  
Zeta Function Technologies

December 24, 2022

## 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  $\mathbf{t} \in \mathbb{F}^N$ . 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<sup>+</sup>22, PK22, GK22, ZGK<sup>+</sup>22] starting from Caulk [ZBK<sup>+</sup>22], which achieve sublinear complexity in the table size  $N$ . The two most recent works in this sequence [GK22, ZGK<sup>+</sup>22] achieved prover complexity  $O(n \cdot \log^2 n)$ .

Moreover, **cq** has the following attractive features.

1. As in [ZBK<sup>+</sup>22, PK22, ZGK<sup>+</sup>22] our construction relies on homomorphic table commitments, which makes them amenable to vector lookups in the manner described in Section 4 of [GW20].
2. As opposed to [ZBK<sup>+</sup>22, PK22, GK22, ZGK<sup>+</sup>22] the **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  $\phi(X) \in \mathbb{F}_{<n}[X]$  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<sup>+</sup>18], **plookup** [GW20] was the first to explicitly describe a solution to this problem in the polynomial-IOP context. **plookup** described a protocol with prover complexity quasilinear in both  $n$  and  $N$ . This

---

\*Pronounced “seek you”.

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

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 - baloo [ZGK<sup>+</sup>22] and **Flookup** [GK22] achieving a runtime of  $O(n \log^2 n)$ . While **Flookup** has better concrete constants, baloo preserved an attractive feature of Caulk - using a *homomorphic commitment* to the table. In the sense that given commitments  $\text{cm}_1, \text{cm}_2$  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  $\text{cm} := \text{cm}_1 + \alpha \cdot \text{cm}_2$  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 for recent constructions - Caulk, Caulk<sup>+</sup>, baloo, **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 recursively aggregate multiple proofs via random combination, in the style described e.g. in Section 8 of [?].

## 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  $O(n \log n)$ .
2. Achieves small constants in proof size and group operations, *while preserving homomorphicity* as in Caulk, Caulk<sup>+</sup> and baloo.
3. Achieves for the first time in this line of work convenient aggregatability by having all verifier contain fixed protocol defined  $\mathbb{G}_2$  arguments.

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

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

<sup>1</sup>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<sup>+</sup>22].

**The innovation of Caulk** To restate the problem, we have an input polynomial  $f(X)$ , a table  $\mathbf{t}$  of size  $N$  encoded in the values of a polynomial  $T(X) \in \mathbb{F}_{<N}[X]$ , and wish to show  $f|_{\mathbb{H}} \subset \mathbf{t}$ , where  $|\mathbb{H}| = n \ll N$ . 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  $T'$  encoding the  $n$  values from  $\mathbf{t}$  *actually used in  $f$* , and then run a lookup protocol using  $T'$ . The challenging problem is to prove  $T'$  *actually encodes values from  $T$* . Speaking imprecisely, the “witness” to  $T'$  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<sup>+</sup>22] is the following observation: If we precompute commitments to certain 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  $T'$  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  $f$ , necessitates interpolation and evaluation of polynomials on arbitrary sets, rather than just subgroups. 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 are used to from FFT’s on subgroups.

Moreover, when desiring to work with a *homomorphic* table commitment, the Caulk approach requires working with a  $T'$  that encodes the subtable values on a “hidden”, prover-defined set of inputs; consequently requiring more ingenuity and complexity to use the subtable, also after it has been extracted.

**Our approach** The key difference between **ca** and [ZBK<sup>+</sup>22, PK22, GK22, ZGK<sup>+</sup>22] is that we use the idea of succinct computation of quotient commitments, not to extract a subtable, but to directly run an existing lookup protocol on the original large table, with a modified prover performing only  $O(n \log n)$  operations.

Specifically, we use as our starting point the “logarithmic derivative based lookup” of [Hab22].

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

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

as rational functions. [Hab22] checks this identity on a random  $\beta$ , by sending commitment to polynomials  $A$  and  $B$  whose values correspond to the summands of the LHS and RHS respectively. Given commitments to  $A, B$ , we can check the above equality holds via various sumcheck techniques, e.g. that described in [BCR<sup>+</sup>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 we precompute the commitments to the Lagrange base of  $\mathbb{V}$ , we can compute  $A$ 's commitment in  $n$  group operations.

The main challenge, and innovation, 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

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

This protocol is amenable, because polynomials involved have sparsity depending on witness - For large table problem is computing  $A$  that agrees with  $m/(t + \beta)$  on  $\mathbb{V}$

- Need way to compute  $A$

## 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  $\lambda$ .

Whenever we use the term *efficient*, we mean an algorithm running in time  $\text{poly}(\lambda)$ . Furthermore, we assume an *object generator*  $\mathcal{O}$  that is run with input  $\lambda$  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  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$ .
- $g_1, g_2$  are uniformly chosen generators such that  $e(g_1, g_2) = g_t$ .

We usually let the  $\lambda$  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  $\{1, \dots, n\}$ . We use the acronym e.w.p for “except with probability”; i.e. e.w.p  $\gamma$  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  $\text{poly}(\lambda)$ -time from an “SRS of monomials” of the form  $\{[x^i]_1\}_{a \leq i \leq b}, \{[x^i]_2\}_{c \leq i \leq d}$ , for uniform  $x \in \mathbb{F}$ , and some integers  $a, b, c, d$  with absolute

value bounded by  $\text{poly}(\lambda)$ . 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  $\text{negl}(\lambda)$  advantage in its probability of producing a proof of any statement.

For notational simplicity, we sometimes use the SRS  $\text{srs}$  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<sup>+</sup>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*

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

## 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  $\text{poly}(\lambda)$ -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  $A = \langle v, \text{srs}_i \rangle$ .

First we say our  $\text{srs}$  has *degree*  $Q$  if all elements of  $\text{srs}_i$  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  $\text{srs}_i$ .

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  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$ .

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

$$[1]_1, [x]_1, \dots, [x^Q]_1, [1]_2, [x]_2, \dots, [x^Q]_2$$

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

**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  $\text{negl}(\lambda)$  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 multiplicative subgroup of order  $N$  which is a power of two. We denote by  $\mathbf{g}$  a generator of  $\mathbb{V}$ . Hence,  $\mathbb{V} = \{\mathbf{g}, \mathbf{g}^2, \dots, \mathbf{g}^N = 1\}$ . 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  $\text{supp}(A) := \{i \in [N] \mid A_i \neq 0\}$ .

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  $\{[x^i]_1\}_{i \in \{0, \dots, N\}}$  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

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

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 the product of a matrix with the vector of  $[X^i]_1$ . 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  $Z'_{\mathbb{V}}(\omega^i)$  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.

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

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

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.

**Theorem 3.2.** *Fix integer parameters  $0 \leq n \leq N$  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  $\mathbf{srs} = \{[x^i]_1\}_{i \in [0, \dots, N]}$  for some  $x \in \mathbb{F}$ . There is an algorithm  $\mathcal{A}$  that after a preprocessing step of  $O(N \log N)$   $\mathbb{F}$ - and  $\mathbb{G}_1$ -operations starting with  $\mathbf{srs}$  does the following.*

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

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

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

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

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

$$\begin{aligned} Q(X) &= \sum_{i \in \text{supp}(A)} A_i \cdot Q_i(X) \\ R(X) &= \sum_{i \in \text{supp}(A)} A_i T_i \cdot L_i(X) \end{aligned}$$

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

$$\begin{aligned} [Q(X)]_1 &= \sum_{i \in \text{supp}(A)} A_i \cdot [Q_i(X)]_1 \\ [R(X)]_1 &= \sum_{i \in \text{supp}(A)} A_i T_i \cdot [L_i(X)]_1 \end{aligned}$$

□

## 4 $\mathfrak{c}\mathfrak{q}$ - 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  $\mathcal{P} = (\text{gen}, \text{lsInTable})$  such that

- $\text{gen}(N, \mathfrak{t})$  is a randomized algorithm receiving as input parameters  $N, \mathfrak{t} \in \mathbb{F}^N$ . Given these inputs  $\text{gen}$  outputs a string  $\text{srs}$ .
- $\text{lsInTable}(\text{cm}, \mathfrak{t}, \text{srs}, \mathbb{H}; f)$  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  $\mathfrak{t}, \text{cm}$  and  $\text{srs} = \text{gen}(N, \mathfrak{t})$ . such that



- **Completeness:** If  $\text{cm} = [f(x)]_1$  and  $f|_{\mathbb{H}} \subset \mathbf{t}$  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  $\text{negl}(\lambda)$ .
  1.  $\mathcal{A}$  chooses integer parameters  $N, n$  and a table  $\mathbf{t} \in \mathbb{F}^N$ .
  2. We compute  $\text{srs} = \text{gen}(\mathbf{t}, N, n)$ .
  3.  $\mathcal{A}$  sends a message  $\text{cm}$  and  $f \in \mathbb{F}_{<N}[X]$  such that  $\text{cm} = [f(x)]_1$ .
  4.  $\mathcal{A}$  and  $\mathbf{V}$  engage in the protocol  $\text{IsInTable}(\mathbf{t}, \text{cm}, \text{srs}, \mathbb{H})$  with  $\mathcal{A}$  taking the role of  $\mathbf{P}$ .
  5.  $\mathcal{A}$  wins if
    - \*  $\mathbf{V}$  outputs acc, and
    - \*  $f|_{\mathbb{H}} \not\subset \mathbf{t}$ .

#### 4.1 The $\mathbf{eq}$ protocol

$\text{gen}(N, \mathbf{t})$ :

1. Choose random  $x \in \mathbb{F}$  compute and outputs  $\{[x^i]_1\}_{i \in \{0, \dots, N-1\}}, \{[x^i]_2\}_{i \in \{0, \dots, N\}}$
2. Compute and output  $[Z_V(x)]_2$ .
3. Compute  $T(X) = \sum_{i \in [N]} \mathbf{t}_i L_i(X)$ . Compute and output  $[T(x)]_2$ .
4. Compute for  $i \in [N]$  compute and output:

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

$$L_i(X) \cdot T(X) = \mathbf{t}_i \cdot L_i(X) + Z_V(X) \cdot Q_i(X).$$

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

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

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

**Round 1:** Committing to the multiplicities vector

1.  $\mathbf{P}$  computes poly  $m \in \mathbb{F}_{<N}[X]$  such that  $m_i =$  number of times  $\mathbf{t}_i$  appears in  $f|_{\mathbb{H}}$ .
2.  $\mathbf{P}$  sends  $\mathbf{m} := [m(x)]_1$ .

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

1.  $\mathbf{V}$  chooses and sends random  $\alpha, \beta \in \mathbb{F}$ .
2.  $\mathbf{P}$  computes  $A \in \mathbb{F}_{<N}[X]$  such that for  $i \in [N]$ ,  $A_i = m_i/(\mathfrak{t}_i + \beta)$ .
3.  $\mathbf{P}$  sends  $\mathbf{a} := [A(x)]_1$ .
4.  $\mathbf{P}$  computes  $\mathbf{q_a} := [Q_A(x)]_2$  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  $B \in \mathbb{F}_{<n}[X]$  such that for  $i \in [n]$ ,  $B_i = 1/(f_i + \beta)$ .
6.  $\mathbf{P}$  sends  $\mathbf{q_b} := [B(x)]_1$ .
7.  $\mathbf{P}$  computes  $Q_B(X)$  such that

$$B(X)(f(x) + \beta) - 1 = Q_B(X) \cdot Z_{\mathbb{H}}(X)$$

8.  $\mathbf{P}$  computes and sends the value  $a_0 := A(0)$ .
9.  $\mathbf{V}$  sets  $b_0 := (N \cdot a_0)/n$ .
10.  $\mathbf{P}$  computes and sends  $\mathbf{p} = [P(x)]_1$  where

$$P(X) := B(X) \cdot X^{N-n}.$$

11.  $\mathbf{V}$  checks that  $A$  encodes the correct values:

$$e(\mathbf{a}, [T(x)]_2 + [\beta]_2) = e(\mathbf{q_a}, [Z_{\mathbb{V}}(x)]_2) \cdot e(\mathbf{m}, [1]_2)$$

12.  $\mathbf{V}$  checks that  $B$  have the appropriate degrees:

$$e(\mathbf{b}, [x^{N-n}]_2) = e(\mathbf{p}, [1]_2).$$

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

1.  $\mathbf{V}$  sends random  $\gamma, \eta, \zeta \in \mathbb{F}$ .
2.  $\mathbf{P}$  sends  $b_\gamma := B(\gamma), f_\gamma := f(\gamma)$ .
3. As part of checking the correctness of  $B$ ,  $\mathbf{V}$  computes  $Z_{\mathbb{H}}(\gamma) = \gamma^n - 1$  and

$$Q_{b,\gamma} := \frac{b_\gamma \cdot (f_\gamma + \beta) - 1}{Z_{\mathbb{H}}(\gamma)}.$$

4. To perform a batched KZG check for the correctness of the values  $a_\gamma, b_\gamma, f_\gamma$

(a)  $\mathbf{V}$  sends random  $\eta \in \mathbb{F}$ .  $\mathbf{P}$  and  $\mathbf{V}$  separately compute

$$v := b_\gamma + \eta \cdot f_\gamma + \eta^2 \cdot Q_{b,\gamma}.$$

(b)  $\mathbf{P}$  computes  $\pi_\gamma := [h(x)]_1$  for

$$h(X) := \frac{B(X) + \eta \cdot f(X) + \eta^2 \cdot Q_B(X) - v}{X - \gamma}$$

(c)  $\mathbf{V}$  computes

$$\mathbf{c} := \mathbf{b} + \eta \cdot \mathbf{f} + \eta^2 \cdot \mathbf{q}_b$$

and checks that

$$e(\mathbf{c} - [v]_1 + \gamma \cdot \pi_\gamma, [1]_2) = e(\pi_\gamma, [x]_2)$$

5. To perform a batched KZG check for the correctness of the values  $a_0, b_0$

(a)  $\mathbf{P}$  and  $\mathbf{V}$  separately compute

$$u := a_0 + \zeta \cdot b_0.$$

(b)  $\mathbf{P}$  computes and sends  $\pi_0 := [h_0(x)]_1$  for

$$h_0(X) := \frac{A(X) + \zeta \cdot B(X)}{X}$$

(c)  $\mathbf{V}$  computes

$$\mathbf{c}_0 := \mathbf{a} + \zeta \mathbf{b}$$

and checks that

$$e(\mathbf{c}_0 - [u]_1, [1]_2) = e(\pi_0, [x]_2)$$

**Stats:** verifier pairings: 5 - pair  $\mathbf{a}$  with random combination of  $T$  and  $[x^{d-N}]_2$ , pair  $\mathbf{q}_a$  with  $Z_{\mathbb{V}}$ . pair  $\mathbf{b}$  with  $[d-n]_2$  for degree check. Proof size - 8  $\mathbb{G}_1$ -  $\mathbf{a}, \mathbf{b}, \mathbf{p}, \mathbf{m}, \mathbf{q}_a, \mathbf{q}_b$   $\pi_\gamma, \pi_0$  4  $\mathbb{F}$ -  $b_\gamma, Q_{b,\gamma}, f_\gamma, a_0$

Note that we can split  $\mathbf{p}$  to two proofs and that reduces a verifier pairing

**Lemma 4.2.** *The element  $q_A$  in Step 4 can be computed in  $n \log n$   $\mathbb{G}_2$ -operations and  $O(n \log n)$   $\mathbb{F}$ -operations*

**Lemma 4.3.** *The elements  $\pi_0, \pi_\gamma$  can be computed in  $2 \cdot n \log n$   $\mathbb{G}_1$ -operations and  $O(n \log n)$   $\mathbb{F}$ -operations*

**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  $\text{negl}(\lambda)$ . Let  $f \in \mathbb{F}_{<d}[X]$  be the polynomial sent by  $\mathcal{A}$  in the first step of the game such that  $\text{cm} = [f(x)]_1$ . As  $\mathcal{A}$  is algebraic, when sending the commitments  $\text{m}, \text{a}, \text{b}, \text{p}, \text{q}_a, \text{q}_b, \pi_\gamma, \pi_0$  during protocol execution it also sends polynomials  $m(X), A(X), B(X), P(X), Q_A(X), Q_B(X), h(X), h_0(X) \in \mathbb{F}_{<d}[X]$  such that the former are their corresponding commitments. Let  $E$  be the event  $\mathbf{V}$  outputs **acc**. Note that the event that  $\mathcal{A}$  wins the game is contained in  $E$ .  $E$  implies all pairing checks passed. Let  $A \subset E$  be the event that one of the corresponding ideal pairing checks didn't pass. According to Lemma 2.3,  $\Pr(A = \text{negl}(\lambda))$ . Given  $A$  didn't occur, we have

- From 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 step 12

$$X^{N-n}B(X) = P(X),$$

which implies that  $\deg(B) < n$ . Note also that we know  $\deg(A) < N$  simply from  $[x^{N-1}]_1$  being the highest  $\mathbb{G}_1$  power in **srs**.<sup>2</sup>

- From the checks of steps 4c and 5c, e.w.p.  $n/|\mathbb{F}|$  over  $\eta, \zeta \in \mathbb{F}$  (see e.g. Section 3 of [GWC19] for an expalantion of the correctness of batched KZG [KZG10]).  $b_\gamma = B(\gamma), Q_{b,\gamma} = Q_B(\gamma), f_\gamma = f(\gamma), a_0 = A(0), b_0 = B(0)$ .
- Which implies by how  $Q_{b,\gamma}$  is set in step 3 that e.w.p.  $(N+n)/|\mathbb{F}|$  over  $\gamma$

$$B(X) \cdot (f(X) + \beta) = 1 + Q_B(X)Z_{\mathbb{H}}(X),$$

which implies for all  $i \in [n]$  that  $B(\omega^i) = \frac{1}{f(\omega^i) + \beta}$ .

- We know have using Lemma 2.1 that

$$N \cdot a_0 = \sum_{i \in [N]} A_i = \sum_{i \in [N]} \frac{m_i}{T_i + \beta}$$

$$n \cdot b_0 = \sum_{i \in [n]} B(\omega^i) = \sum_{i \in [n]} \frac{1}{f(\omega^i) + \beta}$$

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  $f|_{\mathbb{H}} \subset \mathfrak{t}$ .

---

<sup>2</sup>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 summary, we have shown the event that  $\mathbf{V}$  outputs `acc` while  $f|_{\mathbb{H}} \not\in \mathbf{t}$  is contained in a constant number of events with probability  $\text{negl}(\lambda)$ ; and so  $\mathbf{c}\mathbf{q}$  satisfies the knowledge soundness property.

## References

- [BCG<sup>+</sup>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.
- [BCR<sup>+</sup>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>.
- [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<sup>+</sup>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<sup>+</sup>22] A. Zapico, A. Gabizon, D. Khovratovich, M. Maller, and C. Ràfols. Baloo: Nearly optimal lookup arguments. *IACR Cryptol. ePrint Arch.*, page 1565, 2022.