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

Liam Eagen Dario Fiore Ariel Gabizon
IMDEA Zeta Function Technologies

December 21, 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 $\mathsf{t} \in \mathbb{F}^N$. 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, ?, ?] starting from Caulk [ZBK+22], which achieve sublinear complexity in the table size N. The two most recent works in this seqence [?, ?] achieved prover complexity $O(n \cdot \log^2 n)$.

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

1. As in [ZBK+22, PK22, ?] 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+22, PK22, ?, ?] the $\mathfrak{cq}$ verifier doesn't involve pairings with prover defined $\mathbb{G}_2$ points, which makes recursive aggregation of proofs more convenient.
3. The construction can be altered to a version we call $\mathfrak{cq}^*$ so that we lose the mentioned aggregability, increase preprocessing time to $O(n \cdot N)$, and in return reduce prover complexity to a *linear* number of field and group operations!

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+18], $\mathfrak{plookup}$ [GW20] was the first to explicitly describe a solution to this problem in the polynomial-IOP context.

*Pronounced "seek you".

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* 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.

However, the quadratic dependence on n of these works makes them impractical for a circuit with many lookup gates. We resolve this issue by giving a protocol called cq that is quasi-linear in n and has no dependence on N after the preprocessing step.

1.1 Comparison of results

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+22]	O($N \log N$) $\mathbb{F}$,$\mathbb{G}_1$	14 $\mathbb{G}_1$, 1 $\mathbb{G}_2$, 4 $\mathbb{F}$	$3n + m - \ell$ $\mathbb{G}_1$ exp, n $\mathbb{G}_2$ exp	2 $\mathbb{G}_1$, 1 $\mathbb{G}_2$	✓	✗
Caulk+ [PK22]	$O(N \log N)$ $\mathbb{F}$,$\mathbb{G}_1$	7 $\mathbb{G}_1$, 1 $\mathbb{G}_2$, 2 $\mathbb{F}$	$18n$ $\mathbb{G}_1$ exp	4 $\mathbb{G}_1$, 2 $\mathbb{F}$	✓	✗
Flookup [?]	$O(N \log^2 N)$ $\mathbb{F}$,$\mathbb{G}_1$	6 $\mathbb{G}_1$, 1 $\mathbb{G}_2$, 4 $\mathbb{F}$	$273n$ $\mathbb{G}_1$ exp	20 $\mathbb{G}_1$, 16 $\mathbb{F}$	✗	✗
ba*loo* [?]	$O(N \log N)$ $\mathbb{F}$,$\mathbb{G}_1$	12 $\mathbb{G}_1$, 1 $\mathbb{G}_2$, 4 $\mathbb{F}$	$8n$ $\mathbb{G}_1$ exp	6 $\mathbb{G}_1$, 4 $\mathbb{F}$	✓	✗
cq	$O(N \log N)$ $\mathbb{F}$,$\mathbb{G}_1$	$3n + 3a$ $\mathbb{G}_1$, 1 $\mathbb{G}_2$	$11n + 11a$ $\mathbb{G}_1$ exp , $\approx 54(n+a)\log(n+a)$ $\mathbb{F}$ mul	7 $\mathbb{G}_1$, 6 $\mathbb{F}$	✓	✓
cq*	$O(N \cdot n)$ $\mathbb{F}$,$\mathbb{G}_1$	$n + a$ $\mathbb{G}_1$, 1 $\mathbb{G}_2$	$9n + 9a$ $\mathbb{G}_1$ exp , $\approx 54(n+a)\log(n+a)$ $\mathbb{F}$ mul	9 $\mathbb{G}_1$, 6 $\mathbb{F}$	✓	✗

Table with relative proof size, prover ops, verifier ops proof-size caulk caulk+ flookup baloo 12 $\mathbb{G}_1$, 1 $\mathbb{G}_2$, 4 $\mathbb{F}$ this work 6 G1, 1 G2

1.2 Technical Overview

The innovation of Caulk While [ZBK+22, PK22, ?, ?] use preprocessing and pairings to extract a subtable of witness size;

Our approach here we use preprocessing and pairings more directly to run an existing lookup protocol - mvlookup, in time independent from table size -logarithmic derivative method Let's review this protocol: It relies on the following lemma from [?] that says that $f|_{\mathbb{H}} \in \mathsf{t}$ if and only if for some $m \in \mathbb{F}^N$

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

Roughly, the protocol of [?] checks this identity on a random β, by sending polynomials A and B that agree on $\mathbb{V}$ with the rational function values of the LHS and RHS respectively. Given commitments to A, B we can check the equality holds via various sumcheck techniques, e.g. that described in [BCR+19]. The RHS is not a problem because it is a sum of size n. Interpolating A, and computing its commitment is actually not a problem either, because the number of non-zero values is at most n. So if we precompute the commitments to the Lagrange base of $\mathbb{V}$ we're fine.

The main challenge, and innovation, is to convince the verifier $\mathbf{V}$ that A is correctly formed.

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

- Need way to compute A

2 Preliminaries

2.1 Notation:

$\mathbb{H}$- small space $\mathbb{V}$- big space Lagrange bases for big and small space

AGM - real and ideal pairing checks, agm - real and ideal pairing KZG

2.2 log derivative method

Lemma from mvlookup

Lemma 2.1. *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 derived from 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 $\left\{\left[x^i\right]_1\right\}_{i\in\{0,\ldots,N\}}$ 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 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 $Z'_{\mathbb{V}}(\omega^i)$ 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 ω^i/n in $O(N)$ group operations. In total, this takes $O(N \log N)$ field and group operations in $\mathbb{G}_1$. $\square$

We're now ready to state the main theorem of this section.

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 $\mathsf{srs} = \left\{\left[x^i\right]_1\right\}_{i\in[0,\ldots,N]}$ 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 the element $\mathsf{cm} = [Q(x)]_1$ where $Q \in \mathbb{F}_{<N}[X]$ is such that

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

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

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

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

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

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

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

□

4 Main protocol

Definition 4.1. *$\mathcal{R}$ is all pairs (cm, f) such that cm is a commitment to f and $f|_{\mathbb{H}} \subset T$.*
..bla problem is relation is defined only after srs is chosen

4.1 Definitions

Ad-hoc dfn of ks protocol for table lookup

Relations dependent on srs. Tuple gen,$\mathsf{IsInTable}_{\mathbb{H}}$

- $\mathsf{gen}(\mathfrak{t}, N) \rightarrow \mathsf{srs}$
- $\mathsf{IsInTable}_{\mathbb{H}}$ a protocol between $\mathbf{P}$ and $\mathbf{V}$ where $\mathbf{P}$ has input $f \in \mathbb{F}_{<n}[X]$, $\mathbf{V}$ has $[f(x)]_1$. Both have $\mathfrak{t}$ and srs. such that
  - Completeness:If $f|_{\mathbb{H}} \subset \mathfrak{t}$ then $\mathbf{V}$ outputs acc with probability one.
  - Knowledge soundness in the algebraic group model: For any $\mathfrak{t} \in \mathbb{F}^n$, the probability of any algebraic $\mathcal{A}$ to win the following game is $\mathsf{negl}(\lambda)$
    1. Let $\mathsf{srs} = \mathsf{gen}(\mathfrak{t}, N)$.
    2. $\mathcal{A}$ sends a message cm and values $f_1, \ldots, f_n$ such that $\mathsf{cm} = \sum_{i \in [n]} f_i \cdot [L_i(x)]_1$.
    3. $\mathcal{A}$ and $\mathbf{V}$ engage in the protocol $\mathsf{IsInTable}_{\mathbb{H}}(\mathfrak{t}, \mathsf{cm})$ with $\mathcal{A}$ taking the role of $\mathbf{P}$.
    4. $\mathcal{A}$ wins if
       - $\mathbf{V}$ outputs acc
       - $f|_{\mathbb{H}} \not\subset \mathfrak{t}$.

Main protocol: Preprocessed inputs: $[Z_{\mathbb{V}}(x)]_2$, $[T(x)]_2$ Input (cm, f).

Round 1:Committing to the multiplicites vector

1. $\mathbf{P}$ computes poly $m \in \mathbb{F}_{<N}[X]$ such that $m_i =$ number of times $\mathfrak{t}_i$ 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 the identity for A 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 $i \in [N]$, $A_i = m_i/(\mathfrak{t}_i + \beta)$.
3. $\mathbf{P}$ sends $\mathsf{a} := [A(x)]_1$.
4. $\mathbf{P}$ computes $\mathsf{q_a} := [Q_A(x)]_2$ 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 $\mathsf{q_b} := [B(x)]_1$.
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 $\mathsf{p} = [P(x)]_1$ where

$$P(X) := A(X)\cdot X^{d-N}$$

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

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

12. $\mathbf{V}$ checks that A has the appropriate degree:

$$e(\mathsf{a}, \left[x^{d-N}\right]_2) = e(\mathsf{p}, [1]_2).$$

Round 3: Checking the identities for B at random $\gamma \in \mathbb{F}$

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

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

4. As part of checking P is correct, $\mathbf{V}$ computes

$$P_\gamma := b_\gamma\cdot\gamma^{d-n}$$

5. To perform a batched KZG check for the correctness of the values $a_\gamma, b_\gamma, f_\gamma, P_\gamma$
   (a) $\mathbf{V}$ sends random $\eta\in\mathbb{F}$. $\mathbf{P}$ and $\mathbf{V}$ separately compute

$$v := b_\gamma + \eta\cdot f_\gamma + \eta^2\cdot Q_{b,\gamma} + \eta^3\cdot P_\gamma$$

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

$$h(X) := \frac{B(X)+\eta\cdot f(X)+\eta^2\cdot Q_B(X)+\eta^3\cdot P(X)-v}{X-\gamma}$$

(c) **V** computes

$$\mathsf{c} := \mathsf{b} + \eta \cdot \mathsf{f} + \eta^2 \cdot \mathsf{q_b} + \eta^3 \cdot \mathsf{p}$$

and checks that

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

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

(a) **P** and **V** separately compute

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

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

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

(c) **V** computes

$$\mathsf{c_0} := \mathsf{a} + \zeta \mathsf{b}$$

and checks that

$$e(\mathsf{c_0} - [u]_1, [1]_2) = e(\pi_0, [x]_2)$$

Stats: verifier pairings:4 - pair a with random combination of T and $\left[x^{d-N}\right]_2$, pair $\mathsf{q_a}$ with $Z_{\mathbb{V}}$.

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 π_0, π_γ can be computed in $2 \cdot n \log n$ $\mathbb{G}_1$-operations and $O(n \log n)$ $\mathbb{F}$-operations*

Knowledge soundness proof: Look at the following events

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.

[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.

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

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

[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.