cq:[*] Cached quotients for fast lookups

Liam Eagen   Dario Fiore   Ariel Gabizon
IMDEA   Zeta Function Technologies

December 20, 2022

Abstract

We present a protocol for checking the values of a committed polynomial $\phi(X) \in \mathbb{F}_{<N}[X]$ over a multiplicative subgroup $\mathbb{H} \subset \mathbb{F}$ of size n are contained in a table $T \in \mathbb{F}^N$. 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[?, ?, ?, ?] starting from Caulk [?], 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, as in [?, ?, ?] our construction relies on homomorphic table commitments, which makes them amenable to vector lookups in the manner described in Section 4 of [?].

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 [?], plookup [?] 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 left the intriguing question of whether the dependence on N could be made *sublinear* after performing a preprocessing step for the table T. Caulk [?] answered this question in the affirmative by leveraging bi-linear pairings, achieving a run time of $O(n^2 + n \log N)$. Caulk+ [?] 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.

[*]Pronounced "seek you".

1.1 Comparison of results

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 [?, ?, ?, ?] 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 descirbed in [?]. 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

Theorem 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 after a preprocessing step of $O(N \cdot \log N)$ operations. Given input $f \in \mathbb{F}_{<n}[X]$ computes in $O(n)$ $\mathbb{G}_1$ operations $\mathsf{cm} = [Q(x)]_1$ where $Q \in \mathbb{F}_{<N}[X]$ is such that*

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

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

Proof. By lemma 3.2, the quotient commitments $[Q_i(X)]_1$ can be computed in $O(N \log N)$ time. These satisfy the following equations that depend only on $T(X)$, not $f(X)$, and so can be precomputed in a preprocessing step.

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

By assumption, the polynomial $f(X)$ can be written as a linear combination of the n Lagrange basis polynomials for the set $\mathbb{H}$

$$f(X) = \sum_{i \in \mathbb{H}} f_i L_i(X)$$

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

$$f(X)T(X) = \sum_{i \in \mathbb{H}} f_i L_i(X)T(X) = \sum_{i \in \mathbb{H}} f_i t_i L_i(X) + f_i Z_{\mathbb{V}}(X) Q_i(X)$$

Rearranging terms and factoring out $Z_{\mathbb{V}}(X)$, it follows that commitments to both the quotient $Q(X)$ and remainder $R(X)$ can be computed in $O(n)$ group operations.

$$[Q(X)]_1 = \sum_{i \in \mathbb{H}} f_i \, [Q_i(X)]_1$$
$$[R(X)]_1 = \sum_{i \in \mathbb{H}} f_i t_i \, [L_i(X)]_1$$

$\square$

Lemma 3.2. *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. Substituting the definition of the Lagrange polynomial

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

We can write the quotient $Q_i(X)$ as the KZG opening for $T(\omega^i) = t_i$ scaled by the derivative of $Z_{\mathbb{V}}$ at ω^i.

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

First, the algorithm needs to compute the coefficients $T(X) = \sum_{i=0}^{N-1} \hat{t}_i X^i$ given the sequence of t_i values. This is possible in $O(N \log N)$ time using an FFT to interpolate $T(\omega^i) = t_i$.

Then, the algorithm of Feist and Khovratovich [] can be used to compute commitments to all the KZG proofs $[K_i(X)]_1$ for a polynomial in $O(N \log N)$. 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 $\mathbb{G}_1$. □

Lemma 3.3. *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 $\left\{\left[x^i\right]_1\right\}_{i \in \{0,\dots,d\}}$ computes for $i \in [N]$, the elements $q_i := \left[^{d-N} \cdot Q_i(x)\right]_1$ 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. Note that the commitments here are just the commitments from 3.2 scaled by X^{d-N}. Scaling a matrix-vector product by a scalar is equivalent to scaling the vector and then multiplying by the matrix. In this case, scaling the commitments by X^{d-N} is equivalent to performing the matrix multiplication on the vector of commitments $\left[X^i\right]_1$ for $i \in [d-N, d-1]$. The rest of the algorithm remains the same, so these q_i can also be computed using $O(N \log N)$ group operations. □

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,$\mathsf{IsInTable}_{\mathbb{H}}$

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

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

Round 1:Committing to the multiplicites vector

1. **P** computes poly $m \in \mathbb{F}_{<N}[X]$ such that $m_i =$ number of times t_i appears in $f|_{\mathbb{H}}$
2. **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. **V** chooses and sends random $\beta \in \mathbb{F}$.
2. **P** computes $A \in \mathbb{F}_{<N}[X]$ such that for $i \in [N]$, $A_i = m_i/(\mathsf{t}_i + \beta)$.
3. **P** sends $\mathsf{a} := [A(x)]_1$.

4. **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. **P** computes $B \in \mathbb{F}_{<n}[X]$ such that for $i \in [n]$, $B_i = 1/(f_i + \beta)$.
6. **P** sends $\mathsf{q_b} := [B(x)]_1$.
7. **P** computes $Q_B(X)$ such that

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

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

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

11. **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. **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. **V** sends random $\gamma, \eta, \zeta \in \mathbb{F}$.
2. **P** sends $b_\gamma := B(\gamma), Q_{b,\gamma} := Q_B(\gamma), f_\gamma := f(\gamma)$.
3. As part of checking the correctness of $\mathsf{q_b}$, **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, **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) **V** sends random $\eta \in \mathbb{F}$. **P** and **V** separately compute

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

(b) **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