Strict Linear Lookup Argument

Xiang Fu$^{0000-0002-6608-1654}$
Xiang.Fu@hofstra.edu, Hofstra University

June 12, 2023

Keywords: Lookup Argument, Quasi-Adaptive NIZK

Abstract

Given a table $\mathfrak{t} \in \mathbb{F}^N$, and a commitment to a polynomial $f(X) \in \mathbb{F}_{<n}[X]$ over a multiplicative subgroup $\mathbb{H} \subset \mathbb{F}$. The lookup argument asserts that $f|_{\mathbb{H}} \subset \mathfrak{t}$. We present a new lookup argument protocol that achieves strict linear prover complexity, after a pre-processing step of $O(N \log(N))$.

1 Introduction

A lookup argument [GW20] proves that each element of a committed smaller table of size n belongs to a bigger table of size N. The past year has witnessed a rapid improvement of prover complexity from $O(n^2 + n\mathsf{log}(N))$ in Caulk [ZBK+22], to $O(n^2)$ in Caulk+ [PK22], to $O(n\mathsf{log}^2(n))$ in Baloo and Flookup [ZGK+22, GK22], and to $O(n\mathsf{log}(n))$ in $\mathfrak{cq}$ [EFG22]. In this work, we further improve the state of the art to its theoretical optimal: $O(n)$ prover cost, $O(1)$ proof size and $O(1)$ verifier cost. *CORRECTION: we are not able to achieve O(1) proof size/verifier cost at the same time. Will submit another draft.*

Our technique builds upon $\mathfrak{cq}$ [EFG22], which is based on the following observation made in [Hab22]. Let $f(X) = \sum_{i=1}^{n} f_i \tau_i(X)$ where $\tau_i(X)$ are the Lagrange bases for $\mathbb{H}$. $f|_{\mathbb{H}} \subset \mathfrak{t}$ if and only if there exists $m \in \mathbb{F}^N$ such that $\sum_{i=1}^{N} \frac{m_i}{X+\mathfrak{t}_i} = \sum_{i=1}^{n} \frac{1}{X+f_i}$. $\mathfrak{cq}$ uses a Σ-protocol to reason about both sides of the equation and the $O(n\mathsf{log}(n))$ complexity arises in the processing of RHS. Our protocol retains the first half of $\mathfrak{cq}$ and improves the prover complexity of RHS to $O(n)$.

2 Preliminaries

2.1 Notations

Let $\mathcal{G}$ be a generator of bilinear groups, i.e., $(p, \mathbf{g}_1, \mathbf{g}_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$. Here $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ all have prime order p, with $\mathbf{g}_1$ ($\mathbf{g}_2$) as the generator of $\mathbb{G}_1$ ($\mathbb{G}_2$). $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is the bilinear map s.t. for any $a, b \in \mathbb{Z}_p$: $e(\mathbf{g}_1{}^a, \mathbf{g}_2{}^b) = e(\mathbf{g}_1, \mathbf{g}_2)^{ab}$ and $e(\mathbf{g}_1, \mathbf{g}_2)$ is the generator of $\mathbb{G}_T$. Following the notations in Groth16 [Gro16], we write $\mathbb{G}_1$ and $\mathbb{G}_2$ as additive groups. That is: given $a \in \mathbb{Z}_p$, we denote $\mathbf{g}_1{}^a$ as $[a]_1$, and similarly are group elements in $\mathbb{G}_2$ and $\mathbb{G}_T$ denoted. For instance, $\mathbf{g}_1{}^a\mathbf{g}_1{}^b$ is written as $[a]_1 + [b]_1$ or $[a+b]_1$, $(\mathbf{g}_2{}^a)^b$ as $[ab]_2$, and $e(\mathbf{g}_1{}^a, \mathbf{g}_2{}^b)$ is denoted as $[a]_1 \cdot [b]_2$ or $[ab]_T$.

2.2 Summary of $\mathfrak{cq}$ [EFG22]

We recall the idea of $\mathfrak{cq}$. Let n and N be both power of two where $n << N$. Let $\mathbb{V} = \left\{\omega^i\right\}_{i=1}^N$ where $\omega^N = 1$. $\mathbb{V}$, as a multiplicative subgroup of $\mathbb{F}$, needs to have Fast Fourier Transform (FFT) applicable. For instance, BLS12-381 supports N up to 2^{32}. Similarly, define $\mathbb{H} = \{\nu^i\}_{i=1}^n \subset \mathbb{F}$ as a multiplicative subgroup of size n. Define vanishing polynomial $Z_{\mathbb{V}}(X) = \prod_{i=1}^N (X - \omega^i)$, and $Z_{\mathbb{H}}(X) = \prod_{i=1}^n (X - \nu^i)$. Let $\{L_i(X)\}_{i=1}^N$ be the Lagrange bases of $\mathbb{V}$ s.t. $L_i(\omega_i) = 1$ and $L_i(\omega_j) = 0$ for $j \neq i$. Similarly define Lagrange bases $\{\tau_i(X)\}_{i=1}^n$ for $\mathbb{H}$. Assume that a trusted set-up provides KZG commitment [KZG10] keys in the form of $\{[x^i]\}_{i=0}^N$ where x is the trapdoor of the setup. A commitment to a polynomial $p(X)$ is $[p(x)]_1$. **Lookup Argument:** Given a table $\mathfrak{t} \in \mathbb{F}^N$ and $\mathbf{C}_f = [f(x)]_1$ to a polynomial $f(X) \in \mathbb{F}_{<n}[X]$, the goal is to prove that $f|_{\mathbb{H}} \subset \mathfrak{t}$.

The key to achieving prover complexity independent of table size N is that Lagrange bases and quotient polynomials can be pre-computed. First, $\mathfrak{t}$ can be characterized by a polynomial $T(X) = \sum_{i=1}^N \mathfrak{t}_i L_i(X)$ s.t. for each $i \in [1, N]$ $T(\omega^i) = \mathfrak{t}_i$. If $\{[L_i(x)]_1\}_{i=1}^N$ can be pre-computed, then $[T(x)]_1$ can be computed in $O(N)$ time. Similarly, $[f(x)]_1$ can be computed in $O(n)$ time. Define quotient polynomial $Q_i(X) = \frac{T(X) - \mathfrak{t}_i}{Z'_{\mathbb{V}}(\omega^i)(X - \omega^i)}$. It is shown in [EFG22] that $\{[Q_i(x)]\}_{i=1}^N$ can be computed in $O(N\mathsf{log}(N))$ time.

$\mathfrak{cq}$ is based on the following observation made in [Hab22]. $f|_{\mathbb{H}} \subset \mathfrak{t}$ if and only if there exists $m \in \mathbb{F}^N$ such that $\sum_{i=1}^N \frac{m_i}{X + \mathfrak{t}_i} = \sum_{i=1}^n \frac{1}{X + f_i}$. Intuitively, let the elements in $\mathfrak{t}$ be distinct. For each element $\mathfrak{t}_i$, the value of m_i is the number of times that $\mathfrak{t}_i$ appears in table f. Apparently, there are up to n non-zero m_i entries. Let $\beta \in \mathbb{F}$ is a random challenge supplied by the verifier, the equation in [Hab22] can be checked by:

$$\sum_{i=1}^N \frac{m_i}{\beta + \mathfrak{t}_i} = \sum_{i=1}^n \frac{1}{\beta + f_i} \tag{1}$$

Let $A_i = \frac{m_i}{\beta + \mathfrak{t}_i}$ and $A(X) = \sum_{i=1}^N A_i L_i(X)$. Similarly define $M(X) = \sum_{i=1}^N m_i L_i(X)$. Note that the prover does not have to compute them, but can compute their commitment: $[A(x)]_1$ and $[M(x)]_1$ in $O(n)$ time, because up to n entries of m_i are non-zero. The prover sends these commitments to verifier, and proves that they are well-formed by establishing that there exists a polynomial $Q(X)$ s.t.

$$A(X)(T(X) + \beta) - M(X) = Q(X)Z_{\mathbb{V}}(X) \tag{2}$$

Equation 2 can be verified by a pairing check and the key is to provide $[Q(x)]_1$, which can be computed in $O(n)$ from the pre-processed information: $\{[Q_i(x)]_1\}_{i=1}^N$.

Recall that we still need to argue for the LHS = RHS for Equation 1. Its LHS is $\sum_{i=1}^N A_i$. Based on the result of Aurora [BSCR+19], $\sum_{i=1}^N A(\omega^i) = N \cdot A(0)$. Thus, the prover just needs to compute $A(0)$ and provide its KZG evaluation proof $\left[\frac{A(x)-A(0)}{x}\right]_1$, which can be computed in $O(n)$ from the pre-processed information.

For the RHS of Equation 1, a similar polynomial $B(X)$ can be defined such that:

$$B(X)(f(X) + \beta) - 1 = Q_B(X)Z_{\mathbb{H}}(X) \tag{3}$$

Note that, however, it is *different* from Equation 2 in that the $T(X)$ in Equation 2 is replaced by $f(x)$. There is no pre-processed information for computing quotient polynomials for $f(x)$, as it is the secret witness of the prover. Then, the prover would have to compute $B(X)$ for the rest of the proof (round 3 in `cq` [EFG22]), and this incurs $O(n\mathsf{log}(n))$ field operations due to polynomial interpolation via FFT.

Our scheme differs from `cq` in the way how RHS is proved. We rely on the linear subspace argument [KW15] and its improved version in [CFQ19]. By applying a random combination scheme, we are able to prove the value of RHS with linear cost.

2.3 QA-NIZK for Linear Subspace

We recall the QA-NIZK presented in LegoSnark [CFQ19, Appendix D] which is adapted from [KW15] by removing its restriction on matrix dimension.

Given $[M]_1 \in \mathbb{G}_1^{l\times t}$, and $w \in \mathbb{Z}_q^t$, and $[x]_1 \in \mathbb{G}_1^l$, the Linear Subspace QA-NIZK proves the following statement: [1]

$$[x]_1 = [M]_1 \cdot w$$

The QA-NIZK consists of the following operations:

1. $\sigma_{ls} \leftarrow$ `SetupLS`$([M]_1)$ generates a prover/verifier key of size $O(l + t)$. [2] Intuitively, σ_{ls} encodes the public matrix $[M]_1$.

2. $([x]_1, \pi_{ls}) \leftarrow$ `ProveLS`(σ_{ls}, w) computes the the $[x]_1$ and generates the proof for knowledge of w. The prover spends one multi-exponentiation of $O(t)$, and the proof size is $O(1)$.

3. and $0/1 \leftarrow$ `VerifyLS`$(\sigma_{ls}, \pi, [x]_1)$ verifies the claim that the prover knows a secret witness w s.t. $[x]_1 = [M]_1 \cdot w$. It costs $O(l)$ pairings.

[1] To verifier, $[x]_1$ and $[M]_1$ are public and w is the secret witness of the prover.

[2] For convenience, we do not distinguish prover and verifier keys but verifier key is shorter.

```
Trusted Set-up:   $\sigma \leftarrow \texttt{Setup}(1^\lambda, \mathfrak{t}, N, n)$
    (S1-1) Compute $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{g}_1, \mathbf{g}_2, e) \leftarrow \mathcal{G}(1^\lambda)$.
    (S1-2) Sample $x$ from $\mathbb{Z}_p^*$, and compute $\left\{[x^i]_1\right\}_{i=1}^N$, and $\left\{[x^i]_2\right\}_{i=1}^N$.
    (S1-3) Compute $[Z_{\mathbb{V}}(x)]_2$, $[T(x)]_2$, $\{L_i(x)\}_{i=1}^N$, $\left\{\frac{L_i(x)-L_i(0)}{x}\right\}_{i=1}^N$,
    $\{[Q_i(x)]_1\}_{i=1}^N$ using the algorithm in [EFG22].
    (S2-1) Sample $u \in \mathbb{Z}_p^n$ and $\alpha \in \mathbb{Z}_p$, compute $\mathbf{C}_{\vec{1},\alpha} = \sum_{i=1}^n [u_i^2 \alpha^{i-1}]_1$.
    (S2-2) Compute $\{[\tau_i(x)]_j\}_{i=1}^n$ for $j \in \{1, 2\}$.
    (S2-3) Define $M_1 \in \mathbb{F}^{2\times n}$ as $M_1 = \begin{bmatrix} [[u_1]_1, ..., [u_n]_1], \\ [[\alpha^0]_1, ..., [\alpha^{n-1}]_1] \end{bmatrix}$. Define
    $M_2 \in \mathbb{F}^{2\times n}$ as $M_2 = \begin{bmatrix} [[u_1]_1, ..., [u_n]_1], \\ [[1]_1, ..., [1]_1] \end{bmatrix}$. Define $M_3 \in \mathbb{F}^{2\times n}$ as
    $M_3 = \begin{bmatrix} [[u_1]_1, ..., [u_n]_1], \\ [[\tau_1(x)]_1, ..., [\tau_n(x)]_1] \end{bmatrix}$. Compute $\sigma_{M_i} \leftarrow \texttt{SetupLS}(M_i)$ for
    $i \in \{1, 2, 3\}$.
    (S2-4) Return
    $\sigma = \begin{pmatrix} \left\{[x^i]_1\right\}_{i=1}^N, \left\{[x^i]_2\right\}_{i=1}^N, [T(x)]_2, \{L_i(0)\}_{i=1}^N, [1/x]_1, \\ \{[L_i(x)]_1\}_{i=1}^N, \left\{[\frac{L_i(x)-Li(0)}{x}]_1\right\}_{i=1}^N, \{[Q_i(x)]_1\}_{i=1}^N, [Z_{\mathbb{V}}(x)]_2, \\ \{\tau_i(x)\}_{i=1}^n, \{[\tau_i(x)]_j\}_{i=1}^n \text{ for } j \in \{1, 2\}, \\ \{[u_i]_1\}_{i=1}^n, \{[u_i]_2\}_{i=1}^n, \sigma_{M_1}, \sigma_{M_2}, \sigma_{M_3}, \mathbf{C}_{\vec{1},\alpha}, \sum_{i=1}^n [u_i^2]_1 \end{pmatrix}$.
```

Figure 1: Set-Up

3 Linear Lookup Argument Protocol

3.1 Insights

Recall that our goal is to prove RHS of Equation 1 with linear cost. We rely on the QA-NIZK for linear subspace. One can use it to prove the equivalence of two commitments and the sum of secrets behind a commitment. We elaborate the details below.

Let $\{[g_i]_1\}_{i=1}^n$ be a Pedersen Vector Commitment key where the prover has no knowledge of g_i and any linear relation of the key. let $\{[\alpha^{i-1}]_1\}_{i=1}^n$ be a KZG commitment key. Define $\mathbf{C}_f = \sum_{i=1}^n f_i[g_i]_1$ and $\mathbf{C}_{f,\alpha} = \sum_{i=1}^n f_i[\alpha^{i-1}]_1$. Let $[x]_1 = [\mathbf{C}_f, \mathbf{C}_{f,\alpha}]$, and $w = \{f_i\}_{i=1}^n$ for the QA-NIZK for Linear Subspace. One can prove the equivalence of the two commitments (encoding the same vector f), using the following matrix:

$$M_1 = \begin{bmatrix} [g_1]_1 & [g_2]_1 & \dots & [g_n]_1 \\ [\alpha^0] & [\alpha^1]_1 & \dots & [\alpha^{n-1}]_1 \end{bmatrix} \tag{4}$$

Clearly the prover cost is $O(n)$ and verifier cost is $O(1)$, because M has 2 rows.

Using the following M, one can prove the sum of a Pedersen vector commitment, i.e., the $[x]_1$ in the QA-NIZK relation is: $[\mathbf{C}_f, [\sum_{i=1}^n f_i]_1]$, and the witness is vector f.

$$M_2 = \begin{bmatrix} [g_1]_1 & [g_2]_1 & \dots & [g_n]_1 \\ [1]_1 & [1]_1 & \dots & [1]_1 \end{bmatrix} \tag{5}$$

We use the above gadgets to prove the correctness of polynomials for the RHS of Equation 2 and the value of RHS. We present the entire protocol in Figure 2.

3.2 Trusted Set Up

We show the set-up in Figure 1. It has two parts: (1) steps labeled with `S1` are for round-1 of $\mathfrak{cq}$ [EFG22], and (2) steps `S2` for the new RHS proof.[3] The complexity is $O(N\mathsf{log}(N))$. We provide the detailed analysis in Appendix A.1. For convenience of presentation we do not distinguish prover and verifier key.

3.3 LHS: $\mathfrak{cq}$ Round-1

We now present the complete protocol in Figure 2. We apply Fiat-Shamir so that the protocol is converted to non-interactive. It provides the `Prove`() and `Verify`() operations. The `Prove`() takes $\mathfrak{t}$, f as input and generates $\mathbf{C}_f = \sum_{i=1}^n f_i[\tau_i(x)]_1$, and it produces a proof π for f being a sub-table of $\mathfrak{t}$.

The first part (labeled `P1`) in `Prove`() is essentially the round-1 of $\mathfrak{cq}$ [EFG22]. Its goal is to prove that the $A(X)$, $Q(X)$, $M(X)$ on the LHS of Equation 2 are well-formed, and it tries to convince the verifier that the value of LHS is $A(0)\cdot N$. This part is verified using two pairing checks in step (V1) of `Verify`().

The prover cost is $O(n)$ field and group operations. The analysis is presented in in Appendix A.2.

3.4 RHS

We now show how to achieve $O(1)$ prover cost for RHS, using the linear subspace QA-NIZK. The prover has a secret table $f \in \mathbb{Z}_p^n$. Let $s = \sum_{i=1}^n \frac{1}{f_i+\beta}$. The goal is to prove that s is indeed the value of RHS.

The prover prepares two arrays: $\{a_i = f_i + \beta\}_{i=1}^n$ and $\left\{b_i = \frac{1}{f_i+\beta}\right\}_{i=1}^n$. Basically, we need a protocol that certifies that (1) for all i: $a_i b_i = 1$, (2) $a_i = f_i + \beta$, and (3) $s = \sum_{i=1}^n b_i$.

Claim (3) is addressed by `P2-2` in Figure 2. Claim (2) needs two steps. First, we present $\mathbf{C}_{a-\beta} = \sum_{i=1}^n (a_i - \beta)[u_i]_1$ as a commitment to f_i over bases $\{[u_i]_1\}_{i=1}^n$. We use Step `P2-4`, to show that it commits to the same vector (i.e.,

[3] We note that the the process could be actually split into two parts: one trusted set-up which does not take $\mathfrak{t}$, and a pre-processing step prepared by the prover that processes $[Q_i(x)]_1$ using the prover key. They are combined in this way for convenience of presentation.

1 **Prove:** $(\mathbf{C}_f, \pi) \leftarrow \texttt{Prove}(\sigma, \mathsf{t}, f)$

(P1-1) Retrieve data in prover/verifier key σ as shown in `S2-4` in Figure 1. Compute $\mathbf{C}_f = \sum_{i=1}^{n} f_i[\tau_i(x)]_1$.

(P1-2) Compute $m \in \mathbb{F}^N$ s.t. $\sum_{i=1}^{N} \frac{m_i}{\beta+\mathsf{t}_i} = \sum_{i=1}^{n} \frac{1}{\beta+f_i}$.

(P1-3) Define $M(x) = \sum_{i=1}^{N} m_i L_i(X)$. Compute $[M(x)]_1$ using $\{L_i(x)\}_{i=1}^{N}$. Note: prover does not compute $M(x)$. Apply Fiat-Shamir and compute $\beta = \mathsf{hash}(\mathbf{C}_f, [M(x)]_1)$

(P1-4) For each $i \in [1, N]$ define $A_i = \frac{m_i}{\beta+\mathsf{t}_i}$. Define $A(X) = \sum_{i=1}^{N} A_i L_i(X)$. Compute $a_0 = A(0)$, and $[A(x)]_1$.

(P1-5) Let $Q(X)$ be the polynomial s.t. $A(X)(T(X)+\beta) - M(X) = Q(X)Z_{\mathbb{V}}(X)$. Compute $[Q(x)]_1$ using the the $\{[Q_i(x)]_1\}_{i=1}^{N}$ in σ following algorithm in [EFG22].

(P1-6) Compute $\pi_{a_0} = \left[\frac{A(x)-a_0}{x}\right]_1$.

(P1-7) Let $\pi_L = ([A(x)_1], [Q(x)_1], [M(x)]_1, a_0, \pi_{a_0})$.

(P2-1) Compute $a, b \in \mathbb{Z}_p^n$ s.t. for each $i \in [1, n]$: $a_i = f_i + \beta$ and $b_i = \frac{1}{a_i}$. Compute $s = \sum_{i=1}^{n} b_i$.

(P2-2) Compute $([\mathbf{C}_a, \mathbf{C}_{a,\alpha}], \pi_a) \leftarrow \texttt{ProveLS}(\sigma_{\sigma_{M_1}}, a)$.

(P2-3) Compute $([\mathbf{C}_b, [s]_1)], \pi_b) \leftarrow \texttt{ProveLS}(\sigma_{M_2}, b)$. Compute $\mathbf{C}_{b,2} = \sum_{i=1}^{n} b_i[u_i]_2$.

(P2-4) Compute $([\mathbf{C}_{a-\beta}, \mathbf{C}_f)], \pi_f) \leftarrow \texttt{ProveLS}(\sigma_{M_3}, f)$.

(P2-4) Let $\pi_R = (s, \mathbf{C}_a, \mathbf{C}_{a,\alpha}, \mathbf{C}_b, \mathbf{C}_{b2}, \mathbf{C}_{a-\beta}, \pi_a, \pi_b, \pi_f)$.

(P2-5) Let $\pi = (s, \pi_L, \pi_R)$. Return $(\mathbf{C}_f, \pi)$.

2 **Verify:** $0/1 \leftarrow \texttt{VerifyAQ}(\sigma, \pi)$

Retrieve all elements from σ, and parse π as shown in `P1-7` and `P2-4`. Return 1 if and only if all of the following checks pass.

(V1) Proof related to LHS.

1. $e([A(x)]_1, [T(x)]_2) = e([Q(x)]_1, [Z_V(x)]_2) \cdot e([M(x)]_1 - \beta[A(x)]_1, [1]_2)$
2. $e([A(x)]_1 - [a_0]_1, [1]_2) = e(\pi_{a_0}, [x]_2)$

(V2) Proof related to RHS.

1. $\texttt{VerifyLS}(\sigma_{M_1}, [\mathbf{C}_a, \mathbf{C}_{a,\alpha}], \pi_a)$.
2. $\texttt{VerifyLS}(\sigma_{M_2}, [\mathbf{C}_b, [s]_1], \pi_b))$.
3. $\texttt{VerifyLS}(\sigma_{M_3}, [\mathbf{C}_{a-\beta}, \mathbf{C}_f], \pi_f))$.
4. $\mathbf{C}_a = \mathbf{C}_{a-\beta} + \beta(\sum_{i=1}^{n}[u_i]_1)$.
5. $e(\mathbf{C}_{a,\alpha}, \mathbf{C}_{b,2}) = e(\mathbf{C}_{\vec{1},\alpha}, [1]_2)$.
6. $e(\mathbf{C}_b, [1]_2) = e([1]_1, \mathbf{C}_{b,2})$.

(V3) Verify Fiat-Shamir: $\beta = \mathsf{hash}(\mathbf{C}_f, [M(x)]_1)$.

(V4) Verify $a_0 \cdot N = s$.

Figure 2: Complete Linearlookup Protocol

f) as $\mathbf{C}_f$ (albeit they are over two different commitment keys). Then it is easy to show the relation between $\mathbf{C}_{a-\beta}$ and $\mathbf{C}_a$ with:

$$\mathbf{C}_{a-\beta} + \sum_{i=1}^{n} (\beta [u_i]_1) = \mathbf{C}_a$$

Note that as $\sum_{i=1}^{n} [u_i]_1$ is included in the verifier key, the check of the above is $O(1)$.

Lastly, claim (1) is addressed by step `P2-1` and verifier step V2.5. Essentially V2.5 asserts the following:

$$\sum_{i=1}^{n} \alpha^{i-1} (a_i b_i [u_i^2]_1) = \sum_{i=1}^{n} \alpha^{i-1} [u_i^2]_1$$

This is a randomized combination of all equations of $a_i b_i = 1$ for $i \in [1, n]$. We present a detailed analysis of complexity in Appendix A.3. **CORRECTION: this claim is wrong. Will be corrected and another draft will be submitted.**

3.5 Discussion

It is possible to take advantage of the data parallelism of circuit if we feed the RHS of Equation 1 to a general purpose proof system, to accomplish $O(n)$ prover complexity. Most work exploiting data parallel circuits exist in GKR or Sum-check protocol based zk-proof systems [Tha13, WJB+17, LYH+21]. The basic idea is to encode layers of circuit via multi-linear extension polynomials. The prover complexity can be linear, however, for either or both of the proof size and verifier work, the cost is at least $O(\mathsf{log}(n))$ where n is the circuit width (already taking into the account that depth of circuit is a constant).

4 Conclusion

Let N and n be the size of bigger and smaller tables in the lookup argument. By improving the `cq` protocol we show that after a pre-processing step of $O(N \mathsf{log}(N))$, the prover cost is $O(n)$, and the verifier cost and proof size are both $O(1)$.

Acknowledgment: We would like to thank Dr. Ariel Gabizon for correcting the error in the protocol.

A Complexity Analysis

A.1 Set Up

We recall the analysis of the complexity of `cq` set-up [EFG22]. Given secret x, the complexity to compute $\left\{[x^i]_1\right\}_{i=1}^{N}$ and $\left\{[x^i]_2\right\}_{i=1}^{N}$ is $O(N)$. Since FFT is applicable to $\mathbb{V}$, $Z_{\mathbb{V}}(X) = X^N - 1$. Hence computing $[Z_{\mathbb{V}}(x)]_1$ is $O(1)$.

Consider Lagrange polynomial $L_i(X) = \prod_{1\leq j\leq N \ \wedge \ j\neq i} \frac{X-\omega^j}{\omega^i-\omega^j}$, it can be represented as: $L_i(X) = \frac{Z_{\mathbb{V}}(X)}{(X-\omega^i)\prod_{j\in[1,N] \ \wedge \ j\neq i}(\omega^i-\omega^j)}$. It is shown in Baloo [ZGK+22] that to compute $\left\{\prod_{j\in[1,N] \ \wedge \ j\neq i}(\omega^i-\omega^j)\right\}_{i=1}^{N}$ via FFT the cost is $O(N\mathsf{log}(N))$ field operations. Then $\{[L_i(x)]_1\}_{i=1}^N$ and $\left\{[\frac{L_i(x)-L_i(0)}{x}]_1\right\}_{i=1}^{N}$ costs $O(N\mathsf{log}(N))$.

Finally the set-up can compute $\{[Q_i(x)]_1\}_{i=1}^N$ following the algorithm presented in $\mathfrak{cq}$ [EFG22], which uses Toeplitz matrix [FK23]. $T(X)$ can be computed through interpolation over FFT domain which takes $O(N\mathsf{log}(N))$. Thus, the steps `S1-1` to `S1-3` in Figure 1 up takes $O(N\mathsf{log}(N))$ field and group operations. Similarly the complexity for `S2` steps is $O(n\mathsf{log}(n))$.

A.2 LHS Complexity

We briefly show that the complexity of the first part (round-1 of $\mathfrak{cq}$) in Figure 2. The prover does not have to compute $M(x)$, $A(x)$. However, $[M(x)]_1$ can be computed in linear time because there are up to n entries of m_i being nonzero. Similarly $[A(x)]_1$ is computed in $O(n)$. In [EFG22], $[Q(x)]_1$ is computed in $O(n)$ using the prover key $[Q_i(x)]_1$. $A(0)$ can be computed as $\sum_{i=1}^N A_i L_i(0)$, which is $O(n)$ given the $\{L_i(0)\}$ in prover key. Similarly, π_{a_0} can be computed as $[\frac{\sum_{i=1}^N A_i L_i(x) - a_0}{x}]_1$, which can be computed as as $[\frac{\sum_{i=1}^N A_i (L_i(x) - L_i(0))}{x}]_1 + \sum_{i=1}^N (A_i \cdot L_i(0) - a_0/N)[\frac{1}{x}]_1$, which is linear given the pre-processed information in prover key.

A.3 RHS Complexity

We now discuss the complexity of the second part of Figure 2. Apparently, the cost of `P2-2` to `P2-4` are both $O(n)$ because the matrix involved for the linear subspace argument have 2 rows. `P2-1` costs $O(n)$ field operations.

For the `V2` checks in the `Verify`() operation. All `VerifyLS`() operations costs $O(1)$ given the row number of matrices. Step (4) of `V2` has constant cost because $\sum_{i=1}^n [u_i]_1$ is given in verifier key.

In summary, the proof size is $O(1)$ and the verifier cost is $O(1)$.

References

[BSCR+19] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for r1cs. In *EUROCRYPT*, pages 103–128, 2019.

[CFQ19] M. Campanelli, D. Fiore, and A. Querol. LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs. IACR Cryptol. ePrint Arch. 2019:142, 2019.

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

[FK23] D. Feist and D. Khovratovich. Fast Amortized KZG Proofs. IACR Cryptol. ePrint Arch., 2023.

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

[Gro16] J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT*, pages 305–326, 2016.

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

[Hab22] U. Habock. Multivariate lookups based on logarithmic derivatives. IACR Cryptol. ePrint Arch., 2022.

[KW15] E. Kiltz and H. Wee. Quasi-adaptive NIZK for Linear Subspaces Revisited. In *EUROCRYPT*, pages 101–128, 2015.

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

[LYH$^+$21] Y. Li, C. Ye, Y. Hu, I. Morpheus, Y. Guo, C. Zhang, Y. Zheng, Z. Sun, Y. Lu, and H. Wang. ZKCPlus: Optimized Fair-exchange Protocol Supporting Practical and Flexible Data Exchange. In *CCS*, pages 3002–3021, 2021.

[PK22] J. Posen and A. Kattis. CaulkPlus: Table-independent lookup arguments. IACR Cryptol. ePrint Arch., 2022.

[Tha13] J. Thaler. Time-optimal interactive proofs for circuit evaluation. IACR Cryptol. ePrint Arch., 2013.

[WJB$^+$17] R. Wahby, Y. Ji, A. Blumberg, J. Thaler, M. Walfish, and T. Wies. Full Accounting for Verifiable Outsourcing. In *CCS*, pages 2071–2086, 2017.

[ZBK$^+$22] A. Zapico, V. Buterin, D. Khovratovich, M. Maller, A. Nitulescu, and M. Simkin. Caulk: Lookup Arguments in Sublinear Time. In *CCS*, pages 3121–3134, 2022.

[ZGK$^+$22] A. Zapico, A. Gabizon, D. Khovratovich, M. Maller, and C. Ràfols. Baloo: Nearly Optimal Lookup Arguments. IACR Cryptol. ePrint Arch., 2022.