Ba*loo*: Nearly Optimal Lookup Arguments

Arantxa Zapico*, Ariel Gabizon[3], Dmitry Khovratovich[1], Mary Maller[1], and Carla Ràfols[2]

[1] Ethereum Foundation
[2] Universitat Pompeu Fabra
[3] Zeta Function Technologies
arantxa.zapico@upf.edu, ariel.gabizon@gmail.com, khovratovich@gmail.com, mary.maller@ethereum.org, carla.rafols@upf.edu

Abstract. We present Ba*loo*, the first protocol for lookup tables where the prover work is linear on the amount of lookups and independent of the size of the table. Ba*loo* is built over the lookup arguments of Caulk and Caulk+, and the framework for linear relations of Ràfols and Zapico.
Our protocol supports *commit-and-prove expansions*: the prover selects the subtable containing the elements used in the lookup, that is unknown to the verifier, commits to it and later prove relation with the committed element. This feature makes Ba*loo* especially suitable for prover input-output relations on hash functions, and in particular to instantiate the Ethereum Virtual Machine (EVM).

1 Introduction

The rise of succinct proving systems in the recent decade has brought us close to one of the Holy Grails of computer science. Namely, being able to prove a large computation while spending not much more time on the proof than on the computation itself. We know how to make a proof only a handful of bytes large, and how to make the verifier run in a millisecond – but the prover time remains a bottleneck. Even though it is asymptotically a linear function of the computation time, the constants in these asymptotics are far too big to prove even moderate-sized programs.

Quite recently, some great hurdles have been overcome for reducing the prover time. First, we learned how to prove not only finite field statements but also lookups in tables such as caches or databases with Plookup [GW20]. These are crucial to implement regular (2^n) integer arithmetic and bit-oriented algorithms such as modern hash functions. The new technique, however, lower bounds the prover time to the table size N and thus limits the usage of big tables. And here came the second breakthrough, Caulk [ZBK+22,PK22], as the first method to prove m lookups in more reasonable $O(m^2)$ time, i.e. independent of N. Where Caulk is suitable for big tables but inconvenient for big lookups, Caulk has been calling for the last and final improvement, where one could finally prove m lookups in linear time.

At the same time, a number of computationally powerful blockchains, with Ethereum being the most prominent example, barely withstand the demand for higher transaction rate and computational bandwidth. One bold attempt is to get consensus on the computation without every node repeating work is to use a SNARK as a certificate of correctness. However, efforts to build a prover for Ethereum's virtual machine [BaCCL21,Eth22,Pol22,Sta22,Zha22,zks22] have been hindered by the cost of proving the Keccak hash function, even if a prover is lookup-enhanced. As Keccak is used in Merkle tree of the blockchain state, a proof for all state transitions in one block results in tens of millions of lookups — the amount insurmountable even for Caulk.

In this work we present Ba*loo*, a protocol that finally achieves the goal of proving m lookups in (almost) linear time. The prover is quasilinear in the field and linear in the group. This is thanks to a number of new techniques designed around proving statements over sets that are not multiplicative subgroups (where we cannot use Fast Fourier Transforms). The Ba*loo* protocol works smoothly with several multicolumn tables and Ba*loo* can be used as a drop in replacement to the Halo2 lookup argument with much better prover

* This work was done while Arantxa Zapico was a PhD student at Universitat Pompeu Fabra, funded by Protocol Labs PhD Fellowship PL-RGP1-2021-062.

efficiency. In other words Ba*loo* is backwards compatible with instantiations of the Halo2 SNARK that use KZG commitments. This means that Ba*loo* is potentially the difference between a zkEVM protocol being viable or not.

2 Related Work

In Table 1 we compare the concrete costs of the closest schemes to this work, when compiled using the KZG polynomial commitment scheme [KZG10] (the schemes [GW20], [BGH20], [GK22] are described in the IOP setting only). Plookup [GW20] and Halo2 [BGH20] require no preprocessing but the prover work in the group is quasilinear on the size of the table. They can be compiled using any polynomial commitment scheme including solutions that do not require pairings. Caulk [ZBK+22] introduced the first solution with prover work that is sublinear in the size of the table by using preprocessing , but they incur a quadratic cost in the number of lookups. Posen and Kattis introduced Caulk+[PK22] , an improvement over Caulk that leads to a table-independent prover, still quadratic on number of lookups. Gabizon and Khovratovich [GK22] have recently reduced the prover complexity to quasi-linear on the lookups while retaining a table-independent prover. However, their techniques rely on committing to a table as roots of a polynomial instead of coefficients. This means their commitments are not homomorphic, which limits the applicability of their solution to stand alone set membership proofs and makes it challenging to use their lookup to speed up SNARK provers (see Section 8). Ba*loo* also has prover complexity that is quasi-linear on the lookups in the field and linear on the group while retaining a table-independent prover, and the commitments are homomorphic. Currently there does not exist a pairing free lookup argument with quasilinear prover work in the size of the table.

Scheme	Preprocessing	Proof size	Prover work		Verifier work
			group	field	
Plookup [GW20]	–	$5\mathbb{G}_1, 9\mathbb{F}$	$O(N)$	$O(N \log N)$	2P
Halo2 [BGH20]	–	$6\mathbb{G}_1, 5\mathbb{F}$	$O(N)$	$O(N \log N)$	2P
Caulk [ZBK+22]	$O(N \log N)$	$14\mathbb{G}_1$, $1\mathbb{G}_2$, $4\mathbb{F}$	$15m$	$O(m^2 + m\log(N))$	4P
Caulk+ [PK22]	$O(N \log N)$	$7\mathbb{G}_1, 1\mathbb{G}_2, 2\mathbb{F}$	$8m$	$O(m^2)$	3P
Flookup [GK22]	$O(N \log^2 N)$	$7\mathbb{G}_1, 1\mathbb{G}_2, 4\mathbb{F}$	$O(m)$	$O(m \log^2 m)$	3P
This work: Ba*loo*	$O(N \log N)$	$12\mathbb{G}_1, 1\mathbb{G}_2, 4\mathbb{F}$	$14m$	$O(m \log^2 m)$	5P

Table 1. Cost comparison of our scheme with other pairing-based lookups. N is the size of the table and m the size of the set to be opened. The preprocessing costs are given in the number of group operations.

Other approaches, such as discrete-log based [BG13][GK15][BCC+15][BG18] require no trusted setup but incur a linear verifier. Bootle et al. [BCG+18] initially suggested the use of lookup arguments to improve the prover time in proving machine computations. Their solution was targetted the TinyRAM virtual machine [BCG+13]. Campanelli et al. [CFH+21] present also an scheme for position-hiding linkability of RSA accumulators for large prime numbers and Pedersen commitments. Concretely they achieve good efficiency: their proof size is constant and their proving times do not depend on the size of the accumulator. Further, they can support larger lookup tables than Ba*loo* because they are not constrained by the size of their setup. However, their scheme crucially relies on groups of hidden order such as a trusted RSA modulus or class groups.

Lookup arguments are often used in the context of key-value lookups in verifiable registries [CDGM19]. Multiple works [TBP+19,MKL+20,HHK+21,TFBT21] explore how to ensure the correctness of the table that is used in verifiable registries. Campanelli et al. [CEO22] demonstrate how homomorphic commitments can be used to build key-value lookups. Their solution is zero-knowledge, does not require a trusted setup

or pairings, and uses techniques similar to Section 8. However their prover runs in linear time. Agrawal and Raghuraman [AR20] build key-value lookups using hidden order groups. Campanelli et al. [CFG+20] use lookup arguments to construct *verifiable decentralized storage* and achieve a sublinear prover assuming preprocessing.

Benarroch et al. [BCF+21] discuss commit-and-prove set membership proofs which is a useful primitive for constructing modular zero-knowledge proofs.

3 Preliminaries

3.1 Notation

A bilinear group gk is a tuple $gk = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2)$ where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order q, the elements $\mathcal{P}_1$, $\mathcal{P}_2$ are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable, non-degenerate bilinear map, and there is no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. Elements in $\mathbb{G}_\gamma$, are denoted implicitly as $[a]_\gamma = a\mathcal{P}_\gamma$, where $\gamma \in \{1, 2, T\}$ and $[1]_T = e(\mathcal{P}_1, \mathcal{P}_2)$. With this notation, $e([a]_1, [b]_2) = [ab]_T$.

Let $\lambda \in \mathbb{N}$ denote the security parameter and 1^λ its unary representation. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}^+$ is called *negligible* if for all $c > 0$, there exists k_0 such that $\mathsf{negl}(k) < \frac{1}{k^c}$ for all $k > k_0$. For a non-empty set S, let $x \leftarrow S$ denote sampling an element of S uniformly at random and assigning it to x.

PPT denotes probabilistic polynomial-time, and algorithms are randomized unless explicitly noted otherwise. Let $y \leftarrow A(x; r)$ denote running algorithm A on input x and randomness r and assigning its output to y and $y \leftarrow A(x)$ denotes $y \leftarrow A(x; r)$ for a uniformly random r.

Lagrange Polynomials and Roots of Unity. Along this work, we consider two groups of roots of unity.

We use ω to denote a primitive root of unity such that $\omega^N = 1$, and define $\mathbb{H} = \{\omega, \ldots, \omega^N\}$. $\lambda_s(X)$ denotes the sth Lagrange interpolation polynomial, i.e., $\lambda_s(X) = \prod_{i \neq s} \frac{X - \omega^i}{\omega^s - \omega^i}$ and $z_H(X) = \prod_{s=1}^{N}(X - \omega^s) = X^N - 1$ the vanishing polynomial of $\mathbb{H}$. For a set of indexes $I \subset [N]$, we consider the subset $\mathbb{H}_I \subset \mathbb{H}$ of size $|I| = k$ such that $\mathbb{H}_I = \{\omega^s\}_{s \in I} = \{\xi_i\}_{i=1}^k$. $\{\tau_i(X)\}_{i=1}^k$ and $z_I(X)$ are the corresponding Lagrange and vanishing polynomials. We also assume ν to be a primitive root of unity of order m, and $\mathbb{V} = \{\nu^1, \ldots, \nu^m\}$. For simplicity, we may use also ν_j for ν^j. The associated Lagrange interpolation polynomials will be denoted as $\{\mu_j(X)\}_{j=1}^m$ and the vanishing polynomial as $z_V(X)$.

3.2 Cryptographic Assumptions

The security of our protocols holds in the Algebraic Group Model (AGM) of Fuchsbauer et al. [FKL18], using the *bilinear* version of the q-dlog and q-sfrac assumptions [GG17,BB04]. In the AGM, adversaries are restricted to be *algebraic* algorithms, namely, whenever $\mathcal{A}$ outputs a group element $[y]$ in a cyclic group $\mathbb{G}$ of order p, it also outputs its representation as a linear combination of all previously received group elements. In other words, if $[y] \leftarrow \mathcal{A}([x_1], \ldots, [x_m])$, $\mathcal{A}$ must also provide $\vec{z}$ such that $[y] = \sum_{j=1}^m z_j[x_j]$. This definition generalizes naturally in asymmetric bilinear groups with a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where the adversary must construct new elements as a linear combination of elements in the same group.

3.3 The KZG Polynomial Commitment Scheme

Our construction heavily relies on the polynomial commitment introduced by Kate, Zaverucha and Goldberg in [KZG10] that we described below. As noted in Caulk [ZBK+22], the protocol can be slightly modified to support degree checks, so it consists on a tuple $\big(\mathsf{KZG.Setup}, \mathsf{KZG.Commit}, \mathsf{KZG.Open}, \mathsf{KZG.Verify}\big)$ such that:

- $\mathsf{srs}_{\mathsf{KZG}} \leftarrow \mathsf{KZG.Setup}\big(\mathsf{par}_{\mathsf{KZG}}, d\big)$: On input the system parameters and a degree bound d, it outputs a structured reference string $\mathsf{srs}_{\mathsf{KZG}} = \big(\{[x^i]_{1,2}\}_{i=1}^d\big)$.
- $\mathsf{C} \leftarrow \mathsf{KZG.Commit}\big(\mathsf{srs}_{\mathsf{KZG}}, p(X)\big)$: On input polynomial $p(X)$, it outputs $\mathsf{C} = [p(x)]_1$.

– $(s, \pi_{\mathsf{KZG}}) \leftarrow \mathsf{KZG.Open}\big(\mathsf{srs}_{\mathsf{KZG}}, p(X), \alpha\big)$: Let $\deg < d$ be the degree of $p(X)$. Given $\alpha \in \mathbb{F}$, prover computes

$$q(X) = \frac{p(X) - p(\alpha)}{X - \alpha} \ ,$$

sets $s = p(\alpha), [Q]_1 = [q(x)x^{d-\deg+1}]_1$, and outputs $(s, \pi_{\mathsf{KZG}} = [Q]_1)$.
– $1/0 \leftarrow \mathsf{KZG.Verify}\big(\mathsf{srs}_{\mathsf{KZG}}, \mathsf{C}, \deg, \alpha, s, \pi_{\mathsf{KZG}}\big)$: Verifier accepts if and only if

$$e(\mathsf{C} - s, [x^{d-\deg+1}]_2) = e([Q]_1, [x - \alpha]_2).$$

Multiple Openings. We also implement the optimization noted in [ZBK+22] to open one polynomial to many distinct points. In a nutshell, given the polynomial $p(X)$, a vector of opening points $\vec{\alpha} \in \mathbb{F}^t$ and $\vec{s}$ such that $s_i = p(\alpha_i)$ for all $i = 1, \ldots, t$, prover and verifier define $C_{\vec{\alpha}}(X)$ as the unique polynomial of degree $t-1$ such that $C_{\vec{\alpha}}(\alpha_i) = s_i$ for all $i \in [t]$. Then, $p(\alpha_i) = s_i$ for all $i = 1, \ldots, y$ if and only if there exists $q(X)$ such that

$$p(X) - C_{\vec{\alpha}}(X) = \prod_{i=1}^{m}(X - \alpha_i)q(X).$$

Subset openings. It is crucial for our construction the subvector opening scheme of Tomescu et. al [TAB+20] that works for the vector commitment inspired by KZG.

Given an encoding $C(X) = \sum_{s=1}^{N} c_s\lambda_s(X)$ to a vector $\vec{c} \in \mathbb{F}^N$ and $C_I(X) = \sum_{i=1}^{k} \hat{c}_i\tau_i(X)$, where $\{\tau_i(X)\}$ are the Lagrange interpolation polynomials of the set $\{\xi_i\}_{i=1}^k = \{\omega^s\}_{s\in I}$, they note that for $z_I(X) = \prod_{s\in I}(X - \omega^s)$,

$$C(X) - C_I(X) = z_I(X)Q_I(X),$$

for some polynomial $Q_I(X)$ if and only if $\hat{c}_i = c_s$ for the unique pair (i, s) such that $\xi_i = \omega^s$.

What is more, it is demonstrated in [TAB+20] that the prover can compute $[Q_I]_1$ by performing k group and $\mathcal{O}(k\log^2(k))$ field operations, given they already have stored proofs $\{[Q_s]_1\}_{s\in I}$ that $C(\omega^s) = c_s$. Precomputing all the proofs $\{[Q_s]_1\}_{s=1}^N$ can be done in time $\mathcal{O}(N\log N)$ using techniques by Feist and Khovratovich [FK20]. More details are given in Section 7.

3.4 Caulk+ core

We use a subroutine of the lookup argument Caulk+, which we call Caulk+ core, as a building block of Ba*loo*. Caulk+, by Posen and Kattis [PK22], is an improvement on Caulk [ZBK+22] that takes prover computation in the group from $O(m^2 + m\log(B))$ to just $O(m^2)$ for m lookups on tables of size N. Following their blueprint, our first step is to create a subtable. That is, given a public vector $\vec{c}$, encoded as polynomial $C(X)$, and elements $\mathsf{t} \in \mathbb{G}_1$ and $k \in \mathbb{N}$, Caulk+ core proves that t is a commitment to a subvector $\vec{t} \in \mathbb{F}^k$ of $\vec{c}$. In other words, there exist $I \subset [N]$ such that $\vec{t} = (c_s)_{s\in I}$.

Here Caulk+ core considers the subvector length k as a public parameter, and proves the following relation

$$\mathbb{R}_{\mathsf{subtable}} = \left\{ (C(X), \mathsf{t}, [z_I]_2);\ (\mathbb{H}_I, t(X)) \;\middle|\; \begin{array}{l} \mathbb{H}_I = \{\xi_1, \ldots, \xi_k\} \subset \mathbb{H} \\ \forall \xi \in \mathbb{H}_I,\ t(\xi) = C(\xi) \\ \mathsf{t} = [t(x)]_1 \\ [z_I]_2 = [z_I(x)]_2 \text{ for } z_I(X) = \prod_{i=1}^{k}(X - \xi_i) \end{array} \right\},$$

where $C(X) = \sum_{s=1}^{N} c_s\lambda_s(X)$ for $\{\lambda_s(X)\}_{s=1}^N$ the Lagrange interpolation polynomials of a subgroup of roots of unity $\mathbb{H} = \{\omega^s\}_{s=1}^N$.

In Figure 1 we describe the protocol for $\mathbb{R}_{\mathsf{subtable}}$ [PK22]. The prover running time of Caulk+ core is quasilinear in k for field and linear in k for group operations (assuming some precomputations), as we see below (more details in Section 7).

Theorem 1. *The protocol in Fig. 1 is knowledge sound in the algebraic group model assuming the* q-dlog *and the* q-sfrac *assumptions hold.*

We refer the reader to Appendix A for the proof.

$$\mathsf{srs} = \{[x^s]_{1,2}\}_{s=1}^{N},\ \mathsf{C} = [C(x)]_1,\ [z_H(x)]_1 = [\prod_{s=1}^{N}(x-\omega^s)]_1,\ [z_I]_2, \mathsf{t}$$

$\mathsf{Prover}_{\mathsf{C+}}$**:** takes as input $\mathbb{H}_I, t(X), \{[Q_i(x)]_1 = [(C(x)-C(\xi_i))/(x-\xi_i)]_1, [H_i(x)]_1 = [z_H(x)/(x-\xi_i)]_1\}_{i=1}^{k}$

- Set $z_I(X) = \prod_{i=1}^{k}(X-\xi_i)$ and $z_{H\setminus I} = \prod_{s\in[N]\setminus I}(X-\omega^s)$
- Compute $W_1 = \left[\frac{C(x)-t(x)}{z_I(x)}\right]_1$ and $W_2 = [z_{H\setminus I}(x)]_1$
- Compute $W_3 = [(z_I(x) - x^k)x^{N-k+1}]_1$

It outputs $\pi_{\mathsf{C+}} = (W_1, W_2, W_3)$.

$\mathsf{Verifier}_{\mathsf{C+}}$**:** Accepts if and only if

$$(i)\quad e\left(\mathsf{C}-\mathsf{t}, [1]_2\right) = e\left(W_1, [z_I]_2\right)$$

$$(ii)\quad e\left([z_H(x)]_1, [1]_2\right) = e\left(W_2, [z_I]_2\right)$$

$$(iii)\quad e\left([x^{N-k+1}]_1, [z_I]_2 - [x^k]_2\right) = e\left(W_3, [1]_2\right)$$

Fig. 1. The Caulk+ core [PK22] quasilinear-time protocol for proving $\mathbb{R}_{\mathsf{subtable}}$ that a commitment contains a subtable.

3.5 Generalized Sumcheck

Following [BCR+19], in Section 5.4 we construct a scheme for inner product relations that rely on the univariate sumcheck argument for the elements in the set $\mathbb{H}_I$. Since the latter is not enforced to be a group of roots of unity, but just a subset of one, we use the generalized variant of the sumcheck:

Theorem 2 (Generalized Sumcheck [RZ21]). *Let $\mathbb{H}_I = \{\xi_i\}_{i=1}^{k}$ be an arbitrary subset of size k in some finite field $\mathbb{F}$ and $z_I(X)$ its vanishing polynomial. For any $P(X) \in \mathbb{F}[X]$, $\sum_{i=1}^{k} P(\xi_i) = \sigma$ if and only if there exist polynomials $Q(X) \in \mathbb{F}[X]$, $R(X) \in \mathbb{F}_{\leq k-2}[X]$ such that*

$$P(X)N_{\mathbb{H}_I}(X) - \sigma = XR(X) + z_I(X)Q(X),$$

where $N_{\mathbb{H}_I}(X) = \sum_{i=1}^{k} \tau_i(0)^{-1}\tau_i(X)$ and $\tau_i(X)$ is the ith Lagrange polynomial associated to ξ_i and the set $\mathbb{H}_I$.

4 Overview

In this section we provide a technical overview of Ba*loo* protocol, which proves the following statement:

Given element cm *and a public set represented as vector $\vec{c} \in \mathbb{F}^N$, there exists $\vec{a} \in \mathbb{F}^m$ such that all elements of $\vec{a}$ are elements of $\vec{c}$ and* cm *is a commitment to $\vec{a}$.*

We approach this statement in two steps. First we select a subvector $\vec{t}$ of $\vec{c}$ by trimming all elements not in $\vec{a}$, and use Caulk+ core protocol [PK22] to prove, in a committed form, its well formation. Second, as in Caulk and Caulk+, we prove, again in a committed form, that $\vec{a}$ is a result of some expansion of $\vec{t}$, i.e., we design a lookup argument for an *unkown* table $\vec{t}$. However, and this is our main contribution, we replace the (implicit) lookup argument in [ZBK+22,PK22] by a variant of the linear argument in [RZ21] so that **both** steps are quasilinear in the lengths k, m of $\vec{t}$ and $\vec{a}$, assuming some precomputations. As precomputations are the same as in Caulk [ZBK+22], we get a prover with quasilinear of m online time and quasilinear of N offline (precomputation) time in the field and just $O(m)$ group operations in the online phase, as announced.

Select a subvector. Following Caulk+ core (Section 3.4), we denote by C a commitment to $\vec{c}$. Then we create a commitment t to a subvector $\vec{t} \in \mathbb{F}^k$ defined by a set of index $I \subset [N]$, that is, $\vec{t} = (c_s)_{s \in I}$. To prove well formation of t, the prover provides $W_1, [z_I]_2$ such that

$$e(\mathsf{C} - \mathsf{t}, [1]_2) = e(W_1, [z_I]_2),$$

and a proof of well formation of $[z_I]_2$ which convinces the verifier that $[z_I]_2$ is a commitment to the vanishing polynomial of some set $\mathbb{H}_I \subset \mathbb{H}$. Let us denote the roots of unity elements of $\mathbb{H}_I$ by $\{\xi_1, \xi_2, \ldots, \xi_k\}$ i.e. $\xi_i = \omega^s$ for some $s \in I$, following the order in $\mathbb{H}$.

If the prover has access to precomputed proofs $\{[Q_s]_1\}_{s=1}^N$ of opening for all the elements $c_s \in \vec{c}$ and individual proofs $\{[H_s]_1\}_{s=1}^N$ of statements "$(X - \omega^s)|z_H(X)$", it performs only $2k$ group operations to compute W_1 and thus convince the verifier that t is a commitment to some subvector of $\vec{c}$.

Expansion as a linear relation. Now we engage in the second task: given elements t, cm that commit to unknown vectors $\vec{t} \in \mathbb{F}^k, \vec{a} \in \mathbb{F}^m$ using the Lagrange basis corresponding to some unknown set $\mathbb{H}_I$ of size k and $\{\mu_j(X)\}_{j=1}^m$ to public subgroup $\mathbb{V}$, respectively, prove that for every $j \in [m]$ there exists $i \in [k]$ such that $a_j = t_i$. The latter equation has a simple algebraic representation: as $\vec{a}$ is a vector generated using elements of $\vec{t}$, there exists a matrix $M \in \mathbb{F}^{m \times k}$ of 1s and 0s such that

$$\mathbf{M}\vec{t} = \vec{a}. \tag{1}$$

Concretely, the non-zero elements are $m_{j,i}$ for j, i such that $a_j = t_i$. We then prove that relation (1) also holds in the committed form. Here t and cm are commitments to $\vec{t}$ and $\vec{a}$, whereas we employ a special technique to commit to a matrix.

Proving linear relations is usually done via a lincheck argument, e.g. [BCR+19]. Ràfols and Zapico in [RZ21] separate a lincheck argument into two parts. First, prover and verifier engage in a Checkable Subspace Argument (CSS) where prover convinces the verifier that a polynomial $D(X)$ encodes a random vector $\vec{d}$ in the rowspace of $\mathbf{M}$ sampled with the verifiers' coins. This can also be seen as a partial evaluation problem: if one defines a bivariate polynomial that encodes the matrix

$$M(X, Y) = (\mu_1(X), \ldots, \mu_m(X))\ \mathbf{M} \begin{pmatrix} \tau_1(X) \\ \vdots \\ \tau_k(X) \end{pmatrix},$$

the goal of a CSS argument is to show that $D(X) = M(\alpha, X) = \sum_{i=1}^k \sum_{j=1}^m M_{i,j}\tau_i(X)\mu_j(\alpha)$. This allows to reduce the statement in eq. (1) to a single inner product relation $\vec{d} \cdot \vec{t} = \sum_{j=1}^m a_j \mu_j(\alpha)$, that can be proven with the univariate sumcheck [BCR+19].

We modify the framework for linear relation of Ràfols and Zapico in [RZ21] in several ways:

- **CSS.** We give a new definition of Commit-and-Prove CSS which works for matrices that are chosen in a commit-and-prove fashion [CFQ19], that is, the prover selects matrix $\mathbf{M}$, communicates it to the verifier in a succinct manner and then convinces them that $\mathbf{M}\vec{t} = \vec{a}$. Importantly, the prover must convince the verifier that the committed matrix $\mathbf{M}$ has a certain form, i.e., that its rows are unit vectors, so the linear relation represents a lookup. Also, the basis to encode the vectors in the rowspace $\vec{\tau}(X)$ is also communicated succintly by the prover. This is in contrast to the constructions of CSS schemes in the holographic model considered in [RZ21], where the matrix M was fixed and preprocessed offline and where $\vec{\tau}(X)$ was fixed.
  The checkable subspace sampling (CSS) technique ensures that a commitment $[D]_1$ is to $D(X)$, as defined by verifier's coins. For this we adapt a construction of CSS given in [RZ21] for so called basic matrices, with only a non-zero element per column. Since a lookup matrix has only a non-zero element per row, it is the transpose of a basic matrix. Since $\mathbf{M}$ is only known by the prover in our new construction, we replace the offline phase usually performed by some untrusted party with a commitment phase performed by the prover itself.

To adapt the CSS in [RZ21] to our case, we observe that if $M(X,Y) = \sum_{i=1}^{k}\sum_{j=1}^{m} M_{i,j}\tau_i(X)\mu_j(Y)$ is the encoding of a matrix, $E(X) = M(\beta, X)$ encodes a vector sampled in the column space of $\mathbf{M}$ and a vector sampled in the row space of its transpose, a basic matrix, using same coins β. So we use the argument in [RZ21] for well formation of $E(X)$ and then use the fact that if $D(X) = D(X,\alpha)$ encodes a sampling in the row space using coins α, it must be the case that $E(\alpha) = D(\beta)$.

– **Inner Product.** In Section 5.4 we present a scheme for proving inner product relations between encodings $D(X)$ and $t(X)$ to vector $\vec{d}$ as described above and table $\vec{t}$, respectively. Both $t(X)$ and $D(X)$ can be naturally written in the Lagrange basis $\{\tau_i(X)\}_{i=1}^{k}$ corresponding to set $\mathbb{H}_I$. This is not a subgroup of the field. so use a generalized univariate sumcheck that works in this setting due to [RZ21]. We show that this result allows to prove a "twisted inner product relation" $\sum_{i=1}^{k} t_i d_i \tau_i(0)$. To cancel out the undesired $\tau_i(0)$ factors, in the CSS argument we will in fact sample set $\vec{d}$ to be a vector in the rowspace of M where coordinate i is divided by $\tau_i(0)^{-1}$.

Summary. Overall, the expansion protocol takes as input $\phi(X), t(X), z_I(X)$ and aims to show that $\phi(\xi) = t(\xi)$ for all $(X-\xi)$ dividing $z_I(X)$. For that, we

1. Prove that an element $[D]_1$ is the commitment to $D(X) = M(X,\alpha) = \sum_{j=1}^{m} \mu_j(\alpha)\tau_{\mathsf{col}(j)}(X)(\tau_{\mathsf{col}(j)}(0))^{-1}$, where α is a random point sampled by the verifier, and $\mathsf{col}(j)$ is the column of non-zero element in row j i.e. $M_{j,\mathsf{col}(j)} = 1$. Here $\{\mu_j(X)\}_{j=1}^{m}$ is a set of known Lagrange polynomials and $\{\tau_i(X)\}_{i=1}^{k}$ is a set of unknown Lagrange polynomials defined by the set of points $\mathbb{H}_I = \{\xi_i\}_{i=1}^{k}$ such that $(X-\xi_i)$ divides $z_I(X)$. For this step, we create $E(X) = M(\beta, X)$ and prove its well formation, i.e., we prove it is the encoding of a vector sampled in the row space of $\mathbf{M}^\top$. Then, we prove $E(\alpha) = D(\beta)$ and thus $D(X)$ is a vector sampled in the row space of $\mathbf{M}$.
2. From $\mathsf{t}, [D]_1$, prove that $\vec{d}\cdot\vec{t} = \sum_{j=1}^{m} a_j\mu_j(\alpha) = \phi(\alpha)$.

We combine this scheme with Caulk+ core (Fig. 1) and obtain Ba*loo*, a lookup argument proving that all the elements in $\vec{a}$, committed to in cm, are included in $\vec{c}$.

5 Building Blocks

In this section we introduce argument cp-expansion, a polynomial holographic proof (PHP) [CFF+21] to show that the vector $\vec{a}$ encoded in a polynomial $a(X)$ is an expansion of the vector $\vec{t}$ encoded in a polynomial $t(X)$, which means that all for all $j \in [m]$ there exists $i \in [k]$ such that $a_j = t_i$. cp-expansion uses two core building blocks: (i) a checkable subspace sampling proof to prove that some polynomial $D(X)$ encodes a vector in the row space of a matrix $\mathbf{M}$ defining the expansion, and (ii) a proof that some inner product argument holds between the vectors encoded in $D(X), t(X)$, and $a(X)$. We present these building blocks as PHP and compile them in Section 6.

5.1 Commit-and-Prove Checkable Subspace Sampling

The first step for proving the relation between $\vec{t}$ and $\vec{a}$ through polynomial encodings, is that prover and verifier agree on a polynomial $D(X)$ encoding a random element in the row space of matrix $\mathbf{M}$ such that $\mathbf{M}\vec{t} = \vec{a}$. In the setting of SNARKs, matrix $\mathbf{M}$ is part of the instance and, therefore, known by prover and verifier, while in our case, $\mathbf{M}$ is decided by the prover and the verifier only needs to be convinced that it has the correct form. More concretely, $\vec{t}$ is an *expansion* of $\vec{a}$ if and only if $\mathbf{M}$ is a matrix that has unit vectors in its rows, that is, there exists only one non-zero element in each row of $\mathbf{M}$ and it equals 1. Overall, the prover chooses $\mathbf{M}$

For some technical reasons[4], it will be simpler to define the argument avoiding explicit reference to matrix $\mathbf{M}$ and refer instead only to polynomials. In these terms, what we want to prove is that $D(X)$ is the correct

[4] The matrix $\mathbf{M}$ needs to depend on the order of $\tau_i(X)$, although the argument does not enforce any order of $\mathbb{H}_I$. Using polynomials allows us to complete ignore the order of the elements of the set $\mathbb{H}_I$.

linear combination of the polynomials $\vec{M}(X) = (M_1(X), \ldots, M_k(X))^\top$ that are some encoding of the rows $\vec{M}_i$ of $\mathbf{M}$.

We call this variant of CSS schemes a commit-and-prove CSS, a PHP where instead of having an indexer that in an offline phase computes polynomials describing the matrix, we have that the prover commits to a matrix and then proves attributes of it. In our commit-and-prove CSS, the prover and verifier first engage in a commit and sample phase, during which they jointly agree on the statement being proven. Afterwards the prover and verifier engage in a proving phase where the prover demonstrates that the statement is correct.

Definition 1 (Commit-and-Prove CSS). *A commit-and-prove checkable subspace sampling argument over a field $\mathbb{F}$, is a PHP that defines a set of allowed matrices $\mathcal{M}_I$ and runs in four different stages:*

- ***Commit Phase:*** *For some set of maximal size k, the prover $\mathcal{P}_{\mathsf{CSS}}$ sends a commitment $\mathsf{cm}_{\mathbb{H}_I}$ to a set $\mathbb{H}_I$ and a commitment $\mathsf{cm}_{\vec{M}}$ to a vector $\vec{M}(X)$ in some allowed set $\mathcal{M}_I(X)$.*
- ***Sampling Phase****: Prover $\mathcal{P}_{\mathsf{CSS}}$ and verifier $\mathcal{V}_{\mathsf{CSS}}$ engage in an interactive protocol* Sampling*. In some round, the verifier sends $\mathsf{cns} \leftarrow \mathcal{C}$, and the prover replies with a polynomial $D(X) = \vec{s}^\top \vec{M}(X)$, for $\vec{s} = \mathsf{Smp}(\mathsf{cns})$.*
- ***Proving Phase****: $\mathcal{P}_{\mathsf{CSS}}$ and $\mathcal{V}_{\mathsf{CSS}}$ engage in an interactive protocol to prove that*

$$\big(\mathsf{cm}_{\mathbb{H}_I}, \mathsf{cm}_{\mathbf{M}}, \mathsf{cns}, D(X)\big) \in \mathcal{L}_{\mathsf{CSS}}$$

for

$$\mathbb{R}_{\mathsf{CSS}} = \left\{ \big(\mathsf{cm}_{\mathbb{H}_I}, \mathsf{cm}_{\mathbf{M}}, \mathsf{cns}, D(X)\big), \quad (\mathbb{H}_I, \vec{M}(X), D(X)) \;\middle|\; \begin{array}{c} \mathsf{cm}_{\mathbb{H}_I} = \mathsf{Commit}(\mathbb{H}_I),\ |\mathbb{H}_I| = k, \\ \mathsf{cm}_{\vec{M}} = \mathsf{Commit}(\vec{M}(X)),\ \vec{M}(X) \in \mathcal{M}_I(X) \\ \vec{s} = \mathsf{Smp}(\mathsf{cns}); \\ D(X) = \vec{s}^\top \vec{M}(X) \end{array} \right\}.$$

- ***Decision Phase****: When the proving phase is concluded, the verifier outputs a bit indicating acceptance or rejection.*

We note that here the term commit it is used in a loose sense to refer to some well-defined polynomial encoding of the vectors.

Soundness. A Commit-and-Prove checkable subspace sampling argument is ϵ-sound if for any polynomial time prover $\mathcal{P}^*_{CSS}$:

$$\mathsf{Prob}\left[\begin{array}{c} \mathsf{instance} \notin \mathbb{R}_{\mathsf{CSS}} \\ b = 1 \end{array} \;\middle|\; \begin{array}{c} \mathsf{cm} \leftarrow \mathcal{P}^*_{CSS}; \mathsf{instance} \leftarrow \mathsf{Sample}\langle \mathcal{P}^*_{CSS}(\mathbb{H}_I, \vec{M}(X)), \mathcal{V}_{CSS}(\mathsf{cm})\rangle; \\ b \leftarrow \langle \mathcal{P}^*_{\mathsf{CSS}}(\mathsf{instance}), \mathcal{V}_{\mathsf{CSS}}(\mathsf{instance})\rangle \end{array} \right] \leq \epsilon.$$

5.2 Our Concrete CSS Relation

Similar to [RZ21], we encode matrix $\mathbf{M}$ following Marlin [CHM+20], through the bivariate polynomial

$$M(X, Y) = \sum_{j=1}^{m} \tau_{\mathsf{col}(j)}(0)^{-1} \tau_{\mathsf{col}(j)}(X) \mu_j(Y),$$

where $\mathsf{col} : [m] \mapsto [k]$ is such that $\mathsf{col}(j) = i$ if and only if $M_{j,i}$ is the only non-zero element in row j of $\mathbf{M}$, and $\{\mu_j(X)\}_{j=1}^m$, $\{\tau_i(X)\}_{i=1}^k$ are the Lagrange interpolation polynomials of some set $\mathbb{V}$ of size m and $\mathbb{H}_I$ of size k, respectively. As in both mentioned works, computing an encoding of a vector sampled in the row space of $\mathbf{M}$ using verifier's coin α, is done through a partial evaluation $M(X, \alpha)$. That is,

$$D(X) = M(X, \alpha) = \sum_{j=1}^{m} \tau_{\mathsf{col}(j)}(0)^{-1} \tau_{\mathsf{col}(j)}(X) \mu_j(\alpha)$$

is an encoding of $\vec{d} = \sum_{j=1}^{m} \mu_j(\alpha)\vec{M}_j$, where $\vec{M}_j$ is the jth row of $\mathbf{M}$ and is a unit vector in $\mathbb{F}^k$. The only difference between their encodings and ours is that we use the set of polynomials $\{\hat{\tau}_i(X) = \frac{\tau_i(X)}{\tau_i(0)}\}_{i=1}^{k}$ instead of simply $\tau_i(X)$, i.e. $D(X) = \sum_{i=1}^{k} d_i\hat{\tau}_i(X)$. The reason is that for the inner product in Section 5.4, we will need this *normalized* variant of the Lagrange polynomials that interpolate $\mathbb{H}_I$.

For each I, the set of allowed vectors polynomials that encode the set of allowed matrices is

Therefore, we define the allowed set of vectors of polynomials as:

$$\mathcal{M}(X)_I = \left\{(\hat{\tau}_{\mathsf{col}(1)}(X), \ldots, \hat{\tau}_{\mathsf{col}(m)}(X)) : \text{ for some } \mathsf{col} : [m] \mapsto [k]\right\},$$

where $\{\tau_i(X)\}_{i=1}^{k}$ are the Lagrange polynomials for the set $\mathbb{H}_I$, and $\hat{\tau}_i(X) = \tau_i(0)^{-1}\tau_i(X)$.

Proving well formation of $D(X)$ can be thought now as proving that $D(X) = \vec{s}^{\top}\vec{M}(X)$ for some $\vec{(}X) \in \mathcal{M}(X)_I$ and $\vec{s} = \mathsf{Smp}(\mathsf{cns})$.

The commitment algorithm is given by

$$\mathsf{Commit}(\mathbb{H}_I) = z_I(X) = \prod_{\xi \in \mathbb{H}_I} (X - \xi)$$
$$\mathsf{Commit}(\vec{M}(X)) = v(X) \text{ where } v(\nu_j) = \xi_{\mathsf{col}(j)}^{-1} \text{ for all } \nu_j \in \mathbb{V}$$

Note that $v(X)$ uniquely defines some vector $\vec{M}(X) \in \mathcal{M}_{\mathbb{H}_I}$, as it is isomorphic to a map from $[m] \mapsto [k]$.

In the sampling phase the verifier chooses $\alpha \leftarrow \mathbb{F}$ randomly from the field and the prover sets $D(X)$ accordingly. Then the proving phase is run with respect to the relation $\mathbb{R}_{z_I,\vec{M}(X)} \in \mathbb{R}_{\mathsf{CSS}}$.

$$\mathbb{R}_{z_I,\vec{M}(X)} = \left\{ \ (\alpha, D(X)),\ (\vec{M}(X), D(X)) \ : \ \vec{s} = (\mu_j(\alpha))_{j=1}^{m},\ D(X) = \vec{s}^{\top}\vec{M}(X) = \textstyle\sum_{j=1}^{m} \mu_j(\alpha)\hat{\tau}_{\mathsf{col}(j)}(X) \right\}.$$

5.3 The Scheme

In Fig. 2 we provide a formal PHP for our CSS proving system. During the committing and sampling phase the prover and verifier agree an instance $z_I(X), v(X), \alpha, D(X)$ where $z_I(X) = \prod_{i=1}^{k}(X - \xi_i)$. Note that $z_I(X)$ is proven to be correctly formed in the Caulk+ core protocol (see Section 3.4).

To prove well formation of $D(X)$, the prover sends a commitment to the polynomial

$$E(X) = M(X, \beta) = \sum_{j=1}^{m} \tau_{\mathsf{col}(j)}(0)^{-1}\tau_{\mathsf{col}(j)}(\beta)\mu_j(X)$$

It proves that $E(X)$ has exactly this form by showing that (i) $E(X)$ has degree less than m and (ii) $E(\nu_j) = \tau_{\mathsf{col}(j)}(0)^{-1}\tau_{\mathsf{col}(j)}(\beta)$ for all $\nu_j \in V$. We have that point (ii) holds if and only if there exists $Q_1(X)$ such that

$$E(X)(\beta v(X) - 1) + z_I(\beta)z_I(0)^{-1} = z_V(X)Q_1(X).$$

Intuitively, this is done because we have CSS for matrices $\mathbf{M}'$ that have one non-zero element per column (basic matrices in [RZ21]), meaning we can sample an element in the row space of the transpose to $\mathbf{M}$, which is a vector in the column space of $\mathbf{M}$. Later, we prove well formation of the desired encoding $D(X) = M(X, \alpha)$ of a vector in the row space of $\mathbf{M}$ by linking $D(X)$ and $E(X)$. In terms of polynomials, we know how to prove partial evaluation of the polynomial $M(X, Y)$ at different values for X but not Y, so we do it and then relate $M(\beta, X)$ with $M(X, \alpha)$.

Comitting Phase: $\mathcal{P}_{\mathsf{CSS}}$ computes and outputs $z_I(X) = \prod_{i=1}^{k}(X - \xi_i)$ and $v(X) = \sum_{j=1}^{m} \xi_{\mathsf{col}(j)}^{-1}\mu_j(X)$.

Sampling Phase: $\mathcal{V}$ sends $\alpha \in \mathbb{F}$ and $\mathcal{P}$ computes and outputs $D(X) = M(X, \alpha) = \vec{\mu}(\alpha)\vec{M}(X) = \sum_{j=1}^{m} \mu_j(\alpha)\tau_{\mathsf{col}(j)}(X)(\tau_{\mathsf{col}(j)}(0))^{-1}$.

Proving Phase: $\mathcal{V}$ sends $\beta \in \mathbb{F}$. $\mathcal{P}$ computes $E(X) = M(\beta, X) = \sum_{j=1}^{m} \mu_j(X)\tau_{\mathsf{col}(j)}(\beta)(\tau_{\mathsf{col}(j)}(0))^{-1}$ and $Q_1(X)$ such that

$$E(X)(\beta v(X) - 1) + z_I(\beta)z_I(0)^{-1} = z_V(X)Q_1(X).$$

It outputs $\big(E(X), Q_1(X)\big)$, a proof that $\deg(E) < m$, and a proof that $z_I(X)$ is a commitment to k distinct roots.

Decision Phase: Accepts if and only if $(i)\ \deg(E) < m$,

$$(ii)\quad E(X)(\beta v(X) - 1) + z_I(\beta)z_I(0)^{-1} = z_V(X)Q_2(X)$$

$$(iii)\quad D(\beta) = E(\alpha)$$

$$(iv)\quad z_I(X) \text{ is a commitment to } k \text{ distinct roots}^5$$

Fig. 2. Commit-and-Prove CSS for $\vec{M}(X) \in \mathcal{M}_I(X)$.

Theorem 3. *The protocol in Fig 2 satisfies completeness.*

Proof. For equation (ii) recall that $\sum_{j=1}^{m} \mu_j(X) = 1$ and note that

$$\begin{aligned}
E(X)(\beta v(X) - 1) &= \left(\sum_{j=1}^{m} \tau_{\mathsf{col}(j)}(\beta)(\tau_{\mathsf{col}(j)}(0))^{-1}\mu_j(X)\right)\left(\beta\sum_{j=1}^{m} \xi_{\mathsf{col}(j)}^{-1}\mu_j(X) - \sum_{j=1}^{m}\mu_j(X)\right) \\
&= \left(\sum_{j=1}^{m} \tau_{\mathsf{col}(j)}(\beta)(\tau_{\mathsf{col}(j)}(0))^{-1}\mu_j(X)\right)\left(\sum_{j=1}^{m} \big(\beta\xi_{\mathsf{col}(j)}^{-1} - 1\big)\mu_j(X)\right) \\
&= \sum_{j=1}^{m} \tau_{\mathsf{col}(j)}(\beta)(\tau_{\mathsf{col}(j)}(0))^{-1}\big(\beta\xi_{\mathsf{col}(j)}^{-1} - 1\big)\mu_j(X) \mod z_V(X) \\
&= \sum_{j=1}^{m} \prod_{i\neq\mathsf{col}(j)} \frac{\beta - \xi_i}{\xi_{\mathsf{col}(j)} - \xi_i} \prod_{i\neq\mathsf{col}(j)} \frac{\xi_{\mathsf{col}(j)} - \xi_i}{-\xi_i}\ \ \xi_{\mathsf{col}(j)}^{-1}\big(\beta - \xi_{\mathsf{col}(j)}\big)\mu_j(X) \mod z_V(X) \\
&= \sum_{j=1}^{m} \prod_{i\neq\mathsf{col}(j)} \frac{\beta - \xi_i}{-\xi_i}\ \ \xi_{\mathsf{col}(j)}^{-1}\big(\beta - \xi_{\mathsf{col}(j)}\big)\mu_j(X) \mod z_V(X) \\
&= -z_I(\beta)z_I(0)^{-1}\sum_{j=1}^{m}\mu_j(X) \mod z_V(X) = -z_I(\beta)z_I(0)^{-1} \mod z_V(X)
\end{aligned}$$

Thus, there exists $Q_1(X)$ such that $E(X)(\beta v(X) - 1) + z_I(\beta)z_I(0)^{-1} = Q_1(X)z_V(X)$

Equation (iii) follows as $D(X) = M(X, \alpha)$ and $E(X) = M(\beta, X)$. □

Theorem 4. *If $z_I(X) = \prod_{i=1}^{k}(X - \xi_i)$ for unique roots ξ_i then the CSS protocol in Fig. 2 is sound.*

Proof. Set $v_j = v(\nu_j)$ for all $j \in [m]$, then there exists $q(X) \in \mathbb{F}[X]$ such that $v(X) = \sum_{j=1}^{m} v_j\mu_j(X) + z_V(X)q(X)$. We first argue that the PHP enforces the form of $E(X)$ with respect to the evaluations $\{v_j\}_{j\in[m]}$.

[5] In our protocol, well formation of $z_I(X)$ is given by Caulk+ core.

We then argue that given this $E(X)$, the form of $D(X)$ is determined by $v(X)$ and $z_I(X)$ except with negligible probability. Third we will show that v_j^{-1} is a root of $z_I(X)$ for all $j \in [m]$. Finally we will show that $D(X)$ is exactly in our allowed set of matrices.

Form of $E(X)$: We show that

$$E(X) = \sum_{j=1}^{m} \left(\frac{-z_I(\beta)}{z_I(0)(\beta v_j - 1)} \right) \mu_j(X)$$

Since $\deg(E) < m$, there exist coefficients $\{e_j\}_{j=1}^m$ such that $E(X) = \sum_{j=1}^m e_j \mu_j(X)$. So we have:

$$\begin{aligned} E(X)(\beta v(X) - 1) &= \left(\sum_{j=1}^{m} e_j \mu_j(X) \right) \left(\beta \sum_{j=1}^{m} v_j \mu_j(X) + z_V(X) q(X) - \sum_{j=1}^{m} \mu_j(X) \right) \\ &= \sum_{j=1}^{m} e_j \big(\beta v_j - 1\big) \mu_j(X) \mod z_V(X) \end{aligned}$$

and then equation (ii) says that for all $\nu_j \in \mathbb{V}$,

$$\sum_{j=1}^{m} e_j (\beta v_j - 1) \mu_j(\nu_j) + z_I(\beta) z_I(0)^{-1} = 0,$$

and then for all $j \in [m]$, $e_j(\beta v_j - 1) = -z_I(\beta) z_I(0)^{-1}$. Thus for all j, $\beta v_j - 1 \neq 0$ and it follows that $e_j = \frac{-z_I(\beta)}{z_I(0)(\beta v_j - 1)}$.

Form of $D(X)$: We show that

$$D(X) = -\sum_{j=1}^{m} \frac{z_I(X) \mu_j(\alpha)}{z_I(0) v_j (X - v_j^{-1})}$$

except with negligible probability. Let

$$f(X) = \sum_{j=1}^{m} \left(\prod_{s=1, s \neq j}^{m} (X v_s - 1) \right) z_I(0)^{-1} z_I(X) \mu_j(\alpha) + D(X) \prod_{j=1}^{m} (X v_j - 1)$$

Then at random β we have that

$$\begin{aligned} f(\beta) &= \sum_{j=1}^{m} \left(\prod_{s=1, s \neq j}^{m} (\beta v_s - 1) \right) z_I(0)^{-1} z_I(\beta) \mu_j(\alpha) + D(\beta) \prod_{j=1}^{m} (\beta v_j - 1) \\ \Rightarrow \frac{f(\beta)}{\prod_{j=1}^{m} (\beta v_j - 1)} &= \sum_{j=1}^{m} \frac{z_I(\beta)}{z_I(0)(\beta v_j - 1)} \mu_j(\alpha) + D(\beta) \\ \Rightarrow \frac{f(\beta)}{\prod_{j=1}^{m} (\beta v_j - 1)} &= -E(\alpha) + D(\beta) \end{aligned}$$

provided that $\prod_{j=1}^m (\beta v_j - 1) \neq 0$, which is the case as explained above. Now by the verifiers (iii) check we have that $-E(\alpha) + D(\beta) = 0$ and hence that $f(\beta) = 0$. If $f(X) \neq 0$ then $f(\beta) \neq 0$ except with negligible

probability because $v(X), z_I(X), D(X)$ are chosen before β. Thus

$$\begin{aligned} D(X) &= -\sum_{j=1}^{m} \frac{\left(\prod_{s=1,s\neq j}^{m}(Xv_s - 1)\right) z_I(0)^{-1} z_I(X)\mu_j(\alpha)}{\prod_{j=1}^{m}(Xv_j - 1)} \\ &= -\sum_{j=1}^{m} \frac{z_I(X)\mu_j(\alpha)}{z_I(0)v_j(X - v_j^{-1})} \end{aligned}$$

Form of $v(X)$: We show that for all $j \in [m]$, v_j^{-1} is a root of $z_I(X)$. In other words, there exists a map $\mathsf{col} : [m] \mapsto [k]$ such that for all $j \in [m]$, $v_j^{-1} = \xi_{\mathsf{col}(j)}$. Indeed, define $J = \{j : v_j^{-1} \notin \mathbb{H}_I\}$, and the set $V = \{v_j : j \in J\}$. We assume for contradiction that there is some v_j^{-1} that is not a root of $z_I(X)$, which means that V is not empty. Then,

$$D_{J^c}(X) = -\sum_{j\in[m]\setminus J} \frac{z_I(X)\mu_j(\alpha)}{z_I(0)v_j(X - v_j^{-1})}$$

is a polynomial, and we can write:

$$\begin{aligned} D(X) - D_{J^c}(X) &= -\sum_{j\in J} \frac{z_I(X)\mu_j(\alpha)}{z_I(0)v_j(X - v_j^{-1})} \\ &= -\frac{z_I(X)}{z_I(0)} \sum_{v\in V} \frac{1}{v(X - v^{-1})} \left(\sum_{j:v_j=v} \mu_j(\alpha)\right) = -\frac{z_I(X)}{z_I(0)} \sum_{v\in V} \frac{P_v(\alpha)}{v(X - v^{-1})}, \end{aligned}$$

where, for any $v \in V$, $P_v(X) = \sum_{j:v_j=v} \mu_j(X)$. Regardless of α, this identity can only hold if, for all $v \in V$, $P_v(\alpha) = 0$. Indeed, the left side is a polynomial because the prover sent $D(X)$ and $D_{J^c}(X)$ is a polynomial by definition of J, but a polynomial cannot be equal to a sum of non-trivial rational functions with different poles. The probability that $P_v(\alpha) = 0$ for all $v \in V$ can be bounded by the probability that $P_v(\alpha) = 0$ for any single $v \in V$. But this probability is at most $\frac{m-1}{|\mathbb{F}|}$, because all these polynomials were defined independently of α. We conclude that the probability that V is not empty is negligible.

***D*(*X*) is in the space of allowed matrices:** Let us substitute our mapping col from v_j^{-1} to the roots of $z_I(X)$ into our expression for $D(X)$:

$$D(X) = -\sum_{j=1}^{m} \frac{z_I(X)\mu_j(\alpha)}{z_I(0)\xi_{\mathsf{col}(j)}^{-1}(X - \xi_{\mathsf{col}(j)})} = \sum_{j=1}^{m} \frac{-\xi_{\mathsf{col}(j)} z_I(X)\mu_j(\alpha)}{z_I(0)(X - \xi_{\mathsf{col}(j)})}$$

Now the lagrange polynomials for the set $\{\xi_i\}_{i=1}^k$ are given by

$$\tau_i(X) = \prod_{s=1,s\neq i}^{k} \frac{X - \xi_s}{\xi_i - \xi_s} = \frac{z_I(X)}{(X - \xi_i)\prod_{s=1,s\neq i}^{k}(\xi_i - \xi_s)}$$

Thus

$$\tau_{\mathsf{col}(j)}(0) = \frac{z_I(0)}{-\xi_{\mathsf{col}(j)}\prod_{s=1,s\neq \mathsf{col}(j)}^{k}(\xi_{\mathsf{col}(j)} - \xi_s)} \text{ and } \tau_{\mathsf{col}(j)}(0)^{-1}\tau_{\mathsf{col}(j)}(X) = \frac{-\xi_{\mathsf{col}(j)} z_I(X)}{z_I(0)(X - \xi_{\mathsf{col}(j)})}$$

Substituting these into our expression for $D(X)$ yields our result

$$D(X) = \sum_{j=1}^{m} \tau_{\mathsf{col}(j)}(0)^{-1}\tau_{\mathsf{col}(j)}(X)\mu_j(\alpha) = \sum_{j=1}^{m} \mu_j(\alpha)\hat{\tau}_{\mathsf{col}(j)}(X)$$

□

5.4 Inner products from Generalized Sumcheck

In this section, we introduce a scheme for proving that for two vectors $\vec{a}, \vec{b} \in \mathbb{F}^k$ encoded as polynomials $a(X), b(X)$ it is true that $\vec{a} \cdot \vec{b} = \sigma$. Importantly, our scheme uses polynomial encodings of both vectors, but in the case of vector $\vec{a}$ the encoding is *normalized*, in particular, $a(X) = \sum_{i=1}^{k} a_i \hat{\tau}_i(X) = \sum_{i=1}^{k} a_i \frac{\tau_i(X)}{\tau_i(0)}$ for the set of Lagrange interpolation polynomials corresponding to some set $\mathbb{H}_I$ of size k. This is because we will instantiate our inner product argument setting $a(X)$ to be polynomial $D(X)$ from the previous section and $D(X) = \sum_{i=1}^{k} \frac{d_i}{\tau_i(0)} \tau_i(X)$, where $\vec{d}$ is a vector in the rowspace of the lookup matrix $\mathbf{M}$.

Formally, we build a proof system for the following relation:

$$\mathbb{R}_{\mathsf{ginnerprod}} = \left\{ \ (a(X), b(X), z_I(X)) \ ; \ (\vec{a}, \ \vec{b}, \ \mathbb{H}_I) \ \middle| \quad a_i = a(\xi_i)\tau_i(0)^{-1}, \ b_i = b(\xi_i) \ \text{ and } \ \sum_{i=1}^{k} a_i b_i = \sigma \right\}$$

for $\mathbb{H}_I = \{\xi_i\}_{i=1}^{k}$ some fixed set of known size k and $\{\tau_i(X)\}_{i=1}^{k}$ its Lagrange interpolation polynomials. Let $z_I(X)$ be the vanishing polynomial of $\mathbb{H}_I$. Inspired in the linear check of Aurora [BCR+19], we compute an encoding $\sum_{i=i}^{k} a_i b_i \hat{\tau}_i(X)$ of the Hadamard product between $\vec{a}, \vec{b}$ and use a univariate sumcheck argument to obtain the inner product from it. Importantly, since $\mathbb{H}_I$ may contain any set of distinct points that do not necessarily form a multiplicative group, we instantiate our inner product argument with the generalized sumcheck in Section 3.5.

The intuition is that for all Lagrange interpolation polynomials $\{\lambda_j(X)\}_{j=1}^{N}$ corresponding to a multiplicative subgroup $\mathbb{H}$ of size N, we have $\lambda_j(0) = N^{-1}$. Then, for any polynomial $p(X)$, to prove that $\sum_{j=1}^{N} p(\omega^j) = \sigma$, we note that $p(X) = \sum_{j=1}^{N} p(\omega^j)\lambda_j(X) \mod z_H(X)$, and the latter polynomial evaluates to $\sum_{j=1}^{N} p(\omega^j)\lambda_j(0) = \sigma N^{-1}$ in 0. When it comes to arbitrary sets $\mathbb{H}_I$, the corresponding Lagrange polynomials $\{\tau_i(X)\}_{i=1}^{k}$ take different values when evaluated in 0. The generalized sumcheck observes that as soon as one of the encoding polynomials uses *normalized* Lagrange polynomials, that is $\hat{\tau}_i(X) = \frac{\tau_i(X)}{\tau_i(0)}$, the inner product behaves the same way. The protocol is described in Fig. 3.

Committing Phase:

– Set the polynomial

$$z_I(X) = \prod_{\xi \in \mathbb{H}_I} (X - \xi)$$

Proving Phase: $\mathcal{P}_{\mathsf{IP}}$ computes $R(X), Q(X)$ such that

$$a(X)b(X) - \sigma = XR(X) + z_I(X)Q(X).$$

It outputs $\big(R(X), Q(X)\big)$ and a proof that $z_I(X)$ has distinct k roots.

Decision Phase: $\mathcal{V}_{\mathsf{IP}}$ accepts if and only if (i) $\deg(R(X)) < k - 1$, and

$$a(X)b(X) = XR(X) + z_I(X)Q(X),$$

and (ii) $z_I(X)$ has distinct k roots.

Fig. 3. PHP for a generalised inner product argument. As before, well-formation of $z_I(X)$ is given by Caulk+ core.

Theorem 5 (Inner Product Polynomial Relation). *The argument in Fig. 3 is a statistically sound PHP for the relation* $\mathbb{R}_{\mathsf{ginnerprod}}$.

Proof. Let ξ_i, $i = 1, \ldots, k$ be the roots of $z_I(X)$. If we define $a_i = a(\xi_i)\tau_i(0)^{-1}$ and $b_i = b(\xi_i)$, we can represent $a(X) = \sum_{i=1}^{k} a_i\tau_i(0)^{-1}\tau_i(X) + z_I(X)q_1(X)$ and $b(X) = \sum_{j=1}^{k} b_j\tau_j(X) + z_I(X)q_2(X)$. Then there exists $q_3(X)$ such that

$$a(X)b(X) = \left(\sum_{i=1}^{k} a_i\tau_i(0)^{-1}\tau_i(X)\right)\left(\sum_{j=1}^{k} b_j\tau_j(X)\right) + z_I(X)q_3(X).$$

We recall that for $i \neq j$, $z_I(X)|\tau_i(X)\tau_j(X)$ and also $z_I(X)|\tau_i^2(X)$ for all $i \in [k]$. Hence, there exists $q_4(X)$ such that

$$a(X)b(X) = \sum_i a_i b_i \tau_i(0)^{-1}\tau_i(X) + z_I(X)q_4(X)$$

Finally observe that

$$\sum_i a_i b_i \tau_i(0)^{-1}\tau_i(X) = \sum_i a_i b_i + XR(X).$$

This is because the left hand side evaluated at 0 is $\sum_i a_i b_i \tau_i(0)^{-1}\tau_i(0) = \sum_i a_i b_i$. for some $XR(X)$ of degree strictly smaller than $z_I(X)$. Putting this together we have that

$$a(X)b(X) = \sum_i a_i b_i + XR(X) + z_I(X)q_4(X)$$

and therefore $\sum_i a_i b_i = \sigma$ if and only if

$$a(X)b(X) - \sigma = XR(X) + z_I(X)Q(X)$$

for some $R(X)$, $Q(X)$ for $R(X)$ of degree $k-2$. □

6 Ba*loo* Full Construction

In this section, we provide our full Ba*loo* construction for proving the relation

$$\mathbb{R}_{\mathsf{lookup}} = \left\{ \mathsf{cm};\ \phi(X) \;\middle|\; \begin{array}{l} \mathsf{cm} = \mathsf{Commit}(\mathsf{srs}, \phi(X)) \\ \forall \nu \in \mathbb{V},\ \phi(\nu) \in \{c_s\}_{s=1}^{N} \end{array} \right\}$$

where $\mathbb{V}$ is a set of roots of unity that is *independent* from N (the size of the lookup table C) and Commit is the KZG commitment algorithm [KZG10]. For simplicity we have omitted $\vec{c}$ and $\mathbb{V}$ from the relation description. The prover for the full construction is formally given in Fig. 5 and the verifier is given in Fig. 6 with all optimisations included.

Before describing the full construction, we describe the protocol for performing commit-and-prove lookups. Ba*loo* is the result of compiling our building blocks into a succinct proof and use it after the Caulk+ core protocol in Section 3.4. The compiled subprotocol is given in Fig. 4. In other words we describe a protocol for proving the relation

$$\mathbb{R}_{\mathsf{cp\text{-}expansion}} = \left\{ \mathsf{cm};\ \phi(X) \;\middle|\; \begin{array}{l} \mathsf{cm} = \mathsf{Commit}(\mathsf{srs}, \phi(X)) \\ \forall \nu \in \mathbb{V},\ \phi(\nu) \in \{t_i\}_{i=1}^{k} \end{array} \right\}$$

with respect to some table $\{t_i\}_{i=1}^{k}$ that is potentially unknown to the verifier. For simplicity we have omitted $\vec{t}$ and $\mathbb{V}$ from the relation description.

The commit and prove lookup takes as input commitments t, cm to vectors $\vec{t} \in \mathbb{F}^k, \vec{a} \in \mathbb{F}^m$ encoded as polynomials $t(X)$, $\phi(X)$. It also takes as input $[z_I]_2$ a commitment to the vanishing polynomial respect to a set $\mathbb{H}_I$ of known size k. The polynomial $t(X)$ is computed using the Lagrange interpolation basis corresponding to set $\mathbb{H}_I$. The polynomial $\phi(X)$ is computed using the Lagrange interpolation basis corresponding to subgroup $\mathbb{V}$.

The prover aims to convince the verifier that there exists some mapping $\mathsf{col} : [m] \mapsto [k]$ such that $a_j = t_{\mathsf{col}(j)}$ for all $j \in [m]$. The prover and verifier use as subroutine the CSS from Fig. 2 to agree on an encoding $D(X)$ such that

$$D(X) = \sum_{j=1}^{m} \mu_j(\alpha)\hat{\tau}_{\mathsf{col}(j)}(X)$$

at a random point α. Then the prover and verifier engage in the generalised inner product argument from Fig. 3 so show that, if $d_i = D(\xi_i)\tau_i(0)^{-1}$, and $t(x_i) = t_i$,

$$\sum_{i=1}^{k} d_i t_i = \phi(\alpha).$$

Since $d_i = \sum_{j \in \mathsf{col}^{-1}(i)} \mu_j(\alpha)$,

$$\phi(\alpha) = \sum_{i=1}^{k} d_i t_i = \sum_{i=1}^{k} t_i \sum_{j \in \mathsf{col}^{-1}(i)} \mu_j(\alpha) = \sum_{j=1}^{m} \mu_j(\alpha) t_{\mathsf{col}(j)}$$

we thus have that $s_j = \phi(\nu_j) = t_{\mathsf{col}(j)}$ with overwhelming probability, as required.

6.1 Compilation of the cp-expansion Subprotocol

We compile our PHP into a non-interactive succinct argument following the compiler in [CFF+21], and obtain the protocol in Fig. 4. This proves soundness of our scheme under the q-dlog assumption.

Recall that this protocol is a subroutine of Ba*loo* in Fig. 5 and thus the common inputs to the systems are the commitments to $z_I(X), C(X)$ and $t(X)$. The SRS of the full scheme is $\left(\{[x^s]_{1,2}\}_{s=1}^{N}\right)$, where N is the maximum degree among all polynomials. Prover and Verifier instantiate $\mathcal{P}_{\mathsf{IP}}$ and $\mathcal{V}_{\mathsf{IP}}$ for the PHP of Fig. 3. All oracle polynomials sent by $\mathcal{P}_{\mathsf{IP}}$ are translated into polynomials evaluated (in the source groups) at x. Polynomial equations are checked by the verifier from group elements using pairings. For quadratic checks, the prover must send the commitments to the polynomials in different source groups.

All the openings at one point, as well as the degrees of the opened polynomials, are proven using the KZG polynomial commitment. For degree checks with $\deg(p) = d < N$ and $p(X)$ a polynomial that is never opened, the prover sends a single extra polynomial $\hat{p}(X) = X^{N-d}p(X)$, and the verifier checks one extra pairing equation as explained in Section 3.3.

6.2 The Full Ba*loo* Construction

The final construction Ba*loo* is described in Fig. 5 and Fig. 6. It consists simply of combining the Caulk+ core protocol from Section 3.4 and the cp-expansion argument in Fig. 4. We also apply several efficiency optimisations which are specified below.

Efficiency Optimisations In this section we describe some optimizations that can be applied to the protocol in Fig. 4 in order to achieve a construction with smaller proof size and that requires less work from the verifier.

Opening t polynomials in one point. As noted in [GWC19],[CHM+20], whenever we have many openings of different polynomials at the same point, the prover can provide a joint proof after receiving a random element $\gamma \in \mathbb{F}$ from the verifier, i.e., if

$$(u_1, [w_1]_1) \leftarrow \mathsf{KZG.Open}(\mathsf{srs}_{\mathsf{KZG}}, f_1(X), \deg = d, \alpha)$$
$$(u_2, [w_2]_1) \leftarrow \mathsf{KZG.Open}(\mathsf{srs}_{\mathsf{KZG}}, f_2(X), \deg = d, \alpha)$$

then $[w]_1 = [w_1]_1 + \gamma[w_2]_2$ is a proof that $f_1(X) + \gamma f_2(X)$ opens to $u_1 + \gamma u_2$ at α.

Common input: $\mathsf{t} = [t(x)]_1, \mathsf{cm} = [\phi(x)]_1$, $[z_I]_2 = [z_I(x)]_2$ and $\mathsf{srs} = \left\{\{[x^s]_{1,2}\}_{s=1}^{N}\right\}$

$\mathsf{Prover}_{\mathsf{cp\text{-}e}}$**:**

– Compute $v(X) = \sum_{j=1}^{m} \xi_{\mathsf{col}(j)}^{-1} \mu_j(X)$,

– Output $\pi_1 = \big([v]_2 = [v(x)]_2, \big)$.

$\mathsf{Verifier}_{\mathsf{cp\text{-}e}}$**:** Send $\alpha \in \mathbb{F}$

$\mathsf{Prover}_{\mathsf{cp\text{-}e}}$**:**

– Compute $D(X) = M(X, \alpha) = \sum_{j=1}^{m} \mu_j(\alpha) \hat{\tau}_{\mathsf{col}(j)}(X)$ and find $R(X), Q_2(X)$ such that

$$D(X)t(X) - \phi(\alpha) = XR(X) + z_I(X)Q_2(X)$$

– Set $\hat{R} = X^{N-m+2}$

– Output $\pi_2 = \big([D]_2 = [D(x)]_2, [R]_1 = [R(x)]_1, [\hat{R}]_1 = [\hat{R}(x)]_1, [Q_2]_1 = [Q_2(x)]_1\big)$.

$\mathsf{Verifier}_{\mathsf{cp\text{-}e}}$**:** Send $\beta \in \mathbb{F}$

$\mathsf{Prover}_{\mathsf{cp\text{-}e}}$**:**

– Compute $E(X) = M(\beta, X) = \sum_{j=1}^{m} \mu_j(X) \hat{\tau}_{\mathsf{col}(j)}(\beta)$ and $Q_1(X)$ such that

$$E(X)(\beta v(X) - 1) + z_I(\beta) z_I(0)^{-1} = z_V(X) Q_1(X)$$

– $\big(u_1, [w_1]_1\big) \leftarrow \mathsf{Open.KZG}(\mathsf{srs}_{\mathsf{KZG}}, [E]_1, \deg = m-1, \alpha)$

– $\big(u_2, [w_2]_1\big) \leftarrow \mathsf{Open.KZG}(\mathsf{srs}_{\mathsf{KZG}}, \mathsf{cm}, \deg = \bot, \alpha)$

– $\big(u_3, u_4, [w_3]_1\big) \leftarrow \mathsf{Open.KZG}(\mathsf{srs}_{\mathsf{KZG}}, [z_I]_2, \deg = \bot, (0, \beta))$

– $\big(u_5, [w_4]_1\big) \leftarrow \mathsf{Open.KZG}(\mathsf{srs}_{\mathsf{KZG}}, [D]_2, \deg = \bot, \beta)$

– Output $\pi_3 = \big([E]_1 = [E(x)]_1, [Q_1]_1 = [Q_1(x)]_1, (u_1, [w_1]_1), (u_2, [w_2]_1), (u_3, u_4, [w_3]_1), (u_5, [w_4]_1)\big)$.

$\mathsf{Verifier}_{\mathsf{cp\text{-}e}}$**:** Accept if and only if

$$(i) \quad e\left(\mathsf{t}, [D]_2\right) - e\left([1]_1 u_2, [1]_2\right) = e\left([R]_1, [x]_2\right) + e\left([Q_2]_1, [z_I]_2\right)$$

$$(ii) \quad e\left([E]_1, (\beta[v]_2 - 1)\right) + e\left([1]_1(1 - u_3^{-1}u_4), [1]_2\right) = e\left([Q_1]_1, [z_V(x)]_2\right)$$

$$(iii) \quad e([R]_1, [x^{N-m+2}]_2) = e([\hat{R}]_1, [1]_2)$$

$$(iv) \quad u_1 = u_5.$$

$$(v) \quad 1 \leftarrow \mathsf{KZG.Verify}(\mathsf{srs}_{\mathsf{KZG}}, [E]_1, \deg = m-1, \alpha, u_1, [w_1]_1)$$

$$(vi) \quad 1 \leftarrow \mathsf{KZG.Verify}(\mathsf{srs}_{\mathsf{KZG}}, \mathsf{cm}, \deg = \bot, \alpha, u_2, [w_2]_1)$$

$$(vii) \quad 1 \leftarrow \mathsf{KZG.Verify}(\mathsf{srs}_{\mathsf{KZG}}, [z_I]_2, \deg = \bot, (0, \beta), (u_3, u_4), [w_3]_1)$$

$$(viii) \quad 1 \leftarrow \mathsf{KZG.Verify}(\mathsf{srs}_{\mathsf{KZG}}, [D]_2, \deg = \bot, \beta, u_5, [w_4]_1)$$

Fig. 4. cp-expansion argument for proving $\phi(X)$ has entries in a (potentially unknown) subtable $t(X)$.

Openings for Pairings. To save the verifier some work, we use a technique introduced in [GWC19] and attributed to M. Maller. In order to verify that $a(X)b(X) = c(X)d(X)$ for $a(X), b(X), c(X), d(X)$ the algebraic representations of $[a]_1, [b]_2, [c]_1, [d]_2$, instead of asking the verifier to check that

$$e([a]_1, [b]_2) = e([c]_1, [d]_2),$$

we ask the prover to show that $[b]_2, [d]_2$ open to u_1, u_2 at β and that $u_1[a]_1 - u_2[c]_1$ opens to zero at β. Note that now the prover can also commit to $b(X)$ and $d(X)$ in $\mathbb{G}_1$ instead of $\mathbb{G}_2$. We apply this technique to the equations that verify the inner product relation and the well formation of $[E]_1$; that is, equations (i) and (ii). Note that we can open this equations together with other elements. Indeed, we will check equation (i) by opening a polynomial $[P_1]_1$ at β, and batch that KZG opening together with the one for $[D]_1$.

Degree checks. Degree checks as $\deg(f) \leq k < d$ can be included in a KZG proof that $f(\alpha) = u$ if the prover sets $\hat{w}(X) = \frac{f(X)-f(\alpha)}{X-\alpha}$, outputs $(u, [w]_1 = [\hat{w}(x)x^{d-k+1}])$ and the verifier checks

$$e\big([f]_1 - [u]_1, [x^{d-k+1}]_2\big) = e\big([w]_1, [x-\alpha]_2\big),$$

as explained in Section 3.3. This is conditional on α being randomly chosen after $f(X)$.

Throughout Ba*loo* we require 3 degree checks: (i) that $\deg(E(X)) < m-1$, (ii) that $\deg(z_I(X)) = m$, and (iii) that $\deg(R(X)) = m-2$. For (i) we check via a KZG opening that $E(X)$ has bounded degree. For (ii) we check that $z_I(X) - X^m$ has degree bounded by $m-1$ during our opening check that $z_I(0)$ is correct. Degree bounding $f(X) < k$ via an opening at 0 checks that

$$e\big([f]_1 - [u]_1, [1]_2\big) = e\big([w]_1, [x]_2\big) \text{ and } e\big([f]_1, [x^{d-k+2}]_2\big) = e\big([w]_2, [x]_2\big),$$

because 0 is not a random point.

For (iii) we recall that the polynomial $R(X)$ is sent for the inner product relation to show that $a(X)b(X) - \sigma = XR(X) + z_I(X)Q(X)$. In our optimised protocol we instead send $\bar{R}(X) = XR(X)$ and show that $\bar{R}(0) = 0$ and that $\bar{R}(X)$ has degree bounded by $m-1$. We then show that $a(X)b(X) - \sigma = \bar{R}(X) + z_I(X)Q(X)$. This is equivalent because $\bar{R}(0) = 0$ if and only if $\bar{R}(X) = XR(X)$. Where we can batch this check with opening and degree bounding $z_I(X)$ at the same point (namely 0) and with the same degree $(m-1)$, this check is essentially free.

Batching Pairings. We also apply standard techniques to batch pairings that share the same elements in one of the two groups. Namely, upon sampling a uniform $\gamma_2 \in \mathbb{F}$, the verifier can aggregate the equations

$$e([a]_1, [b_1]_2) = e([c_1]_1, [d]_2) \text{ and } e([a]_1, [b_2]_2) = e([c_2]_1, [d]_2),$$
$$\text{as } e([a]_1, [b_1 + \gamma b_2]_2) = e([c_1 + \gamma c_2]_1, [d]_2)$$

Note that we can adapt KZG openings equations so they can be batched further, namely if we parse the verification pairing as $e\big([f]_1 - u + [w]_1\alpha, [1]_2\big) = e\big([w]_1, [x]_2\big)$, then two openings of different polynomials at different points can be verified by two pairings.

Finally, note that in order to check $E(\alpha) = D(\beta)$, the proves needs to provide proof of both openings but can only send $u_2 = E(\alpha)$ and the verifier checks $D(\beta)$ opens to u_2 as well.

7 Ba*loo* Prover Cost

In this section we elaborate on the Prover's computational costs while showing that those are quasilinear in m.

Common input: $\mathsf{C} = [C(x)]_1, [z_H(x)]_1 = [\prod_{i=0}^{N-1}(x-\omega^i)]_1$, $\mathsf{srs} = \{[x^i]_{1,2}\}_{i=1}^{d}$

1. Take as input $\{[Q_i(x)]_1, [H_i(x)]_1\}_{i=1}^{N}$ and $\phi(X)$
2. Choose $I \subset [N]$ such that $|I| = k$ and $\forall \nu \in \mathbb{V}$, $\exists \xi \in \mathbb{H}_I$ s.t. $\phi(\nu) = C(\xi)$
3. Compute $v(X) = \sum_{j=1}^{m} \xi_{col(j)}^{-1} \mu_j(X)$
4. Output $\pi_1 = \big([z_I]_2 = [z_I(x)]_2, [v]_1 = [v(x)]_1, \mathsf{t} = [t(x)]_1\big)$.
5. Receive $\alpha \in \mathbb{F}$
6. Compute $D(X) = M(X, \alpha) = \sum_{j=1}^{m} \mu_j(\alpha) \hat{\tau}_{\mathsf{col}(j)}(X)$
7. Find $R(X), Q_2(X)$ such that $\deg(R(X)) < m-1$, $R(0) = 0$, and
$$D(X)t(X) - \phi(\alpha) = R(X) + z_I(X)Q_2(X)$$
8. Output $\pi_2 = \big([D]_1 = [D(x)]_1, [R]_1 = [R(x)]_1, [Q_2]_1 = [Q_2(x)]_1, \big)$.
9. Receive $\beta \in \mathbb{F}$
10. Compute $E(X) = M(\beta, X) = \sum_{j=1}^{m} \mu_j(X) \hat{\tau}_{\mathsf{col}(j)}(\beta)$ and $Q_1(X)$ such that
$$E(X)(\beta v(X) - 1) + z_I(\beta) z_I(0)^{-1} = z_V(X) Q_1(X)$$
11. Output $\pi_3 = \big([E]_1 = [E(x)]_1, [Q_1]_1 = [Q_1(x)]_1\big)$
12. Receive $\rho, \gamma \in \mathbb{F}$
13. Compute $([a_1]_1, [a_2]_1, \_) \leftarrow \mathsf{Prover}_{\mathsf{C+}}(t(X), \mathbb{H}_I)$ and set $[a]_1 = [a_1]_1 + \gamma[a_2]_2$
    # Compress Caulk+ proof.
14. Set $u_1 = E(\alpha)$, $u_2 = \phi(\alpha)$,
$$\hat{w}_1(X) = \frac{E(X) - u_1}{X - \alpha} + \gamma \frac{\phi(X) - u_2}{X - \alpha}$$
    # Prove that $E(\alpha) = u_1$, $\phi(\alpha) = u_2$, $\deg(E(X)) < m$
15. Set $u_3 = z_I(0)$ and
$$w_2(X) = \frac{z_I(X) - u_3}{X} + \gamma \frac{R(X)}{X} + \gamma^2 X^{d-m+1}(z_I(X) - X^m) + \gamma^3 X^{d-m+1} R(X)$$
    # Prove that $z_I(0) = u_3$, $R(0) = 0$, $\deg(z_I(X)) = m$ and $\deg(R(X)) < m$
16. Set $P_1(X) = t(X)D(\beta) - \phi(\alpha) - R(X) - z_I(\beta)Q_2(X)$, $u_4 = z_I(\beta)$ and
$$w_3(X) = \frac{D(X) - u_1}{X - \beta} + \gamma \frac{z_I(X) - u_4}{X - \beta} + \gamma^2 \frac{P_1(X)}{X - \beta}$$
    # Prove that $D(\beta) = E(\alpha)$, $z_I(\beta) = u_4$, $t(X)D(X) - \phi(\alpha) = R(X) + z_I(X)Q_2(X)$
17. Set $u_5 = E(\rho)$, $P_2(X) = E(\rho)(\beta v(X) - 1) + z_I(\beta) z_I(0)^{-1} - z_V(\rho) Q_1(X)$,
$$w_4(X) = \frac{E(X) - u_5}{(X - \rho)} + \gamma \frac{P_2(X)}{X - \rho}$$
    # Prove that $E(X)(\beta v(X) - 1) + z_I(\beta) z_I(0)^{-1} = z_V(X) Q_1(X)$
18. Set $s = d - m + 1$ for d the maximum power of x in srs and output
$$\pi_3 = \big(u_1, u_2, u_3, u_4, u_5, [a]_1, [w_1]_1 = [\hat{w}_1(x)x^s]_1, [w_2]_1 = [w_2(x)]_1, [w_3]_1 = [w_3(x)]_1, [w_4]_1 = [w_4(x)]_1\big).$$

Fig. 5. Optimized Ba*loo* prover. Underlined steps are messages from Verifier (Fig. 6).

Common input: $\mathsf{C} = [C(x)]_1, [z_H(x)]_1 = [\prod_{i=0}^{N-1}(x-\omega^s)]_1$, $\mathsf{srs} = \{[x^i]_{1,2}\}_{i=1}^{d}$

Take cm as input.

Receive $\pi_1 = \big([z_I]_2, [v]_1, [\mathsf{t}]_1\big)$ and send $\alpha \in \mathbb{F}$

Receive $\pi_2 = \big([D]_1, [R]_1, [Q_2]_1\big)$ and send $\beta \in \mathbb{F}$

Receive $\pi_3 = \big([E]_1, [Q_1]_1\big)$ and send $\rho, \gamma \in \mathbb{F}$

Receive $\pi_4 = (u_1, u_2, u_3, u_4, u_5, [a]_1, [w_1]_1, [w_2]_1, [w_3]_1, [w_4]_1)$

Compute

$$[P_1]_1 = u_1[\mathsf{t}] - u_2[1]_1 - [R]_1 - u_4[Q_2]_1$$

\# $[P_1]_1 = [t(x)D(\beta) - \phi(\alpha) - R(x) - z_I(\beta)Q_2(x)]_1$

$$[P_2]_1 = u_5(\beta[v]_1 - 1) + u_3^{-1}u_4 - z_V(\rho)[Q_1]_1$$

\# $[P_2]_1 = [E(\rho)(\beta v(x) - 1) + z_I(\beta)z_I(0)^{-1} - z_V(\rho)Q_1(x)]_1$

Set $s = d - m + 1$ for d the maximum power of x in srs and accept if and only if

1. $e((\mathsf{C} - \mathsf{t}) + \gamma[z_H(x)]_1, [1]_2) - e([a]_1, [z_I]_2) = 0$
   \# Check that C+ elements $[a_1]_1$ and $[a_2]_1$ verify.

2. $e(\alpha[w_1]_1, [1]_2) - e([w_1]_1, [x]_2) + e([E]_1 + \gamma\mathsf{cm} - [u_1 + \gamma u_2]_1, [x^s]_2) = 0$
   \# Check that $E(\alpha) = u_1$, $\phi(\alpha) = u_2$, $\deg(E(X)) < m$

3. $e(-[u_3]_1 + \gamma[R]_1, [1]_2) - e([w_2]_1, [x]_2) + e([1 + \gamma^2 x^{s+1}]_1, [z_I]_2) + e(-\gamma^2[x^m]_1 + \gamma^3[R]_1, [x^{s+1}]_2) = 0$
   \# Check that $z_I(0) = u_3$, $R(0) = 0$, $\deg(z_I(X)) = m$ and $\deg(R(X)) < m$

4. $e(\beta[w_3]_1 + [D]_1 + \gamma^2[P_1]_1 - [u_1 + \gamma u_4]_1, [1]_2) - e([w_3]_1, [x]_2) + e([\gamma]_1, [z_I]_2) = 0$
   \# Check that $D(\beta) = E(\alpha)$, $z_I(\beta) = u_4$, $t(X)D(X) - \phi(\alpha) = R(X) + z_I(X)Q_2(X)$

5. $e(\rho[w_4]_1 + [E]_1 + \gamma[P_2]_1 - [u_5]_1, [1]_2) - e([w_4]_1, [x]_2) = 0$
   \# Check that $E(\rho) = u_5$ and $E(X)(\beta v(X) - 1) + z_I(\beta)z_I(0)^{-1} = z_V(X)Q_1(X)$

These checks can be batched into 1 equation with 5 pairings.

Fig. 6. Optimized Ba*loo* verifier.

7.1 Generic algorithms

An excellent survey of various algorithms for polynomials with pseudocode is given in [vzGG13]. Let $\mathbb{F}$ be a domain with Fast Fourier Transform of size N. Polynomials in $\mathbb{F}[X]$ are considered as vectors of coefficients in the standard basis $\{1, X, X^2, \ldots, X^N\}$ unless stated otherwise. The set I does not support FFTs. The computational costs are counted in operations in $\mathbb{F}$. We are using the following basic results (everywhere $d < N$):

- Multiplication: two polynomials of degree d can be multiplied in $O(d \log d)$ time.
- Inversion: given a polynomial f of degree d can be inverted modulo X^ℓ, $\ell > d$, in $O(d \log d)$ time.
- Division: a polynomial f of degree d can be divided with a remainder by a polynomial g of degree $d' < d$ in $O(d \log d)$ time, i.e. we can find $q(X), r(X)$ of degree $d'' < d$ such that

$$f(X) = q(X)g(X) + r(X)$$

- Vanishing polynomial: a polynomial $z_I(X)$ that vanishes on set I of size d can be computed in $O(d \log^2 d)$ time.
- Evaluation: a polynomial f of degree d can be evaluated in d points in $O(d \log^2 d)$ time.
- Interpolation: a polynomial f of degree d with values c_i at points x_i, $0 \le i \le d$, can be computed (interpolated) in $O(d \log^2 d)$ time.

7.2 Prover costs in Ba*loo*

For simplicity we assume that $m = k$.

Aggregation of individual proofs . The subset opening proofs for the set of points $\mathbb{H}_I \subseteq \mathbb{H}$ are computed as

$$[H]_1 = \sum_{i \in I} \left(\prod_{s \in I, s \neq i}^{m} \frac{1}{(\omega^i - \omega^s)} \right) [H_i]_1$$

The coefficients $r_i = \left(\prod_{s \in I, s \neq i}^{m} \frac{1}{(\omega^i - \omega^s)} \right)$ are altogether computed in $O(m \log^2 m)$ time as follows. Let $Z'_I(X)$ be the derivative of $Z_I(X)$ then we have $r_i = \frac{1}{Z'_I(\omega^i)}$ [vzGG13, p. 300]. We use a vanishing polynomial reconstruction algorithm (see above) and symbolically compute $Z'_I(X)$ in $O(m \log^2 m)$ time. Then we evaluate $Z'_I(X)$ over I also in $O(d \log^2 d)$ time. Thus $[H]_1$ can be computed in m group operations.

Running time of Caulk+ core. The costs of Fig. 5 break down as follows:

- Polynomial $\frac{C(x) - t(x)}{z_I(x)}$ similarly to $[H]_1$ using $O(m \log^2 m)$ field operations. Then it takes m group operations to compute W_1.
- The group element W_2 is computed as a linear combination of $[H_i(x)]_1$ as in [ZBK+22] in time m group and $O(k \log^2 k)$ field operations (see above).
- The polynomial in W_3 has $O(m)$ nonzero coefficients and thus needs at most m group operations to be computed.

In Fig. 5 the element W_3 is unused and this computation can be omitted. Overall we need $2m$ group operations and $O(m \log^2 m)$ field operations.

Running time of cp-expansion *argument Fig. 4* We first note that Lagrange polynomials $\mu_j(X)$ and $\tau_i(X)$ have succinct form. Concretely we have, assuming $|\mathbb{V}| = m$:

$$\mu_j(X) = \frac{X^m - 1}{m\nu^{-j}(X - \nu^j)}, \quad \tau_i(X) = \frac{z_I(X)}{z_I'(\xi_i)(X - \xi_i)} = \frac{r_i z_I(X)}{(X - \xi_i)}$$

where $z_I'(X)$ is the derivative of $z_I(X)$. All $\mu_j(X)$ can be batch-evaluated in m points in $m \log m$ time as one evaluation is $\log m$ time. For $\tau_i(X)$ we compute $r_i = \frac{1}{z_I'(\xi_i)}$ using the evaluation algorithm for $z_I(X)$ in $O(m \log^2 m)$ time, and then inverting in $m \log(m)$ time. Then in order to batch-evaluate all $\tau_i(X)$ at some point β, we evaluate $z_I(X)$ at β in m time and each $\frac{r_i}{\beta - \xi_i}$ in $\log m$ time. These costs are all in $\mathbb{F}$.

The field operation costs of Fig. 4 break down as follows:

- Polynomial $v(X)$ has degree m and can be computed via interpolation in $O(m \log^2 m)$.
- Polynomial $D(X)$ is computed by interpolation as follows. We first batch-evaluate $\mu_j(X)$ at α in $O(m \log m)$ time. Then we batch-evaluate $\tau_i(X)$ at 0 in $O(m \log^2 m)$ time, so that we know all coefficients of $\tau_{\mathsf{col}(j)}(X)$ in the sum. Those coefficients are exactly values of $D(X)$ at ξ_i. From those we interpolate $D(X)$ in $O(m \log^2 m)$ time.
- Polynomials $R(X), Q_1(X), Q_2(X)$ can be computed using the division algorithm (above) in $O(m \log^2 m)$ time
- Polynomial $E(X)$ is computed by interpolation again. As for $D(X)$ we batch-evaluate all $\tau_i(X)$ at β so that we know all coefficients of $\mu_j(X)$ in $O(m \log^2 m)$ time. As $\mu_j(X)$ are defined over a subgroup of roots of unity, the interpolation of $E(X)$ is in $O(m \log m)$ time.

Computing $[z_I(x)]_2$ takes m $\mathbb{G}_2$ operations. Computing the 12 $\mathbb{G}_1$ elements that Prover sends takes $11m$ $\mathbb{G}_1$ operations as those commitments are either to polynomials of degree m (those are $t, v, D, E, Q_1, R, Q_2, w_3, w_4$) or have at most m non-zero coefficients ($[w_1]_1, [w_2]_1$), or need constant time ($[a]_1$). Overall we need $11m$ $\mathbb{G}_1$ operations, m $\mathbb{G}_2$ operations, and $O(m \log^2 m)$ field operations.

Full running cost By summing up the prover costs for Caulk+ core and cp-expansion we get that the total Prover cost in Ba*loo* is $13m$ $\mathbb{G}_1$ operations, m $\mathbb{G}_2$ operations and $O(m \log^2 m)$ field operations.

8 Faster SNARKs using Ba*loo*

Commit-and-prove lookup tables are especially suitable for the Ethereum Foundation's zero-knowledge Ethereum Virtual Machine (zkEVM), which nowadays uses Halo2 with KZG commitments as a backend. The zkEVM is an effort to outsource the evaluation of the Ethereum Virtual Machine on some chosen inputs to a powerful prover, and then demonstrate that the result is correct to a computationally constrained verifier. Importantly, what the zkEVM aims to is a succinct proof of validity of the blocks, and since they are public, no zero-knowledge is required. In other words, the proofs are sound but not zero-knowledge, and Ba*loo* can be safely used.

In this section we describe an overview of how lookups are currently used in the Halo2 proving system [BGH20] and claim Ba*loo* can be used as a drop in replacement to the Halo2 lookup argument with better prover efficiency. In other words Ba*loo* is backwards compatible with instantiations of Halo2 that use KZG commitments.

Ba*loo* is a proving system for the relation that

$$\mathbb{R}_{\mathsf{lookup}} = \left\{ \begin{array}{l|l} \mathsf{cm};\ \phi(X) & \begin{array}{l} \mathsf{cm} = \mathsf{Commit}(\mathsf{srs}, \phi(X)) \\ \forall \nu \in \mathbb{V},\ \phi(\nu) \in \{c_s\}_{s=1}^N \end{array} \end{array} \right\}$$

where $\mathbb{V}$ is a set of roots of unity that is *independent* from N (the size of the lookup table C) and Commit is the KZG commitment algorithm [KZG10]. This lookup argument only handles a single column. Suppose instead that we want to prove $f(x) = y$ by precomputing all possible values $T = \{(x_s, f(x_s))\}_{s=1}^N$ and looking up whether $(x, y) \in T$. To achieve this we require more functionality from our lookup argument. In particular we need to be able to prove a lookup argument over *multicolumned* tables.

Multi-column Baloo We build on the Halo2 approach [6].

Given a lookup with input column polynomials $[\phi_0(X), \ldots, \phi_{k-1}(X)]$ and a multi-columned table of the form $C = \{c_{i,j}\}_{i,s=1}^{i=k,s=N}$, their prover shows that for all $\nu \in \mathbb{V}$, there exists some s such that $(\phi_0(\nu), \ldots, \phi_{k-1}(\nu)) = (c_{0,s}, \ldots, c_{k-1,s})$. It does this by taking a random linear combination of the input column polynomials and the table columns and then running a lookup argument over the compressed values. We present a similar compression for Ba*loo* such that we can run lookups over multi-columned tables.

Similarly than in Caulk+, described in Section 3.4, the pre-processing phase commits to the table $\{c_{i,s}\}_{i,s=1}^{k,N}$ by committing to the column polynomials

$$C_s(X) = \sum_{s=1}^{N} c_s \lambda_s(X)$$

for $\{\lambda_s(X)\}_{s=1}^{N}$ a set of roots of unity $H = \{\omega^s\}_{s=1}^{N}$ of size N. The pre-processing phase also computes auxiliary information for the prover, namely it computes commitments to the polynomials

$$\{Q_{i,s}(X) = (C_s(X) - C_s(\omega_j))/(X - \omega^s)\}_{i,s=1}^{k,N}, H_s(X) = \{z_H(X)/(X - \omega^s)\}_{s=1}^{N}$$

in time $kN \log_2(N)$.

Given the input column polynomials $[\phi_0(X), \ldots, \phi_{k-1}(X)]$ we sample a random value θ. Suppose that I is the set of points such that $j \in I$ if and only if $(c_{0,s}, \ldots, c_{k-1,s})$ appears in the input columns i.e. $(c_{0,s}, \ldots, c_{k-1,s}) \in \{(\phi_0(\nu), \ldots, \phi_{k-1}(\nu))\}_{\nu \in \mathbb{V}}$. Then the prover commitments to the compressed polynomials

$$\begin{aligned}
\phi_{\mathsf{compressed}}(X) = \phi_0(X) + \theta\phi_1(X) \ldots + \theta^{k-1}\phi_{k-1}(X) \quad &= \sum_{j=1}^{m} \left(\sum_{i=0}^{k} \theta^i \phi_i(\nu_j) \right) \mu_j(X) \\
C_{\mathsf{compressed}}(X) = C_0(X) + \theta C_1(X) \ldots + \theta^{k-1} C_{k-1}(X) \quad &= \sum_{s=1}^{N} \left(\sum_{i=0}^{k} \theta^i c_{i,s} \right) \lambda_s(X) \\
\text{for } s \in I,\ Q_{\mathsf{compressed},j} = Q_{0,s}(X) + \theta Q_{1,s}(X) \ldots + \theta^{k-1} Q_{k-1,s}(X) \quad &= \left(\sum_{i=0}^{k-1} \theta^i C_i(X) - \theta^i C_i(\omega^s)) \right) /(X - \omega^s)
\end{aligned}$$

Then we have that $C_{\mathsf{compressed}}(X)$ describes the table $C_{\mathsf{compressed}} = \{\sum_{i=0}^{k-1} \theta^i c_{i,s}\}_{s=1}^{N}$. The randomiser θ is sampled from a large field. Thus the probability that $\sum_{i=0}^{k} \theta^i \phi_i(\nu) \in C_{\mathsf{compressed}}$ is negligible unless there exists some s such that $(\phi_0(\nu), \ldots, \phi_{k-1}(\nu)) = (c_{0,s}, \ldots, c_{k-1,s})$. The auxiliary information that the prover requires for efficiency, namely commitments to $\{H_s(X), Q_{\mathsf{compressed},s}(X)\}_{s \in I}$, can be computed in km time where $m = |\mathbb{V}|$ is the number of lookups.

Thus for $\mathsf{cm}_{\mathsf{compressed}}$ a commitment to $\phi_{\mathsf{compressed}}(X)$, the prover demonstrates that $\mathsf{cm}_{\mathsf{compressed}} \in R_{\mathsf{lookup}}$ with respect to the table $C_{\mathsf{compressed}}$.

References

AR20. Shashank Agrawal and Srinivasan Raghuraman. Kvac: Key-value commitments for blockchains and beyond. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 839–869. Springer, 2020. 3

BaCCL21. Olivier Bégassat, Alexandre Belling andThéodore Chapuis-Chkaiban, and Nicolas Liochon1. A specification for a zk-evm, 2021. `https://ethresear.ch/t/a-zk-evm-specification/11549`. 1

[6] `https://zcash.github.io/halo2/design/proving-system/circuit-commitments.html`

BB04. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004. 3

BCC+15. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*, volume 9326 of *Lecture Notes in Computer Science*, pages 243–265. Springer, 2015. 2

BCF+21. Daniel Benarroch, Matteo Campanelli, Dario Fiore, Kobi Gurkan, and Dimitris Kolonelos. Zero-knowledge proofs for set membership: Efficient, succinct, modular. In Nikita Borisov and Claudia Díaz, editors, *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I*, volume 12674 of *Lecture Notes in Computer Science*, pages 393–414. Springer, 2021. 3

BCG+13. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for C: verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013. 2

BCG+18. Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune K. Jakobsen, and Mary 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. 2

BCR+19. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *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*, volume 11476 of *Lecture Notes in Computer Science*, pages 103–128. Springer, 2019. 5, 6, 13

BG13. Stephanie Bayer and Jens Groth. Zero-knowledge argument for polynomial evaluation with application to blacklists. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 646–663. Springer, 2013. 2

BG18. Jonathan Bootle and Jens Groth. Efficient batch zero-knowledge arguments for low degree polynomials. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 561–588. Springer, 2018. 2

BGH20. Sean Bowe, Jack Grigg, and Daira Hopwood. Halo2. *URL: https://github. com/zcash/halo2*, 2020. 2, 21

CDGM19. Melissa Chase, Apoorvaa Deshpande, Esha Ghosh, and Harjasleen Malvai. Seemless: Secure end-to-end encrypted messaging with less trust trust. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 1639–1656. ACM, 2019. 2

CEO22. Matteo Campanelli, Felix Engelmann, and Claudio Orlandi. Zero-knowledge for homomorphic key-value commitments with applications to privacy-preserving ledgers. In Clemente Galdi and Stanislaw Jarecki, editors, *Security and Cryptography for Networks - 13th International Conference, SCN 2022, Amalfi, Italy, September 12-14, 2022, Proceedings*, volume 13409 of *Lecture Notes in Computer Science*, pages 761–784. Springer, 2022. 2

CFF+21. Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zksnarks and commit-and-prove extensions. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part III*, volume 13092 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2021. 7, 15

CFG+20. Matteo Campanelli, Dario Fiore, Nicola Greco, Dimitris Kolonelos, and Luca Nizzardo. Incrementally aggregatable vector commitments and applications to verifiable decentralized storage. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 3–35. Springer, 2020. 3

CFH+21. Matteo Campanelli, Dario Fiore, Semin Han, Jihye Kim, Dimitris Kolonelos, and Hyunok Oh. Succinct zero-knowledge batch proofs for set accumulators. *IACR Cryptol. ePrint Arch.*, page 1672, 2021. 2

CFQ19. Matteo Campanelli, Dario Fiore, and Anaïs Querol. Legosnark: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2075–2092. ACM, 2019. 6

CHM+20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zksnarks with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 738–768. Springer, 2020. 8, 15

Eth22. Zkevm introduction, 2022. `https://github.com/privacy-scaling-explorations/zkevm-specs/blob/master/specs/introduction.md`. 1

FK20. Dankrad Feist and Dmitry Khovratovich. Fast amortized kate proofs, 2020. 4

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, 2018. 3

GG17. Essam Ghadafi and Jens Groth. Towards a classification of non-interactive computational assumptions in cyclic groups. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 66–96. Springer, 2017. 3

GK15. Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 253–280. Springer, 2015. 2

GK22. Ariel Gabizon and Dmitry Khovratovich. flookup: Fractional decomposition-based lookups in quasi-linear time independent of table size. *Cryptology ePrint Archive*, 2022. 2

GW20. Ariel Gabizon and Zachary J Williamson. plookup: A simplified polynomial protocol for lookup tables. *Cryptology ePrint Archive*, 2020. 1, 2

GWC19. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptol. ePrint Arch.*, page 953, 2019. 15, 17

HHK+21. Yuncong Hu, Kian Hooshmand, Harika Kalidhindi, Seung Jin Yang, and Raluca Ada Popa. Merkle^2: A low-latency transparency log system. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 285–303. IEEE, 2021. 2

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2010. 2, 3, 14, 21

MKL+20. Sarah Meiklejohn, Pavel Kalinnikov, Cindy S. Lin, Martin Hutchinson, Gary Belvin, Mariana Raykova, and Al Cutter. Think global, act local: Gossip and client audits in verifiable data structures. *CoRR*, abs/2011.04551, 2020. 2

PK22. Jim Posen and Assimakis A. Kattis. Caulk+: Table-independent lookup arguments. Cryptology ePrint Archive, Paper 2022/957, 2022. `https://eprint.iacr.org/2022/957`. 1, 2, 4, 5

Pol22. Polygon zkevm documentation, 2022. `https://docs.hermez.io/zkEVM/Overview/Overview/`. 1

RZ21. Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable snarks. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International*

Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I, volume 12825 of *Lecture Notes in Computer Science*, pages 774–804. Springer, 2021. 5, 6, 7, 8, 9

Sta22. Starknet, 2022. `https://starkware.co/starknet/`. 1

TAB+20. Alin Tomescu, Ittai Abraham, Vitalik Buterin, Justin Drake, Dankrad Feist, and Dmitry Khovratovich. Aggregatable subvector commitments for stateless cryptocurrencies. In Clemente Galdi and Vladimir Kolesnikov, editors, *Security and Cryptography for Networks - 12th International Conference, SCN 2020, Amalfi, Italy, September 14-16, 2020, Proceedings*, volume 12238 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2020. 4

TBP+19. Alin Tomescu, Vivek Bhupatiraju, Dimitrios Papadopoulos, Charalampos Papamanthou, Nikos Triandopoulos, and Srinivas Devadas. Transparency logs via append-only authenticated dictionaries. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 1299–1316. ACM, 2019. 2

TFBT21. Nirvan Tyagi, Ben Fisch, Joseph Bonneau, and Stefano Tessaro. Client-auditable verifiable registries. *IACR Cryptol. ePrint Arch.*, page 627, 2021. 2

vzGG13. Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra (3. ed.)*. Cambridge University Press, 2013. 20

ZBK+22. Arantxa Zapico, Vitalik Buterin, Dmitry Khovratovich, Mary Maller, Anca Nitulescu, and Mark Simkin. Caulk: Lookup arguments in sublinear time. CCS '22: 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, November 7 - 11, 2022, 2022. 1, 2, 3, 4, 5, 20

Zha22. Ye Zhang. Introducing zkevm, 2022. `https://scroll.io/blog/zkEVM`. 1

zks22. zkevm faq, 2022. `https://docs.zksync.io/zkevm/`. 1

A Proof of Thm 1

Proof. Let $\mathcal{A}$ be an algebraic adversary attempting to break knowledge soundness. We design an extractor $\mathcal{E}$ that behaves as follows. When the adversary $(C(X), \mathsf{t}, [z_I]_2, \pi) \leftarrow \mathcal{A}(\mathsf{srs})$ outputs a proof $\pi = (W_1, W_2, W_3)$, then it also outputs the representations $w_1(X), w_3(X)$ of maximum degree N such that

$$W_1 = [w_1(x)]_1, W_3 = [w_3(x)]_1$$

The extractor $\mathcal{E}$ computes

$$z_I(X) = w_3(X)X^{-(N-k+1)} + X^k, t(X) = C(X) - w_1(X)z_I(X)$$

and returns $\mathbb{H}_I, t(X)$ where $\mathbb{H}_I$ consists of the roots of $z_I(X)$.

We show that either $\mathcal{E}$ succeeds with overwhelming probability or we can construct a reductions $\mathcal{B}_1$ and $\mathcal{B}_2$ such that

$$\mathcal{A}^0_{\mathcal{A},\mathcal{E}}(\lambda) \leq \mathcal{A}^{\mathsf{q\text{-}sfrac}}_{\mathcal{B}_1}(\lambda) + \mathcal{A}^{\mathsf{q\text{-}dlog}}_{\mathcal{B}_2}(\lambda)$$

$\underline{\mathsf{Game}^0 \mapsto \mathsf{Game}^1}$: Let Game^0 be the original knowledge soundness game. We first transition to a game Game^1 that behaves identically to Game^0 except that, when $\mathcal{A}$ outputs the representation $w_3(X)$, if $z_I(X) = w_3(X)X^{-(N-k+1)} + X^k$ is not a degree k polynomial (with positive degree monomials only) then Game^1 aborts.

We show the existence of a reduction $\mathcal{B}_1$ such that

$$\mathcal{A}^0_{\mathcal{A},\mathcal{E}}(\lambda) \leq \mathcal{A}^1_{\mathcal{A},\mathcal{E}}(\lambda) + \mathcal{A}^{\mathsf{q\text{-}sfrac}}_{\mathcal{B}_1}(\lambda)$$

The reduction $\mathcal{B}_1$ gets as input srs and forwards this reference string to run $(C(X), \mathsf{t}, [z_I]_2, \pi) \leftarrow \mathcal{A}(\mathsf{srs})$. When $\mathcal{A}$ outputs a proof $\pi = (W_1, W_2, W_3)$, then it also outputs the representation $w_3(X)$ of maximum degree N such that

$$z_I = [w_3(x)x^{-N+k-1} + x^k]_2$$

Write $w_3(X) = \sum_{s=0}^{N} a_s X^s$ Then $\mathcal{B}_1$ returns

$$\sum_{s=0}^{N-k} a_s X^s, X^{N-k+1}, [z_I]_2 - [x^k]_2 - [\sum_{s=N-k+1}^{N} a_s x^s]_2$$

If $a_s \neq 0$ for $0 \leq s \leq N-k$, then the degree of $\sum_{s=0}^{N-k} a_s X^s$ is less than $N-k+1$ and hence $\mathcal{B}_1$ breaks the q-sfrac assumption.

If $a_s = 0$ for all $0 \leq s \leq N-k$ then

$$z_I(X) = w_3(X)X^{-(N-k+1)} + X^k = \sum_{s=0}^{k-1} a_{N-k+1+s}X^s + X^k$$

which is a degree k polynomial.

$\underline{\mathsf{Game}^1 \mapsto \mathsf{Game}^2}$ We second transition to a game Game^2 that behaves identically to Game^1 except that, when $\mathcal{A}$ outputs the representation $w_3(X)$, if $z_I(X) = w_3(X)X^{-(N-k+1)} + X^k$ does not divide $z_H(X)$, then Game^2 aborts. We show the existence of a reduction $\mathcal{B}_2$ such that

$$\mathcal{A}^1_{\mathcal{A},\mathcal{E}}(\lambda) \leq \mathcal{A}^2_{\mathcal{A},\mathcal{E}}(\lambda) + \mathcal{A}^{\mathsf{q\text{-}dlog}}_{\mathcal{B}_1}(\lambda)$$

See that if Game^2 does not abort, then $z_I = [f(X)]$ for some $f(X)$ of degree k that divides $z_H(X)$. This means that $z_I(X) = \prod_{i=1}^{k}(X - \xi_i)$ for $\mathbb{H}_I$ some subset of $\mathbb{H}$ of size k.

The reduction $\mathcal{B}_2$ gets as input srs and forwards this reference string to run $(C(X), \mathsf{t}, [z_I]_2, \pi) \leftarrow \mathcal{A}(\mathsf{srs})$. When $\mathcal{A}$ outputs a proof $\pi = (W_1, W_2, W_3)$, then it also outputs the representation $w_3(X), w_2(X)$ of maximum degree d such that

$$z_I = [f(x)]_2 = [w_3(x)x^{-N+k-1} + x^k]_2, W_2 = [w_2(x)]_1$$

Then $\mathcal{B}_2$ computes the degree N polynomial $g(X) = Z_H(X) - z_I(X)w_2(X)$ and solves to find the N roots. It checks amongst these roots whether any solution x corresponds to the qdlog challenge and if yes it returns x. Else it aborts.

By the second verification equation we have that $Z_H(x) - g(x)w_2(x) = 0$ whenever $\mathcal{A}$ convinces the verifier. See that if $g(X)$ does not divide $Z_H(X)$, then $Z_H(X) - g(X)w_2(X) \neq 0$. But then x must lie in the roots and $\mathcal{B}_2$ succeeds.

$\underline{\mathsf{Game}^2 \mapsto 0}$ We finally show that for any adversary $\mathcal{A}$

$$\mathcal{A}^2_{\mathcal{A},\mathcal{E}}(\lambda) = 0$$

Indeed, when $\mathcal{A}$ also outputs the representation $w_3(X)$, we have that either $[z_I]_2 = [z_I(x)]_2$ for $z_I(X) = w_3(X)X^{-(N-k+1)} + X^k$ a degree N polynomial dividing $z_H(X)$, or Game^2 aborts. The adversary $\mathcal{A}$ also outputs a representation $w_1(X)$ of maximum degree N such that $[W]_1 = [w_1(x)]_1$. By the first verification equation we have that $t(X) = C(X) - w_1(X)z_I(X)$ is such that $\mathsf{t} = [t(X)]_1$. Further $t(\xi_i) = C(\xi_i) + 0$ for all $i \in [k]$, making $z_I(X)$ and $t(X)$ a correct witness. □