REFERENCES

[1] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. 2016. Structure-Preserving Signatures and Commitments to Group Elements. *J. Cryotology* 2 (2016), 363–421. Issue 29.

[2] S. Angel, E. Ioannidis, E. Margolin, S. Setty, and J. Woods. 2023. Reef: Fast Succinct Non-Interactive Zero-Knowledge Regex Proofs. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2023/1886.

[3] A. Arun, S. Setty, and J. Thaler. 2023. Jolt: Snarks for virtual machines via lookups. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2023/1217.

[4] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. 2019. Aurora: Transparent Succinct Arguments for R1CS. In *EUROCRYPT*. 103–128.

[5] S. Bowe, J. Grigg, and D. Hopwood. 2019. Recursive Proof Composition Without a Trusted Setup. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2019/1021.

[6] B.ünz and J. Chen. 2024. Proofs for Deep Thought: Accumulation for Large Memories and Deterministic Computations. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2024/325.

[7] B. Bünz and B. Chen. 2023. Protostar: Generic Efficient Accumulation/Folding for Special-sound Protocols. IACR Cryptol. ePrint Arch..

[8] B. Bünz, A. Chiesa, P. Mishra, and N. Spooner. 2020. Proof-Carrying Data from Accumulation Schemes. IACR. ePrint Arch 2020/499.

[9] B. Bünz, M. Maller, P. Mishara, N. Tyagi, and P. Vesely. 2021. Proofs for Inner Pairing Products and Applications. In *ASIACRYT*. 65–97.

[10] J. Camenisch and M. Stadler. 1997. Efficient Group Signature Schemes for Large Groups. In *CRYPTO (LNCS 1296)*. 410–424.

[11] M. Campanelli, A. Faonio, D. Fiore, T. Li, and H. Lipmaa. 2023. Lookup Arguments: Improvements, Extensions and Applications to Zero-Knowledge Decision Trees. https://hal.science/hal-04234948/document.

[12] M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodriguez. 2021. Lunar: A Toolbox for More Efficient Universal and Updatable zkSNARKs and Commit-and-Prove Extensions. In *ASIACRYPT*. 3–33.

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

[14] B. Chen, B. Bünz, D. Boneh, and Z. Zhang. 2023. HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates. In *EUROCRYPT*. 499–530.

[15] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. 2020. Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS.. In *EUROCRYPT*. 738–768.

[16] A. Choudhuri, S. Garg, A. Goel, S. Sekar, and R. Sinha. 2023. SublonK: Sublinear Prover Plonk. https://eprint.iacr.org/2023/902.

[17] T. Conley, N. Diaz, D. Espada, A. Kuruvilla, and X. Fu. 2023. IZPR: Instant Zero Knowledge Proof of Reserve. https://eprint.iacr.org/2023/1156.

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

[19] L. Eagen and A. Gabizon. 2023. ProtoGalaxy: Efficient ProtoStar-Style Folding of Multiple Instances. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2023/1106.

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

[21] A. Gabizon, Z. Williamson, and O. Ciobotaru. 2019. PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2019/953.

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

[23] N. Gailly, M. Maller, and A. Nitulescu. 2022. SnarkPack: Practical SNARK Aggregation. In *FC*. 203–229.

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

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

[26] A. Kate, G. M. Zaverucha, and I. Goldberg. 2010. Constant-Size Commitments to Polynomials and Their Applications. In *ASIACRYPT*. 177–194.

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

[28] A. Kothapalli and S. Setty. 2022. SuperNova: Proving universal machine executions without universal circuits. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2022/1758.

[29] A. Kothapalli and S. Setty. 2023. CycleFold: Folding-scheme-based Recursive Arguments Over a Cycle of Elliptic Curves. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2023/1192.

[30] A. Kothapalli and S. Setty. 2023. HyperNova: Recursive arguments for customizable constraint systems. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2023/573.pdf.

[31] A. Kothapalli, S. Setty, and I. Tzialla. 2022. Nova: Recursive Zero-Knowledge Arguments from Folding Schemes.. In *CRYPTO*. 359–388.

[32] N. Luo, C. Weng, J. Singh, G. Tan, R. Piskac, and M. Raykova. 2023. Privacy-Preserving Regular Expression Matching using Nondeterministic Finite Automata. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2023/643.pdf.

[33] W. Nguyen, D. Boneh, and S. Setty. 2023. Revisiting the Nova Proof System on a Cycle of Curves. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2023/969.

[34] W. Nguyen, T. Datta, B. Chen, N. Tyagi, and D. Boneh. 2024. Mangrove: A Scalable Framework for Folding-based SNARKs. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2024/416.

[35] L Pearson, J. Fitzgerald, and H. Masip. 2022. PlonKup: Reconciling Plonk with plookup. IACR Cryptol. ePrint Arch. 2022/086.

[36] M. Raymond, G. Evers, J. Ponti, D. Krishnan, and X. Fu. 2023. Efficient Zero Knowledge for Regular Language. In *SecureCOM*. (to appear).

[37] R. Sedgewick and K. Wayne. 2011. *Algorithms 4e.* Addison-Wesley Professional.

[38] S. Setty, J. Thaler, and R. Wahby. 2023. Unlocking the lookup singularity with Lasso. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2023/1216.

[39] L. Soukhanov. 2023. Reverie: an end-to-end accumulation scheme from Cyclefold. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2023/1888.

[40] S. Srinivasan, I. Karantaidou, F. Baldimtsi, and C. Papamanthou. 2022. Batching, Aggregation, and Zero-Knowledge Proofs in Bilinear Accumulators. In *CCS*. 2719–2733.

[41] A. Vark and Y. X. Zhang. 2024. Folding Custom Gates with Verifier Input. https://arxiv.org/abs/2401.11364.

[42] A. Zapico, A. Gabizon, D. Khovratovich, M. Maller, and C. Råfols. 2022. Baloo: Nearly Optimal Lookup Arguments. IACR Cryptol. ePrint Arch..

[43] zcash. 2022. The Halo2 Book. https://zcash.github.io/halo2/concepts.html.

[44] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou. 2017. A Zero-Knowledge Version of vSQL. IACR Cryptol. ePrint Arch. 2017:1146.

A ZERO KNOWLEDGE cq PROTOCOL

We provide a zero knowledge version of the cq [18] protocol. In [11] two highly optimized zero knowledge protocols (cq+, cq++) are presented (with proof size $7\mathbb{G}_1 + 1\mathbb{F}$). Our protocol has the same asymptotic (faster concrete in certain situations) prover cost, a slightly higher proof size ($7\mathbb{G}_1 + 2\mathbb{F}$) and verifier cost (one more pairing than cq++). Compared with [11] and the original cq protocol, the advantage of our protocol is that the size of the query table (i.e., number of lookups) does not have to be a factor of the scalar field size or the lookup table size (e.g., de-factor power of 2 as in [11]). In some application scenarios, this is desirable as one does not have to fill the query tables with "dummy" entries to round up its size. Then the concrete performance of the prover in our protocol can be potentially 0.6× faster, e.g., if the query table size is 2^k+1. The main idea of our protocol is to use QA-NIZK for linear subspace [13, 27] to reason about sum of arrays, instead of relying on the Aurora Lemma [4], which needs to work with a multiplicative subgroup of the scalar field.

A.1 Problem Statement

We use $\lceil n \rceil$ to denote n rounded up to the next power of 2 (e.g., 2048 for 1025). Let $\mathbf{T} \in \mathbb{F}^{\mathbf{N}}$ be a secret lookup table of size N. We define $T(X) = \sum_{i=1}^{N} \mathbf{T}_i L_{N,i}(X)$ as its *encoding polynomial*. Let r_T be a random nonce and define $\hat{T}(X) = T(X) + z_{\mathbb{H}_N}(X)$, where $z_{\mathbb{H}_N}(X) = X^N - 1$ is the vanishing polynomial for all roots of unity of $\mathbb{H}_N$. We say $[\hat{T}(s)]_1$ is the Pedersen (vector) commitment to $\mathbf{T}$. Similarly, given a query table $\mathbf{t} \in \mathbb{F}^n$ (where n is not necessarily a power of 2 that supports FFT), define $t(X) = \sum_{i=1}^{n} \mathbf{t}_i L_{\lceil n \rceil,i}(X)$ and $\hat{t}(X) = t(X) + r_t z_{\mathbb{H}_{\lceil n \rceil}}(X)$ for randomness r_t. The vector commitment to $\mathbf{t}$ is $[\hat{t}(s)]_1$. Note that the computing cost of $[\hat{t}(s)]_1$, in terms of group operations, is n instead of $\lceil n \rceil$ (even though the Lagrange base polynomials are defined over domain of $\lceil n \rceil$). We often use $\mathsf{C}_{\mathbf{T}}$ to denote $[\hat{T}(s)]_1$. Similar notations apply to all other vectors and their encoding polynomials in our protocol.

We denote the protocol as $\pi_{\mathsf{ZK\_CQ}}$, which provides a collection of algorithms as shown below.

(1) $(\mathsf{pk}_{\mathsf{kzg}}, \mathsf{vk}_{\mathsf{kzg}}) \leftarrow \mathtt{SetupCQ}(s, N, n)$: a trusted set-up given security parameter λ and lookup (query) table size $N(n)$, samples bilinear groups and generates the prover and verifier keys $(\mathsf{pk}_{\mathsf{kzg}}, \mathsf{vk}_{\mathsf{kzg}})$ for the KZG polynomial commitment [26]. We require N and $\lceil n \rceil$ (note: not n) to be a factor of $|\mathbb{F}|$ which allows FFT. [3]

(2) $(\mathsf{pk}, \mathsf{vk}, \mathsf{C}_{\mathbf{T}}) \leftarrow \mathtt{PreprocessCQ}(\mathbf{T}, r_T, \mathsf{pk}_{\mathsf{kzg}}, \mathsf{vk}_{\mathsf{kzg}})$: The prover generates the auxiliary information (e.g., the cached polynomials for cq) based on $\mathsf{pk}_{\mathsf{kzg}}$. It also returns $\mathsf{C}_{\mathbf{T}} = [\hat{T}(s)]_1$, and also expands $\mathsf{vk}_{\mathsf{kzg}}$ with $[\hat{T}(s)]_2$. Note it does not have access of trapdoor.

(3) $(\mathsf{C}_{\mathbf{t}}, \pi) \leftarrow \mathtt{ProveCQ}(\mathsf{pk}, \mathbf{t}, r_t)$: It produces a zero knowledge proof π for $\mathbf{t} \hat{\subset} \mathbf{T}$, and the vector commitment of $\mathbf{t}$.

(4) $1/0 \leftarrow \mathtt{VerCQ}(\mathsf{vk}, \mathsf{C}_{\mathbf{t}}, \pi)$: It checks that π is valid. Note that the proof implies $\mathsf{C}_{\mathbf{t}}$ hides a vector of size n.

(5) $\pi \leftarrow \mathtt{FoldProveCQ}(\mathsf{pk}_{\mathbf{U}}, \mathsf{pk}_{\mathbf{V}}, \mathbf{u}, \mathbf{v}, \alpha)$: Given the prover keys for two lookup tables $\mathbf{U}$ and $\mathbf{V}$, it produces π for asserting that $\mathbf{u} + \alpha\mathbf{v} \hat{\subset} \mathbf{U} + \alpha\mathbf{V}$. It is required that both prover keys agree on the same (N, n).

A.2 Adding Zero Knowledge

We use two different techniques for lifting cq to zk, one bounded-leaky and the other is complete zk.

A.2.1 Bounded Leaky-Zk. The idea is to mask a polynomial with a random polynomial. It is formally defined as bounded-zk in Lunar [12], but was unofficially used in PlonK [21], [4] and then summarized in [16, Section 3.6]. This technique is used in three concurrent proposals in making cq zero-knowledge [11, 16, 17]. We formally define it below.

Lemma A.1. *Let $\mathbf{f} \in \mathbb{F}^n$, and let $f(X) = \sum_{i=1}^{n} \mathbf{f}_i \cdot L_{\lceil n \rceil,i}(X)$. Given a fixed $b > 0$, we say that $\hat{f}$ is a b-*bounded-zk extension *of $f(X)$ in the sense that there exists a b-1 degree random polynomial $R(X)$ s.t.*

$$\hat{f}(X) = f(X) + R(X) z_{\mathbb{H}_{\lceil n \rceil}}(X)$$

$\hat{f}$ satisfies the following properties:

(1) *For $i \in [n]$: $\hat{f}(\omega^i_{\lceil n \rceil}) = \mathbf{f}_i$, where $\omega_{\lceil n \rceil}$ is the $\lceil n \rceil$'th root of unity of $\mathbb{F}$.*

(2) *Fix b* random *points. Given the evaluation of $\hat{f}$ over these points, for any $\mathbf{u} \in \mathbb{F}^{\lceil n \rceil}$, there exists one and only one b-1 degree polynomial $U(X)$ s.t. $\hat{f}(X) = u(X) + U(X) z_{\mathbb{H}_{\lceil n \rceil}}(X)$, where $u(X) = \sum_{i=1}^{\lceil n \rceil} \mathbf{u}_i \cdot L_{\lceil n \rceil,i}(X)$.*

Property (2) essentially states that as long as there are no more than b evaluations, $\hat{f}$ can be generated from any arbitrary vector (and since there is a one-to-one correspondence of the mask polynomial, the distribution of $\mathbf{u}$ is uniform), thus leaking no information about $\mathbf{f}$. Note that for the 1-leaky case, $R(X)$ is degree 0.

(Proof Sketch:) Property (1) is apparent in that for any $i \in [n]$: $z_{\mathbb{H}_{\lceil n \rceil}}\left(\omega^i_{\lceil n \rceil}\right) = 0$. Property (2) is true because given b random points $\{r_i\}_{i=1}^{b}$: one has for each $i \in [b]$: $U(r_i) = (\hat{f}(r_i) - u(r_i))/z_{\mathbb{H}_{\lceil n \rceil}}(r_i)$. Given that r_i is randomly sampled, the probability that $z_{\mathbb{H}_{\lceil n \rceil}}(r_i) = 0$ is negligible and it is also unlikely there are $i \neq j$ s.t. $r_i = r_j$. In this case, given b points, $U(X)$ can always be computed using Lagrange interpolation and its degree is bounded by $b-1$, and it is the unique $b-1$ degree polynomial that passes through these points.

Remark: for bounded-leaky-zk to apply, $\hat{f}(X)$ has to be evaluated at *random* points. If one evaluates $\hat{f}$ at known points, e.g., $\omega^i_{\lceil n \rceil}$, it will still leak information about $\mathbf{f}$. In [17], we rely on a univariate zk-VPD scheme which is introduced in [44]. In this paper, we introduce the use of QA-NIZK for a more efficient construction.

A.2.2 QA-NIZK [13, 27]. In the cq protocol, given a vector $\mathbf{A} \in \mathbb{F}^N$ and $\mathbf{B} \in \mathbb{F}^n$, there is a step which needs to prove that $\sum_{i=1}^{N} \mathbf{A}_i = \sum_{i=1}^{n} \mathbf{B}_i$. The Aurora lemma [4] is applied to reason about this: let

[3] To us it is still an unknown problem if the restriction on N being a power of 2 that allows FFT can be relaxed. It involves modifying the algorithm for batch computing $[Q_i(s)]_1$ using the amortized KZG proof algorithm (FK algorithm) in [20]. A factor polynomial of $z_{\mathbb{H}_{\lceil N \rceil}}(X)/z_{\mathbb{H}_N}(X)$ needs to be applied to each quotient polynomial. An extension of the FK algorithm for it seems not straightforward, and the difficulty arises in applying the Toeplitz matrix.

[4] NOTE: It was mentioned in Lunar's introduction in bounded-leaky zk, however, didn't find where it is in [21], **CHECK LATER.**

$A(X)$ and $B(X)$ be the encoding polynomial for $\mathbf{A}$ and $\mathbf{B}$ respectively, it is known that $A(0) = \frac{\sum_{i=1}^{N} \mathbf{A}_i}{N}$, and similar applies to $B(X)$. In [17], zk is achieved using zk-VPD scheme [44] where the value of $A(0)$ hides in a Pedersen commitment. In [11], an additional $C(X)$ is defined "roughly" as $A(X) - \frac{n \cdot z_{\mathbb{H}_N}(X)}{N \cdot z_{\mathbb{H}_n}(X)} B(X)$, and Aurora lemma is applied again (but in zero knowledge) to reason about $C(X)$. This approach results in the most efficient zero knowledge cq protocol so far ($7\mathbb{G}_1 + 1\mathbb{F}$ proof, $8n$ group operations for prover cost and 6 pairings).

In this paper, we leverage the QA-NIZK technique, which provides a conceptually easier and even more efficient prover (in situations where $n \neq \lceil n \rceil$). In particular, our technique allows n (query table size) not a power of two, and n does not have to be a factor of N (which is required by the techniques of [11]).

Our starting point is the quasi-adaptive NIZK (QA-NIZK) for linear subspace of vector of group elements [27]. Given $\mathbf{M} \in \mathbb{G}_1^{n \times t}$, and $\mathbf{y} \in \mathbb{G}_1^n$, the technique proves the knowledge of a secret vector $\mathbf{x} \in \mathbb{F}^t$ s.t. $\mathbf{y} = \mathbf{M}\mathbf{x}$. In [27], there is a restriction that $n > t$ (the number of equations needs to be greater than the number of variables). This restriction is removed in LegoSnark [13, Appendix D], which also ports the proof to the algebraic group model (AGM). We now apply the LegoSnark variation, and provides the zk-SNARK for the following relation:

$$\pi_{\mathsf{qa}}(N, n, \mathrm{C_A}, \mathrm{C_B})\{(\mathbf{A}, \mathbf{B}, r_a, r_b) :$$
$$\mathrm{C_A} = \sum_{i=1}^{N} \mathbf{A}_i [L_{N,i}(s)]_1 + r_a [z_{\mathbb{H}_N}(s)]_1 \wedge$$
$$\mathrm{C_B} = \sum_{i=1}^{n} \mathbf{B}_i [L_{\lceil n \rceil, i}(s)]_1 + r_b [z_{\mathbb{H}_{\lceil n \rceil}}(s)]_1 \wedge$$
$$\sum_{i=1}^{N} \mathbf{A}_i = \sum_{i=1}^{n} \mathbf{B}_i$$
$$\}$$

Intuitively, π_{qa} states that the sum of two committed vectors are the same, and note that it explicitly asserts the vector size (degree bound) of the two vectors.

We present the construction for π_{qa} in Figure 1. Matrix $\mathbf{M}$ has three rows. The first row encodes the relation for generating $[\hat{A}(s)]_1$, and the second for $[\hat{B}(s)]_1$, and the third produces $[0]_1$ if the sum of $\mathbf{A}$ is equal to that of $\mathbf{B}$. Notice that in ProveQANIZK, the witness $\mathbf{w}$ is the concatenation of $\mathbf{A}$, its random nonce r_a, $\mathbf{B}$, and its random nonce r_b.

Lemma A.2. *[13, Appendix D] Under the discrete logarithm assumption in AGM, the construction in Figure 1 is perfectly complete, computationally knowledge sound, and perfectly zero knowledge.*

Lemma A.2 directly follows Theorem D.1 in [13, Appendix D], because the distribution of $\mathbf{M}$ is witness sampleable.

Efficiency: When the protocol is used in the zk-cq protocol, the prover cost is $2n$ because $\mathbf{A}$ is sparse (up to n non-zero entries). The verifier cost is actually 3 pairings, because the last entry in $\mathbf{x}$ in VerifyQANIZK is $[0]_1$, and its pairing can be skipped.

Remark: When the QA-NIZK protocol is used in zk-cq, the adversary that attacks knowledge soundness will have access of extra $\mathbb{G}_1$ elements: $([s^i])_{i=0}^{N}$ (in addition to the group elements in $\mathbf{M}$ and $\mathbf{P}$). This will not increase the power of adversary in AGM. We provide a brief proof below.

1 $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathtt{SetupQANIZK}(N, n, s)$
Define $M \in \mathbb{F}^{3 \times (N+n+2)}$ as following:
$$\begin{pmatrix} ([L_{N,i}(s)]_1)_{i=1}^{N}, [z_{\mathbb{H}_N}(s)]_1, ([0])_{i=1}^{n+1} \\ ([0]_1)_{i=1}^{N+1}, ([L_{\lceil n \rceil, i}]_1)_{i=1}^{n}, [z_{\mathbb{H}_{\lceil n \rceil}}(s)]_1 \\ ([1]_1)_{i=1}^{N}, [0]_1, ([|\mathbb{F}| - 1]_1)_{i=1}^{n}, [0]_1 \end{pmatrix}$$
Sample $\mathbf{k} \in \mathbb{F}^3$, $a \in \mathbb{F}$.
Compute $\mathbf{P} \leftarrow \mathbf{M}^\intercal \mathbf{k} \in \mathbb{G}_1^{N+n+2}$, $\mathbf{C} = a \cdot [\mathbf{k}]_2 \in \mathbb{G}_2^3$.
Let $\mathsf{pk} = \mathbf{P}$ and $\mathsf{vk} = (\mathbf{C}, [a]_2)$. Return $(\mathsf{pk}, \mathsf{vk})$.
2 $(\mathbf{x}, \pi) \leftarrow \mathtt{ProveQANIZK}(\mathbf{A}, \mathbf{B}, \mathsf{pk})$:
Sample r_a, r_b from $\mathbb{F}$ and define $\mathbf{w} = \mathbf{A} || r_a || \mathbf{B} || r_b \in \mathbb{F}^{N+n+2}$.
Compute $[\hat{A}(s)] \leftarrow \sum_{i=1}^{N} \mathbf{M}_{1,i} \mathbf{A}_i + \mathbf{M}_{1,N+1} r_a \in \mathbb{G}_1$.
Compute $[\hat{B}(s)] \leftarrow \sum_{i=1}^{n} \mathbf{M}_{2,i+N+1} \mathbf{B}_i + \mathbf{M}_{2,N+n+2} r_b \in \mathbb{G}_1$.
Compute $\pi \leftarrow \mathbf{w}^\intercal \mathbf{P} \in \mathbb{G}_1$.
Let $\mathbf{x} = ([\hat{A}(s)]_1, [\hat{B}(s)]_1, [0]_1) \in \mathbb{G}_1^3$.
Return $(\mathbf{x}, \pi)$.
3 $1/0 \leftarrow \mathtt{VerifyQANIZK}(([\hat{A}(s)]_1, [\hat{B}(s)]_1), \pi, \mathsf{vk})$:
Let $\mathbf{x} = ([\hat{A}(s)]_1, [\hat{B}(s)]_1, [0]_1) \in \mathbb{G}_1^3$.
Return 1 if and only if $\prod_{i=1}^{3} e(\mathbf{x}_i, \mathbf{C}_i) = e(\pi, [a]_2)$

Figure 1: QA-NIZK for $\sum_{i=1}^{N} \mathbf{A}_i = \sum_{i=1}^{n} \mathbf{B}_i$.

Let $\mathbf{z} \in \mathbb{G}_1^{2N+2n+7}$ be the collection of group elements collected from $\mathbf{M}$ and $\mathbf{P}$. Let $\mathbf{u} = ([s^i]_1)_{i=1}^{N}$. Assume that the adversary can construct a false proof π given $\mathbf{z}$ and $\mathbf{u}$ s.t. $\mathtt{VerifyQANIZK}(\mathrm{C_A}, \mathrm{C_B}, \pi)$ returns 1 but $\sum_{i=1}^{N} \mathbf{A}_i \neq \sum_{i=1}^{n} \mathbf{B}_i$. We argue that the same π can be produced using $\mathbf{z}$ only.

By the AGM model, there exist $\mathbf{a} \in \mathbb{F}^{2N+2n+7}$ and $\mathbf{b} \in \mathbb{F}^N$ s.t. $\pi = \sum_{i=1}^{2N+2n+7} \mathbf{a}_i \mathbf{z}_i + \sum_{i=1}^{N} \mathbf{b}_i \mathbf{u}_i$. Note that $\sum_{i=1}^{N} \mathbf{b}_i \mathbf{u}_i$ is a KZG commitment to an N-degree polynomial $b(X)$. Then $b(X)$ can always be re-written as $\sum_{i=1}^{N} \mathbf{c}_i L_{N,i}(X) + \mathbf{c}_{N+1} z_{\mathbb{H}_N}(X)$. Then merge array $\mathbf{c}$ into $\mathbf{a}$ for the corresponding keys in $\mathbf{z}$, one obtains π using $\mathbf{z}$ only.

A.3 Zero Knowledge cq Protocol

In Figure 2 we present the details of the zero knowledge cq protocol. As the look-up table $\mathbf{T}$ is secret, we split the setup process into two parts: a trusted set-up which generates KZG polynomial commitment keys, and a pre-process procedure which generates the augment information (e.g., cached quotients) for the cq protocol. Given $\mathrm{C_T}$, which is to be re-used multiple times in multiple lookup proofs, the prover and verifier needs to run a *one-time* proof for (1) the prover's knowledge of $\mathbf{T}$, and (2) $\mathrm{C_T}$ is a well formed commitment of $\mathbf{T}$ (i.e., the size of $\mathbf{T}$ is bounded by N). This can be trivially accomplished re-using the QA-NIZK proof, and we skip the details here.

All other group elements in $\mathsf{pk}_{\mathsf{kzg}}$ other than the $([s^i]_1)_{i=1}^{N}$ can be computed from it, thus does not increase power of the soundness adversary. Note that in the pk, the $z_{\mathbb{H}_n}(X)$ represents the vector of co-efficients of $z_{\mathbb{H}_n}(X)$.[5] The PreprocessCQ mainly computes the information (cached quotients) dependent on $\mathbf{T}$. This step can be merged with SetupCQ if $\mathbf{T}$ is public, to avoid costly group FFT. However, notice that the asymptotic complexity (in terms of group

[5] As n is not guaranteed to be a power of 2, we do not have the convenience as $z_{\mathbb{H}_{\lceil n \rceil}}(X) = X^{\lceil n \rceil} - 1$.

1 $(\mathsf{pk}_{\mathsf{kzg}}, \mathsf{vk}_{\mathsf{kzg}}) \leftarrow \mathtt{SetupCQ}(s, N, n)$:
Given trapdoor s, compute $((\mathbf{M}, \mathbf{P}), (\mathbf{D}, [a]_2)) \leftarrow \mathtt{SetupQANIZK}(N, n, s)$.
Let $\mathsf{pk}_{\mathsf{kzg}} \leftarrow \left(([s^i]_1, [s^i]_2)_{i=1}^N, \mathbf{M}, \mathbf{P}\right)$.
Let $\mathsf{vk}_{\mathsf{kzg}} \leftarrow ([1]_2, [s]_2, \mathbf{D}, [a]_2)$.
Return $(\mathsf{pk}_{\mathsf{kzg}}, \mathsf{vk}_{\mathsf{kzg}})$.

2 $(\mathsf{pk}, \mathsf{vk}, \mathsf{C}_{\mathbf{T}}) \leftarrow \mathtt{PreprocessCQ}(\mathbf{T}, r_T, \mathsf{pk}_{\mathsf{kzg}}, \mathsf{vk}_{\mathsf{kzg}})$:
Parse $\mathsf{pk}_{\mathsf{kzg}}$ and $\mathsf{vk}_{\mathsf{kzg}}$ as shown in SetupCQ.
Define $T(X) = \sum_{i=1}^N \mathbf{T}_i L_{N,i}(X)$, and $\hat{T}(X) = T(X) + r_T z_{\mathbb{H}_N}(X)$.
For $i \in [N]$ define $Q_i(X) = \frac{L_{N,i}(X) \cdot T(X) - \mathbf{T}_i \cdot L_{N,i}(X)}{Z_{\mathbb{H}_N}(X)}$.
Compute $[\hat{T}(s)]_1, [\hat{T}(s)]_2$ using $\mathsf{pk}_{\mathsf{kzg}}$.
Compute $([Q_i(s)]_1)_{i=1}^N$ using $\mathsf{pk}_{\mathsf{kzg}}$.
Let $\mathsf{pk} = \mathsf{pk}_{\mathsf{kzg}} || [\hat{T}(s)]_1 || ([Q_i(s)]_1, L_{N,i}(s))_{i=1}^N || (L_{\lceil n \rceil, i}(s))_{i=1}^n || [z_{\mathbb{H}_{\lceil n \rceil}}(s)]_1 || [z_{\mathbb{H}_N}(s)]_1 || z_{\mathbb{H}_n}(X)$.
let $\mathsf{vk} = \mathsf{vk}_{\mathsf{kzg}} || [\hat{T}(s)]_2 || [z_{\mathbb{H}_N}(s)]_2 || [z_{\mathbb{H}_n}(s)]_1$.
Return $(\mathsf{pk}, \mathsf{vk}, \mathsf{C}_{\mathbf{T}})$.

3 $(\mathsf{C}_{\mathbf{t}}, \pi) \leftarrow \mathtt{ProveCQ}(\mathsf{pk}, \mathbf{t}, r_t)$:
(1) Parse pk and vk as shown in PreprocessCQ. Sample r_A, r_B, r_m from $\mathbb{F}$.
(2) Let $\hat{t}(X) = \sum_{i=1}^n \mathbf{t}_i L_{\lceil n \rceil, i}(X) + r_t z_{\mathbb{H}_{\lceil n \rceil}}(X)$. Compute $[\hat{t}(s)]_1$.
(3) Define $\mathbf{m} \in \mathbb{F}^N$ s.t. each $\mathbf{m}_i$ is the multiplicity of $\mathbf{T}_i$ in $\mathbf{t}$. Define $\hat{m}(X) = \sum_{i=1}^N \mathbf{m}_i L_{N,i}(X) + r_m z_{\mathbb{H}_N}(X)$. Compute $[\hat{m}(s)]_1$.
(4) Fiat-Shamir: $\beta = \mathsf{hash}(\mathsf{vk} || [\hat{t}(s)]_1 || [\hat{m}(s)]_1)$.
(5) Define $\mathbf{A} \in \mathbb{F}^N$ s.t. for each $i \in [N]$: $\mathbf{A}_i = \mathbf{m}_i / (\mathbf{T}_i + \beta)$. Define $A(X) = \sum_{i=1}^N \mathbf{A}_i L_{N,i}(X)$, and $\hat{A}(X) = A(X) + r_A z_{\mathbb{H}_N}(X)$. Compute $[\hat{A}(s)]_1$.
(6) Define $Q_A(X) = (A(X)(T(X) + \beta) - m(X))/z_{\mathbb{H}_N}(X)$. Compute $[Q_A(s)]_1$ using the $[Q_i(s)]_1$ in pk. Then compute $[\hat{Q}_A(s)]_1 \leftarrow [Q_A(s)]_1 + r_A[T(s)]_1 + r_T[A(s)]_1 + r_A r_T [z_{\mathbb{H}_N}(s)]_1 + [r_A\beta - r_m]_1$.
(7) Define $\mathbf{B} \in \mathbb{F}^n$ s.t. $\mathbf{B}_i = 1/(\mathbf{t}_i + \beta)$ for each $i \in [n]$. Define $B(X) = \sum_{i=1}^n B_i \cdot L_{\lceil n \rceil, i}(X)$, and $\hat{B}(X) = B(X) + r_B z_{\mathbb{H}_{\lceil n \rceil}}(X)$. Compute $[\hat{B}(s)]_1$.
(8) computes $\hat{Q}_B(X) \leftarrow \frac{\hat{B}(X)(\hat{t}(X)+\beta)-1}{z_{\mathbb{H}_n}(X)}$. Compute $[\hat{Q}_B(s)]_1$.
(9) Fiat-Shamir
$(\gamma, \eta) \leftarrow \mathsf{hash}(\beta || [\hat{A}(s)]_1 || [\hat{Q}_A(s)]_1 || [\hat{B}(s)]_1 || [\hat{Q}_B(s)]_1)$.
(10) Compute $B_\gamma \leftarrow \hat{B}(\gamma)$. $Z_\gamma \leftarrow z_{\mathbb{H}_n}(\gamma)$. Compute $D(X) = B_\gamma(\hat{t}(X) + \beta) - 1 - \hat{Q}_B(X) Z_\gamma$. Compute $P(X) = \frac{(B(X) - B_\gamma) + \eta D(X) + \eta^2 (z_{\mathbb{H}_n}(X) - Z_\gamma)}{X - \gamma}$. Compute $[P(s)]_1$.
(11) Let $\mathbf{w} = \mathbf{A} || r_A || \mathbf{B} || r_B$, and compute $\pi_{\mathsf{qa}} \leftarrow \mathbf{w}^\intercal \mathbf{P}$.
(12) Let $\pi = ([\hat{m}(s)]_1, [\hat{A}(s)]_1, [\hat{Q}_A(s)]_1, [\hat{B}(s)]_1, [\hat{Q}_B(s)]_1, [P(s)]_1, \pi_{\mathsf{qa}}, B_\beta, Z_\gamma)$.
Return $([\hat{t}(s)]_1, \pi)$.

4 $1/0 \leftarrow \mathtt{VerCQ}(\mathsf{vk}, \mathsf{C}_{\mathbf{t}}, \pi)$:
Parse π as $(\mathsf{C}_{\hat{m}}, \mathsf{C}_{\hat{A}}, \mathsf{C}_{Q_{\hat{A}}}, \mathsf{C}_{\hat{B}}, \mathsf{C}_{Q_{\hat{B}}}, \mathsf{C}_P, \pi_{\mathsf{qa}}, B_\beta, Z_\gamma)$. Parse vk as shown in PreprocessCQ. Compute $\beta \leftarrow \mathsf{hash}(\mathsf{vk} || \mathsf{C}_{\mathbf{t}} || \mathsf{C}_{\hat{m}})$, $(\gamma, \eta) \leftarrow \mathsf{hash}(\beta || \mathsf{C}_{\hat{A}} || \mathsf{C}_{Q_{\hat{A}}} || \mathsf{C}_{\hat{B}} || \mathsf{C}_{Q_{\hat{B}}})$. Return true if and only if all of the following checks pass.
(1) $e\left(\mathsf{C}_{\hat{A}}, [\hat{T}(s)]_2\right) = e\left(\mathsf{C}_{Q_{\hat{A}}}, [z_{\mathbb{H}_N}(s)]_2\right)\ e\left(\mathsf{C}_{\hat{m}} - \beta \mathsf{C}_{\hat{A}}, [1]_2\right)$
(2) $e\left(\pi_{\mathsf{qa}}, [a]_2\right) = e\left(\mathsf{C}_{\hat{A}}, \mathbf{D}_1\right) e\left(\mathsf{C}_{\hat{B}}, \mathbf{D}_2\right)$.
(3) $e\Big(\mathsf{C}_{\hat{B}} - [B_\gamma]_1 + \eta\left(B_\gamma(\mathsf{C}_{\mathbf{t}} + [\beta]_1) - [1]_1 - Z_\gamma \mathsf{C}_{Q_{\hat{B}}}\right) + \eta^2\left([z_{\mathbb{H}_n}(s)]_1 - [Z_\gamma]\right), [1]_2\Big) = e\left(\mathsf{C}_P, [s - \gamma]_2\right)$

Figure 2: Zero Knowledge cq Protocol

operations) of PreprocessCQ is still $O(N \log(N))$, the same as the original cq protocol. However it incurs a higher field operation cost of $O(n \log^2(n))$ for pre-computing $z_{\mathbb{H}_n}(X)$, which is negligible compared with group operation cost when n is small.

We now elaborate the design idea of the ProveCQ(). It follows the basic idea of [25], that $\mathbf{t} \hat{\subset} \mathbf{T}$ if and only if there exists a vector $\mathbf{m}$ s.t. $\mathbf{m}_i$ represents the multiplicity $\mathbf{T}_i$ in $\mathbf{t}$. For a uniformly sampled β:

$$\sum_{i=1}^N \frac{\mathbf{m}_i}{\mathbf{T}_i + \beta} = \sum_{i=1}^n \frac{1}{\mathbf{t}_i + \beta} \tag{1}$$

Then the protocol (steps 1-7) follows cq. It declares polynomials $A(X)$ and $B(X)$ to model the LHS and RHS of Equation 1. For example, it computes $Q_A(X)$ s.t. $A(X)(T(X) + \beta) - m(X) = Q_A(X) z_{\mathbb{H}_N}(X)$. The difference is that all polynomials are masked by a blind factor, e.g., $\hat{A}(X) = A(X) + r_A z_{\mathbb{H}_N}(X)$. This is also the same as [11, 16]. Note that $[\hat{Q}_A(s)]_1$ can be computed using the cached $([Q_i(s)]_1)_{i=1}^N$ and thus only costing $O(n)$ given $\mathbf{A}$ is sparse.

At Steps 8-10, we adopt the technique introduced in [11] so that we can save two field elements for proving $\hat{B}(X)(\hat{t}(X) + \beta) - 1 = \hat{Q}_B(X) z_{\mathbb{H}_n}(X)$. The basic idea is to let the verifier sample γ and then provide $B_\gamma = \hat{B}(\gamma)$. Define $D(X) = B_\gamma(\hat{t}(X) + \beta) - 1 - \hat{Q}_B(X) z_{\mathbb{H}_n}(\gamma)$. It is clear that $D(X)$ evaluates to 0 at γ. Note that $\hat{B}(X) - B_\gamma$ also evaluates to 0 at γ. Let verifier raise another random challenge η, the two polynomial checks can be batched as: there exists a $P(X)$ s.t. the following is true by applying the KZG polynomial commitment check:

$$(B(X) - B_\gamma) + \eta(D(X) - 0) = P(X)(X - \gamma)$$

Consider the cost of computing $z_{\mathbb{H}_n}(\gamma)$ at verifier. It incurs $O(n)$ field operations.[6] To cut the asymptotic verifier complexity to $O(1)$, instead, we let the prover compute $Z_\gamma = z_{\mathbb{H}_n}(\gamma)$ and pass the value in proof (which incurs one more field element than cq++). Then we need to provide a KZG proof for it, which again, we can fold into $P(X)$. In this case, we re-define $P(X)$ as the proof polynomial for the following folded KZG checks:

$$(B(X) - B_\gamma) + \eta(D(X) - 0) + \eta^2(z_{\mathbb{H}_n}(X) - Z_\gamma) = P(X)(X - \gamma)$$

Then it remains to prove that the sum of $\mathbf{A}$ is the same as $\mathbf{B}$. Our protocol then departs from cq and cq+ [11]. The earlier work including [11, 18] applies the Aurora Lemma [4, Remark 5.6], and it needs n to be the size of a *multiplicative subgroup* of $\mathbb{F}$. By Lagrange theorem, n should divide $|\mathbb{F}|$, and for example n is (de-factor) required to be a power of 2 in [11]. In this work, this restriction is removed because we use the QA-NIZK.

Steps 11-12 accomplish the QA-NIZK proof with $2n$ (instead of $2\lceil n \rceil$) group operations (because $\mathbf{A}$ is sparse). We end up with a proof of 7 $\mathbb{G}_1$ +2 $\mathbb{F}$ elements, which is the same concrete prover cost (and better when $n \neq \lceil n \rceil$) compared with cq++ in [11] and 7 pairings in verifier (1 more than cq++).

To save the verifier cost, Equations (1) and (3) of VerCQ can be batched using a random μ as following:

$$\begin{aligned} & e\Big(\mathsf{C}_{\hat{B}} - [B_\gamma]_1 + \eta(B_\gamma(\mathsf{C}_{\mathbf{t}} + [\beta]_1) - [1]_1 - Z_\gamma \mathsf{C}_{Q_{\hat{B}}}) + \\ & \eta^2([z_{\mathbb{H}_n}(s)]_1 - [Z_\gamma]_1) + \mu(\mathsf{C}_{\hat{m}} - \beta \mathsf{C}_{\hat{A}}), [1]_2\Big) \\ & e\left(\mu \mathsf{C}_{Q_{\hat{A}}}, [z_{\mathbb{H}_N}(s)]_2\right) \\ = \quad & e\left(\mathsf{C}_P, [s - \gamma]_2\right)\ e\left(\mu \mathsf{C}_{\hat{A}}, [\hat{T}(s)]_2\right) \end{aligned}$$

Theorem A.3. *Under the Q-DLOG assumption in the AGM model and assuming the Random Oracle Model, the construction presented*

[6] Note that n may not be a power of 2, so we do not have the convenience of $z_{\mathbb{H}_{\lceil n \rceil}}(X) = X^N - 1$.

in Figure 2 is complete, [7] *computational knowledge sound, and perfectly zero knowledge, assuming* $\mathsf{C_t}$ *has an extra knowledge proof which asserts it is well formed that there exists a* $\mathbf{t}$ *and* r_t *s.t.* $\mathsf{C_t} = [\sum_{i=1}^{n} \mathbf{t}_i L_{\lceil n \rceil,i}(s) + r_t z_{\mathbb{H}_n}(s)]_1$. [8].

Proof. **Zero Knowledge:** The proof for completeness is trivial. We first prove the perfectly zero knowledge property. The simulator samples trap-door $s, \mathbf{k}, a$ and constructs the $\mathsf{pk}_{\mathsf{kzg}}, \mathsf{vk}_{\mathsf{kzg}}$ using SetupCQ() (without using PreprocessCQ()). The simulator is then given $\mathsf{C_T}$ and $\mathsf{C_t}$, and tries to produce a transcript for proving $\mathbf{t} \hat{\subset} \mathbf{T}$ without knowing $\mathbf{t}$ and $\mathbf{T}$.

The simulator first samples random N-degree polynomials: $\hat{m}(X)$, $\hat{A}(X)$, and n-degree polynomial $\hat{B}(X)$, and verifier challenges β, γ, and η. Then the simulator computes the following using trapdoor s: $\mathsf{C}_{\hat{m}} \leftarrow [\hat{m}(s)]_1, \mathsf{C}_{\hat{A}} \leftarrow [\hat{A}(s)]_1, \mathsf{C}_{\hat{B}} \leftarrow [\hat{B}(s)]_1$. Note that the simulator knows for example $\hat{m}(s)$. Let $\mathbf{x} = (\mathsf{C}_{\hat{A}}, \mathsf{C}_{\hat{B}}, [0]_1) \in \mathbb{G}_1^3$, and $\mathbf{k}$ the trapdoor in QA-NIZK. Compute $\pi_{\mathsf{qa}} \leftarrow \mathbf{k}^\intercal \mathbf{x}$. The simulator also computes $Z_\gamma = z_{\mathbb{H}_n}(\gamma)$.

Let $U = (\mathsf{C_T}, \mathsf{C_t}, \mathsf{C}_{\hat{A}}, \mathsf{C}_{\hat{B}}, \mathsf{C}_{\hat{m}}, \pi_{\mathsf{qa}}, \beta, \gamma, \eta)$. One can see that elements of U are independent of each other and uniformly distributed. This is the same as the real distribution of an honest verifier communicating with a prover. We now show that U uniquely decides the rest of the elements in the proof. The simulator then computes the following with constant cost:

$\mathsf{C}_{Q_{\hat{A}}} \leftarrow (A(s)\mathsf{C_T} + \beta \mathsf{C}_{\hat{A}} - \mathsf{C}_{\hat{m}}) z_{\mathbb{H}_N}(s)^{-1}$.

$B_\gamma \leftarrow \hat{B}(\gamma)$.

$\mathsf{C}_{Q_{\hat{B}}} \leftarrow \left(\hat{B}(s)(\mathsf{C_t} + [\beta]_1) - [1]_1\right) z_{\mathbb{H}_n}(s)^{-1}$.

$\mathsf{C}_P \leftarrow \frac{[\hat{B}(s) - B_\gamma]_1 + \eta\left(B_\gamma(\mathsf{C_t} + [\beta]_1) - [1]_1 - z_{\mathbb{H}_n}(s)\mathsf{C}_{Q_{\hat{B}}}\right) + \eta^2([z_{\mathbb{H}_n}(s) - Z_\gamma]_1)}{s - \gamma}$.

Apparently $(\mathsf{C}_{Q_{\hat{A}}}, B_\gamma, \mathsf{C}_{Q_{\hat{B}}}, \mathsf{C}_P)$ has the same distribution as the real transcripts, as they are also uniquely determined by U in real conversations.

Knowledge Soundness: We now follow the notations in VerCQ(), i.e., the proof is a tuple $(\mathsf{C}_{\hat{m}}, \mathsf{C}_{\hat{A}}, ..., Z_\gamma)$. By the Q-DLOG assumption and ideal testing lemma in cq [18], one can see that all group elements in the proof can be represented as a multi-variate polynomial over variables X, K_1, K_2, K_3 (corresponding to trapdoor $s, \mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3$). This is different from the original cq protocol because we've introduced extra trapdoor $\mathbf{k}$ in the QA-NIZK prover keys.

However, from QA-NIZK proof (Equation (2) in VerCQ) we know that $\mathsf{C}_{\hat{A}}$ and $\mathsf{C}_{\hat{B}}$ in the proof represents two univariate polynomials $\hat{A}(X)$ and $\hat{B}(X)$ of degree N and $\lceil n \rceil$. One could probably also prove that $\hat{m}(X)$ and $\hat{Q_B}(X)$ are univariate polynomials via QA-NIZK, but for saving prover cost, we will discharge them via formal proof below.

We first consider $\hat{Q_B}$ (i.e., the polynomial behind $\mathsf{C}_{Q_{\hat{B}}}$) in Check (3) of VerCQ. It is known (by assumption) that $\hat{t}$ is a univariate polynomial, but not necessarily for $\hat{Q_B}$. Following the same argument in [11], because η is a random verifier challenge, by Schwartz-Zippel, with high probability $(X - \gamma)$ divides (1) $\hat{B}(X) - B_\gamma$, (2) $z_{\mathbb{H}_n}(X) - Z_\gamma$, and (3) $D(X, K_1, K_2, K_3)$. Then there exists $P_1(X)$ s.t. $\hat{B}(X) - B_\gamma = P_1(X)(X - \gamma)$, i.e., $B_\gamma = \hat{B}(X) - P_1(X)(X - \gamma)$. One can also reach there exists $P_3(X)$ s.t. $Z_\gamma = z_{\mathbb{H}_n}(X) - P_3(X)(X - \gamma)$. Plug in the above into $D(X, K_1, K_2, K_3)$: there exists a $P(X, K_1, K_2, K_3)$ s.t. the following is true.

$$\hat{B}(X)(\hat{t}(X) + \beta) - 1 - z_{\mathbb{H}_n}(X)\hat{Q_B}(X, K_1, K_2, K_3) = P(X, K_1, K_2, K_3)(X - \gamma) \quad (2)$$

Observe the polynomial on the LHS of Equation 2, it is a committed polynomial before γ is sampled by the verifier. For any random γ it can be divided by $(X - \gamma)$, i.e., the LHS evaluates to 0 at γ. This implies that the LHS is a zero polynomial. We write $\hat{Q_B}(X, K_1, K_2, K_3) = \hat{Q_{B_1}}(X) + \hat{Q_{B_2}}(X, K_1, K_2, K_3)$ s.t. $\hat{Q_{B_2}}$ includes all terms with at least one of K_i variables appearing (thus, the constant term belongs to $\hat{Q_{B_1}}(X)$). Separating the terms over variables K_i, we have the following:

$$\hat{B}(X)(\hat{t}(X) + \beta) - 1 - z_{\mathbb{H}_n}(X)\hat{Q_{B_1}}(X) = 0 \quad (3)$$

$$z_{\mathbb{H}_n}(X)\hat{Q_{B_2}}(X, K_1, K_2, K_3) = 0 \quad (4)$$

From Equation 3 we reach:

$$\forall i \in [n] : \mathbf{b} = \frac{1}{\mathbf{t}_i + \beta} \quad (5)$$

We then apply the same proof technique to check (1) of VerCQ(). It is already known that $\mathsf{C}_{\hat{A}}$ hides univariate polynomials, but not for $\mathsf{C}_{Q_{\hat{A}}}$ and $\mathsf{C}_{\hat{m}}$. By AGM, the polynomial behind $\mathsf{C}_{Q_{\hat{A}}}$ can be written as $\hat{Q_A}(X) + \hat{Q'_A}(X, K_1, K_2, K_3)$ where $\hat{Q'_A}$ includes all terms with at least one of K_i variables appearing (thus, the constant term belongs to $\hat{Q_A}$). Similarly, $\mathsf{C}_{\hat{m}}$ can be split into two polynomials. Then from Equation (1) of VerCQ() we have:

$$\begin{aligned} & \hat{A}(X)(\hat{T}(X) + \beta) - (\hat{m}(X) + \hat{m'}(X, K_1, K_2, K_3)) \\ = \; & (\hat{Q_A}(X) + \hat{Q'_A}(X, K_1, K_2, K_3)) z_{\mathbb{H}_N}(X) \end{aligned} \quad (6)$$

For Equation 6 to hold, the polynomial of LHS-RHS must be a zero (multi-variate) polynomial. Then we have both of the following should be zero polynomial by separating the monorails by degrees on K_i variables.

$$\hat{A}(X)(\hat{T}(X) + \beta) - \hat{m}(X) - \hat{Q_A}(X) z_{\mathbb{H}_N}(X) \quad (7)$$

$$\hat{m'}(X, K_1, K_2, K_3) + \hat{Q'_A}(X, K_1, K_2, K_3) z_{\mathbb{H}_N}(X) \quad (8)$$

From Equation 7, we obtain: there exists $\mathbf{m}, \mathbf{A}$ for $\mathbf{T}$ s.t.

$$\forall 1 \le i \le N : \mathbf{A}_i = \frac{\mathbf{m}_i}{\mathbf{T}_i + \beta} \quad (9)$$

Then using the QA-NIZK proof in (2) of VerCQ() we reach the conclusion that the sum of $\mathbf{A}$ is the same as $\mathbf{B}$, which concludes the soundness proof. The knowledge extractor is provided directly by the AGM model. □

Concrete Cost: It takes n group operations to compute each of $[\hat{m}(s)]_1$, $[\hat{A}(s)]_1$, $[\hat{Q_A}(s)]_1$, and $[\hat{B}(s)]_1$. It takes $2n$ for π_{qa}, as $\mathbf{A}$ is sparse (i.e., it has no more than n non-zero elements). It takes $\lceil n \rceil$ to compute each of $[\hat{Q_B}(s)]_1$, and $[P(s)]_1$. The total prover cost is $6n + 2\lceil n \rceil$ group operations. The verifier cost is 7 pairings.

[7] Note that if the random challenge falls in $\mathbb{H}_n$, it might fail the check, with negligible probability. It can be improved to perfect completness by increasing the degree of check function by 1 using the techniques in ProtoStar [7].

[8] In our context, such an extra proof is available as $\mathsf{C_t}$ will be retrieved from commit-and-prove scheme for individual PM-regex patterns. It is also possible to extend our proof at slight additional cost: n more group operations at prover and 1 more pairing by verifier, via adding one more row to encode $\mathsf{C_t}$ in the QA-NIZK proof

A.4 Fold Prove

We briefly discuss the implementation details and cost. Formally, the operation is defined as:

$$\pi \leftarrow \texttt{FoldProveCQ}(\mathsf{pk}_{\mathbf{U}}, \mathsf{pk}_{\mathbf{V}}, \mathbf{u}, \mathbf{v}, \alpha)$$

It produces a zero knowledge proof π for $\mathbf{u}+\alpha\mathbf{v} \hat{\subset} \mathbf{U}+\alpha\mathbf{V}$. Looking in the perspective of a "single" lookup argument, $\mathbf{T} = \mathbf{U} + \alpha\mathbf{V}$ and $\mathbf{t} = \mathbf{u}+\alpha\mathbf{v}$. We briefly discuss each of the steps of the prove function.

The calculation of $\mathbf{m}$ can be applied to just $(\mathbf{U}, \mathbf{u})$, as the folding require that $\mathbf{m}$ would be the same for $(\mathbf{V}, \mathbf{v})$. Compute $\mathbf{A}_i \leftarrow \mathbf{m}_i/(\mathbf{U}_i + \alpha\mathbf{V} + \beta)$ and similarly for $\mathbf{B}_i$. Both can be computed using the same cost as the original `Prove`. The computation of $\mathsf{C}_{\hat{A}}$, $\mathsf{C}_{\hat{B}}$, $\mathsf{C}_{Q_{\hat{B}}}$, C_P are also the same.

The tricky part is how to compute $\mathsf{C}_{Q_{\hat{A}}}$. Consider how the quotient polynomials are computed:

$$L_{N,i}(X)U(X) = Q_{i,U}(X)z_{\mathbb{H}_N}(X) + R_{i,U}(X)$$
$$L_{N,i}(X)V(X) = Q_{i,V}(X)z_{\mathbb{H}_N}(X) + R_{i,V}(X)$$

They are homomorphic and quickly leads to:

$$L_{N,i}(X)(U(X) + \alpha V(X)) = (Q_{i,U}(X) + \alpha Q_{i,V}(X))z_{\mathbb{H}_N}(X) + (R_{i,U}(X) + \alpha R_{i,V}(X))$$

Then $[Q_i(s)]_1$ can be computed as $[Q_{i,U}(s)]_1 + \alpha[Q_{i,V}(s)]_1$, and then based on which $[Q_{\hat{A}}]_1$ can be computed.

A.5 Support of Halo2

We integrate the zk-cq protocol with Halo2 [43], and provides a fork of the Halo2-PSE-fork (replacing IPA with KZG commitment schemes), where the presented zk-cq protocol is available, following a similar interface of the halo2 lookup (plookup) in the system.

To be compatable with Halo2, we need to support the fact each column (including the compressed input column) of CQ has t blinder items for zero knowledge. These t blinder items are located at the end of the column, and they do not have to belong to the lookup table. In this case, Algorithm 2 can be updated very easily: Simply replace the $z_{\mathbb{H}_n}(X)$ in step (8) by $z_{\mathbb{H}_{n-t}}(X)$. In addition, in Halo2, as blinding factors are used at the end of witness columns, there is no need to apply the $rz_{\mathbb{H}_N}(X)$ approach (just need to add one more blinding factor), which reduces the size of FFT domain.