Lookup Arguments: Improvements, Extensions and Applications to Zero-Knowledge Decision Trees

Matteo Campanelli[1], Antonio Faonio[2], Dario Fiore[3], Tianyu Li[4], and Helger Lipmaa[4]

[1] Protocol Labs matteo@protocol.ai
[2] EURECOM faonio@eurecom.fr
[3] IMDEA Software Institute dario.fiore@imdea.org
[4] Delft University of Technology tianyu.li@tudelft.nl
[5] University of Tartu helger.lipmaa@gmail.com

Abstract. Lookup arguments allow to prove that the elements of a committed vector come from a (bigger) committed table. They enable novel approaches to reduce the prover complexity of general-purpose zkSNARKs, implementing "non-arithmetic operations" such as range checks, XOR and AND more efficiently. We extend the notion of lookup arguments along two directions and improve their efficiency: (1) we extend vector lookups to matrix lookups (where we can prove that a committed matrix is a submatrix of a committed table). (2) We consider the notion of zero-knowledge lookup argument that keeps the privacy of both the sub-vector/sub-matrix and the table. (3) We present new zero-knowledge lookup arguments, dubbed cq+, zkcq+ and cq++, more efficient than the state of art, namely the recent work by Eagen, Fiore and Gabizon named cq. Finally, we give a novel application of zero-knowledge matrix lookup argument to the domain of zero-knowledge decision tree where the model provider releases a commitment to a decision tree and can prove in zero-knowledge statistics over the committed data structure. Our scheme based on lookup arguments has succinct verification, prover's time complexity asymptotically better than the state of the art, and is secure in a strong security model where the commitment to the decision tree can be malicious.

1 Introduction

General-purpose zero-knowledge succinct arguments of knowledge (zkSNARKs) promise to efficiently and succinctly prove any kind of NP-statement while keeping privacy, integrity and verifiability guarantees. Thanks to their generality, a great number of real-world applications can be performed with built-in security. The two-step recipe for building a brand new zero-knowledge application typically consists of first describing the application in a low-level constraint system (for example, Rank-1 Constraint System [4] or Plonk arithemization [18]) and then use the latest fully-developed zkSNARK as *backend*. Unfortunately, most

often, the *unfolded circuit* of the applications at hand become huge and, thus, the proving time could become unfeasible for real-world applications.

Lookup arguments [6,13,30,36,37] are a novel approach used to reduce the size of the unfolded circuits, bringing back to the real world many interesting applications. Briefly and very informally, a lookup argument allows to trade *sub-circuits* evaluations for lookup into their truth tables. For example, instead of having the n different sub circuits describing the computation of a hash function in the final unfolded circuit, the protocol designer could define n different *custom gates* that perform efficient lookup operations in the truth table of such a hash function. More concretely, lookup arguments are used in current zkSNARKs for representing "non-arithmetic operations", namely operations that cannot be expressed efficiently through the finite field operations supported by the zkSNARK, such as range checks, XOR and AND (see for example [6,17]). Very recently, the work of Arun, Setty and Thaler [3] shows how to use lookup arguments to create SNARKs for virtual-machine executions; namely a new SNARK scheme, called Jolt, that allows to verify the correct execution of a computer program specified with assembly language of the RISC-V instruction set architecture. Very informally, in Jolt, the truth table of each of the assembly instructions is encoded in a (predefined and highly structured) table, then lookup arguments enforce the correct instructions execution, namely the correct input-output behavior described by their truth tables.

In this work, we advance on lookup arguments in multiple ways. We propose new lookup arguments that improve over the state of the art [13]. One of our schemes enjoy, almost for free, of an extended notion of zero-knowledge, that we call fully zero-knowledge, which protects the privacy for arbitrary commitments to the tables. Orthogonally, we consider two natural extensions from vectors to matrices and give constructions for such extensions. Finally, we motivate the extensions to matrix and to fully zero-knowledge by giving a new application to privacy-preserving machine learning that crucially relies on them.

New Lookup Arguments based on cq. In a lookup argument, the prover aims to show that each coefficient of a (short) committed vector $\mathbf{f}$ of size n belongs to the (large) table $\mathbf{t}$ of size $N \gg n$. Since $N \gg n$, one of the desiderata of lookup arguments is that the prover's computation does not depend on N. Following a fast-pace line of recent works, Eagen, Fiore, and Gabizon [13] proposed an efficient lookup argument called cq (cq for *cached quotients*). Notably, cq's prover's computation is quasi-linear in n, while the proof size and verifier's computation are constant (e.g., proofs are 3840 bits, when using the standard BLS12-381 elliptic curve). In spite of appearing nearly optimal in efficiency, cq comes with two shortcomings. The first one is that it is not designed to have zero knowledge in mind. The second, more technical, one is that its use in larger protocols likely requires additional proof elements and pairing computations.[6] In this work, we propose a new lookup argument, dubbed cq$^+$, that addresses

[6] This is due to the fact that cq assumes an SRS of the same size as the table $\mathbf{t}$, and this allows avoiding a degree check. This condition, though, is often not guaranteed (e.g., in a SNARK for constraint systems larger than such a table).

all these shortcomings of cq and even achieves better efficiency. Namely, cq^+ achieves (standard) zero-knowledge at no overhead: it has the same prover's computation of cq and shorter proofs (3328 bits, and 2944 bits without ZK). Additionally, we consider two variations of cq^+: the first, dubbed zkcq^+, is fully zero-knowledge while the second, dubbed cq^{++}, has shorter proofs. Both schemes require in verification only one pairing computation more than cq^+.

Lookup Arguments for Matrices. A lookup argument could be used to show that a database $\mathbf{f}$ is a selection of the rows of a database $\mathbf{t}$. However, to naively use lookup arguments for such an application, each row of the database must be efficiently encoded in one single field element (supported by the lookup argument). In this paper, we extend the notion of lookup arguments from vectors to matrices in two natural ways. In particular, we consider an extension of the vector commitment of Kate *et al.* [24] (also known as KZG commitment scheme) to matrices. We then show two lookup arguments for matrices that internally calls a lookup argument for KZG vector commitments. (We find this modularity useful given the current fast pace of the research on lookup arguments.) The first scheme allows to prove that a committed database $\mathbf{f}$ is a selection of the rows of a committed database $\mathbf{t}$, while the second scheme allows to prove that $\mathbf{f}$ is a selection of a projection of a database $\mathbf{t}$.

A New Approach to Zero-Knowledge for Decision Trees. Our third contribution is an application of zero-knowledge matrix lookup arguments which exemplifies how a mix of specialized argument systems and general-purpose ones can substantially improve prover time complexity. We improve over the framework of zero-knowledge for decision trees of Zhang *et al.* [38], which showed zkSNARKs for evaluations of committed decision trees and zkSNARKs for accuracy of committed decision trees. The former kind of zero-knowledge protocols can prove that a committed decision tree T, on input a vector $\mathbf{x}$, outputs a label v, while the latter schemes enable to validate the accuracy (namely, the ratio of true positives) of a decision tree on a given dataset.

Our framework can instantiate different kinds of statistics over committed decision trees, which include evaluation and accuracy. The design of our scheme decouples the computation of the committed decision tree and the performed statistics. This allows for a plug-and-play approach. As for security, we extend the notion of security from [38] considering possibly maliciously generated commitments to decision trees.

Our Contributions. We can summarize our contributions as follows:

1. A zero-knowledge lookup argument that improves the state of the art for arbitrary tables [13].
2. A construction for zero-knowledge matrix lookup argument based on zero-knowledge (vector) lookup arguments for KZG-based vector commitment.
3. A new paradigm for proving decision tree classification in zero-knowledge. We can instantiate our paradigm with our matrix lookup arguments and obtain speedups of two orders of magnitude for proving time compared to previous work.

4. Strengthening the security model for zero-knowledge decision trees: we formalize a setting where the commitment to a decision tree may not be trusted.

1.1 Technical Overview

Our Zero-Knowledge Lookup Arguments. Similarly to cq, cq$^+$ uses the technique of logarithmic derivates of Haböck [22]. However, we diverge from cq early, introducing several novel ideas that allow us to improve on cq's efficiency. One of the differences is that, while cq uses Aurora's sumcheck [5] twice, our cq$^+$ only runs it once. Nicely, this technique allows us to kill two birds with one stone, in fact, cq$^+$ does not require any additional low-degree tests. We defer the reader to Section 4.1 where we give a more detailed technical overview and Appendix C for the formal proofs.

Matrix lookup from vector lookup. To succinctly commit to a matrix we can commit the concatenation of the rows of the matrix. Our matrix lookup argument first needs to label all the entries of such a vectorization with the coordinate that identifies the position of each cell of the sub-matrix $\mathbf{F}$ into the bigger table $\mathbf{T}$. Similarly, in the precomputation phase, it needs to label each of the cell in the big table $\mathbf{T}$ with its coordinate. To prove that the k-th row of $\mathbf{F}$ appears in $\mathbf{T}$ we show that the *labelled* matrix $\mathbf{F}^* = (i_j, j, \mathbf{F}_{k,j})_{j\in[d]}$ is a sub-matrix of labeled table $\mathbf{T}^* = (i, j, \mathbf{T}_{i,j})_{i,j}$ and that $i_1 = i_2 = \cdots = i_d$ (in particular $i_j = k$), where d is the number of columns of the matrices. Notice that the first claim can be proved efficiently with a (non-succinct) matrix commitment for matrices with $N \cdot d$ rows and 3 columns following techniques from [6], while the second claim can be efficiently expressed through polynomial equations following techniques from [9]. In particular, for the first part, given a challenge $\rho \leftarrow \mathbb{F}$ the prover can hash $h(\mathbf{F}^*) = \sum_{i=1}^{3} \rho^{i-1} \cdot \mathbf{F}_i^*$ to a single column (where $\mathbf{F}_i^*$ are the columns of $\mathbf{F}^*$). Since $h(\cdot)$ is an universal hash function, if $h(\mathbf{F}^*)$ is a subvector of $h(\mathbf{T}^*)$ then with overwhelming probability $\mathbf{F}^*$ is a submatrix of $\mathbf{T}^*$, thus reducing matrix lookup argument to vector lookup. For the second part, we notice that the first column $\mathbf{F}_1^*$ of $\mathbf{F}^*$ is a *step* function, thus we first commit to the shift of $\mathbf{F}_1^*$ and then show that the difference between the shifted column and the column $\mathbf{F}_1^*$ is a function that has zeros in well-defined positions. More details in Section 5.2. We can go even further and prove that a matrix $\mathbf{F}$ with d' columns and $d' < d$ is a submatrix of $\mathbf{T}$. As before, we set $\mathbf{F}^* = (i, j, \mathbf{F}_{i,j})_{i\in R, j\in D}$ for subset $R = \{r_1, \ldots, r_{d'}\} \subset [N]$ and $D \subset [d]$ and we additionally show, using the technique of the shifted polynomials, that $\mathbf{F}^*_{2,id'+j} = \mathbf{F}^*_{2,(i+1)d'+j} = r_j$ for any i and j. More details in Appendix D.

Both of our compilers preserve quasi-linear running time in n thanks to the linear homomorphic property of KZG commitments.

Our Approach to ZK for Decision Trees. A decision tree is an algorithm that performs a sequence of adaptive queries reading from its input and eventually it outputs a value. At each query the algorithm moves from a node in the tree to one of its children, the output is defined by the label of the reached leaf.

Two important parameters are the total number of nodes N_{tot} and the number of features d of the inputs. Following the work of Chen *et al.* [10], we can efficiently (although redundantly) encode a decision tree as a matrix with N_{tot} rows and $2d+1$ column. An evaluation of a decision tree under this alternative representation consists in locating the row corresponding to the correct leaf and then by showing that the input vector matches all the constraints described by such a row. Thus, we can commit to a decision tree by committing to its matrix encoding and, to prove correct evaluation, we can commit to the single row corresponding to the correct leaf and prove with a matrix lookup argument that the committed row is indeed a leaf of the committed decision tree. Once isolated such a row, we can then prove that the input vector matches all the constraints described by the row. Notice that our strategy scales well with the number of different evaluations. In fact, to prove statements which involve multiple input vectors for the decision tree, we can commit at proving time to a matrix whose rows correspond to the entries of the leaves reached by the evaluations (instead of committing to a single row). Thanks to the efficiency property of the matrix lookup argument, the prover time complexity is independent of the size of decision tree.

Beyond a Trusted Commitment to the Tree. A malicious committer could commit to a matrix that contains a row that matches a leaf with label, let say, 0, and another row that matches the same leaf but where it maliciously assigns the label 1. Now, such a bogus commitment to a decision tree could allow the malicious prover to show both $\mathsf{T}(\mathbf{x}) = 0$ and $\mathsf{T}(\mathbf{x}) = 1$. The problem is that the committed matrix does not *encode* a decision tree. To solve this problem we show a set of sufficient algebraic conditions (cf. Section 6.2) for a matrix to *encode* a decision tree. We can check efficiently these algebraic conditions through a general-purpose zkSNARKs for R1CS (see for example [5,7,20,27,29,31,32]). However, the number of constraints is $O(dN_{\mathsf{tot}}^2)$ and thus the prover time complexity is quadratic in the number of nodes. The algebraic constraints we propose are essentially linear equations between matrices and Hadamard-product equations, which are the kind of equation checks performed in a R1CS-based zkSNARKs. In fact, if we gave up on the privacy of the decision tree[7] we could define an R1CS circuit that depends on the tree-structure of the decision tree and we would go down to $O(dN_{\mathsf{tot}})$ number of constrains. We show that we can restore zero-knowledge using this approach, by privately committing to such an R1CS-like circuit and prove in zero-knowledge that the circuit belongs to a well-defined family of circuits (defined by the algebraic constraints in Section 6.2). In particular, we use the techniques from Zapico *et al.* [36] for committing to a *basic* matrix, namely a matrix whose rows are elementary vectors, and to prove its basic-matrix structure and the permutation argument from Plonk [18] to prove the rows of the matrix are all different.

[7] Specifically, giving up only to the privacy of the *structure of* the decision tree while keeping private the values of the thresholds and labels

1.2 Related Work

Lookup Arguments. Lookup arguments were introduced by Bootle, Cerulli, Groth, Jakobsen and Maller [6]. The state-of-art for lookup arguments for arbitrary tables[8] is the recent work of Eagen, Fiore, Gabizon [13] named cq and based on the technique of logarithmic derivates of Haböck [22]. cq has prover complexity proportional only to the size of the smaller vector and independent of the bigger table assuming pre-processing for table. To our knowledge, all lookup arguments with similar efficiency properties are based on the Kate *et al.* (commonly known as KZG) polynomial commitment scheme [24]. Among these we mention Caulk+ by Posen and Kattis [30] (based on Caulk [36] by Zapico *et al.*) and Baloo [37] by Zapico *et al.*. The latter work introduces the notion of Commit-and-Prove Checkable Subspace Argument (extending over [31]) that we use for our (extractable) commitment scheme (cf. Section 6.3).

Comparison with [13]. As previously mentioned, we diverge from cq introducing several novel ideas that allow us to improve on cq's efficiency. As the end result, cq^+'s communication is about 14% (or even 23% in a variant without the ZK) better than cq's. All other efficiency parameters of cq^+ are similar to cq's. Moreover, we propose cq^{++}, a batched variant of cq^+. cq^{++} saves 23% (or 33%, in a variant without ZK) communication compared to cq. A slight drawback of cq^{++} is that the verifier has to execute one more pairing. We emphasize that cq is already almost optimally efficient, and thus improving on it is non-trivial.

Concurrent Work. Choudhuri *et al.* [12] very recently introduced the notion of *segment lookup arguments* which, besides some syntactical differences, matches the simpler of our notions of matrix lookup arguments. Additionally, in [12] they show, in our lingo, a matrix lookup argument based on cq. Their matrix lookup argument is less efficient than ours; we defer to Table 1 for more details. Interestingly, in the same paper, the authors build a general-purpose zkSNARK based on Plonk and matrix lookup which they call Sublonk, showing another application for matrix lookup arguments. The main feature of Sublonk is that the prover's running time grows with the size of the *active part* of the circuit, namely the part of the circuit activated by its execution on a given instance. Sublonk makes black-box use of the underlying matrix lookup argument, therefore we could plug-in our matrix lookup argument obtaining a more efficient version of Sublonk.

Privacy-Preserving Machine Learning. We focus on the related work on zero-knowledge proofs for decision trees and, more in general, for machine learning algorithms. The main related work for decision trees is the paper of Zhang *et al.* [38], where they introduce the notions of zero-knowledge proofs for decision

[8] Very recently, Setty, Thaler and Wahby [33] introduced a new lookup argument for a restricted subclass of tables. Their work is extremely efficient, and in particular more efficient than cq, for such a restrict class of tables. On the other hand, cq can handle arbitrary tables, for this reason we refer to cq as the state-of-art for arbitrary tables.

tree predictions and accuracy. Besides decision trees, zero-knowledge proofs and verifiable computation for machine learning is a vibrant area of research (see for example [1,15,23,25,28,34,35]).

Comparison with [38]. Briefly, the main techniques of [38] consist of an authenticated data structure for committing to decision trees and highly-tuned R1CS circuits to evaluate the authenticated data structure in zero-knowledge. More in detail, they commit to a decision tree with a *labelled Merkle Tree* whose labelled nodes are the nodes of the decision tree. This commitment scheme is binding and hiding and allows for path openings (with proof size proportional to the length of the path). On top of this authenticated data structure, they use general-purpose zkSNARKs for R1CS to prove, for example, the knowledge of a valid opening for a path and the labelling of the leaf. While the basic ideas are simple, the paper needs to solve many technical details and presents many optimizations which are necessary to obtain a practical scheme. The *backend* general-purpose zero-knowledge scheme they use is Aurora [5]. Thanks to this choice and because of the Merkle-Tree approach, their zero-knowledge scheme has a transparent setup and is presumably post-quantum secure.

Their security model stipulates that the decision tree is adversarially chosen, but the commitment to such a decision tree is honestly generated. On the other hand, in our security model, we require the commitment scheme to be extractable, thus allowing for maliciously generated commitments. We notice that, besides improving security, our definitional choices allow for more efficient design. In fact, the (proof for the) extractable commitment is generated only once, let say in an offline phase, while the (multiple) proofs of evaluation, in the online phase, can leverage the extra security properties offered by the extractable commitment and thus be faster.

For comparison with our work, we consider the extractability of their scheme for decision tree evaluation. This is not immediate: the main reason is that the witness for the zkSNARK is a single path from the root to the evaluated leaf (which could be extracted) while, to obtain our notion of extractability, it would be required to extract the full decision tree. Additionally, their authenticated data structure could allow to commit (and prove statements) to 2-fan-in direct-acyclic graphs (DAGs), which are more general than trees [9]. We believe their second scheme for the accuracy of decision trees can be proved secure in our model. In fact, proposed as an efficiency optimization, their second scheme computes a consistency check over the full decision tree. Thanks to this, we could extract the full tree from the zkSNARK. We also believe that our techniques could be integrated into theirs. Our approach separating the extractable commitment from the "online-stage" of the zero-knowledge proof could be adapted to their scheme for accuracy (thus improving its efficiency). Interestingly, by using our approach, their scheme could be interpreted as an application of a lookup argument based on [6] and [5] to decision trees. The main difference is this: our scheme runs lookup arguments over the leaves associated with the evaluation

[9] We believe that this does not pose any problems neither for correctness nor for soundness, as indeed, one could argue this is a feature rather than a bug.

vectors, while the scheme in [38] requires lookups for paths from the root to the leaves associated with the evaluation vectors.

For other points of comparison efficiency-wise (we refer the reader to Section 6.5 for more details), we mention that their commitments require hashing only, while ours requires multiexponentations in a group. Therefore their commitment stage is faster than ours. Our proof size is concretely smaller (few kilobytes vs hundreds of kilobytes). To compare proving time, we start from observing the asymptotic advantages of our solutions: their prover is linear in the size of the tree and in the complexity of a hash function; ours is sublinear in all these dimensions. This results in *concretely* faster proving times *despite* the fact that our prover requires group operations and theirs only field operations. This is a consequence of removing the constants deriving from the hash function size, the sublinearity in the tree and of the efficient lookup argument instantiations[10]. Our improvements also translate to a better verification time. Our estimates show improvements of almost one order of magnitude for proving time (regardless of the underlying backend proof systems for [38]; see Appendix A) and two orders of magnitude for verification time.

2 Preliminaries

We denote matrices with capital and bold, for example, $\mathbf{M}$, and vectors with lowercase and bold, for example, $\mathbf{v}$. We denote with $\circ$ the Hadamard product between two matrices/vectors of the same size, while $\cdot$ is reserved for the matrix-vector/vector-vector multiplication. Given two vectors $\mathbf{a}, \mathbf{b}$ we define $\mathbf{a} < \mathbf{b}$ if and only if $\forall i : \mathbf{a}_i < \mathbf{b}_i$ (and similarly for $\leq$). We denote $\|$ the concatenation by columns of two matrices. We denote by $\mathbb{F}$ a finite field, by $\mathbb{F}[X]$ the ring of univariate polynomials, and by $\mathbb{F}_{<d}[X]$ (resp. $\mathbb{F}_{\leq d}[X]$) the set of polynomials in $\mathbb{F}[X]$ of degree $< d$ (resp. $\leq d$). For any subset $S \subseteq \mathbb{F}$, we denote by $\nu_S(X) \stackrel{\text{def}}{=} \prod_{s \in S}(X - s)$ the *vanishing polynomial* of S, and by $\lambda_s^S(X)$ the *s-th Lagrange basis polynomial*, which is the unique polynomial of degree at most $|S| - 1$ such that for any $s' \in S$, it evaluates to 1 if $s = s'$ and to 0 otherwise. Any multiplicative subgroup of a finite field is cyclic. Thus, given a group $\mathbb{H}$, we can find an element ω that generates the subgroup $\mathbb{H}$. For convenience, given a subgroup $\mathbb{H}$ of order n we denote with ω_n a fixed generator of $\mathbb{H}$. If $\mathbb{H} \subseteq \mathbb{F}$ is a multiplicative subgroup of order n, then its vanishing polynomial has a compact representation $\nu_{\mathbb{H}}(X) = (X^n - 1)$ and $\lambda_i^{\mathbb{H}}(X) = \nu_{\mathbb{H}}(X)\omega_n^{i-1}/(n(X - \omega_n^{i-1}))$. Both $\nu_{\mathbb{H}}(X)$ and $\lambda_i^{\mathbb{H}}(X)$ can be evaluated in $O(\log n)$ field operations. For any vector $\mathbf{v} \in \mathbb{F}^n$, we denote by $v_{\mathbb{H}}(X)$ the *low degree encoding* (LDE) in $\mathbb{H}$ of $\mathbf{v}$, i.e., the unique degree-$(|\mathbb{H}| - 1)$ polynomial such that, $v_{\mathbb{H}}(\omega_n^{i-1}) = v_i$, when the subgroup $\mathbb{H}$ is clear from the context, we simply write $v(X)$. Similarly, we consider

[10] As a bottleneck, the dependency [38] has on the hash function is one that is hard to remove . Applying a hash function optimized for SNARK constraints, e.g. the one we used to experimentally run [38]—SWIFFT—*nonetheless* yields high constants in practice *regardless* of the proof system used as a backend.

the k-degree *randomized low-degree encoding* (RLDE) in $\mathbb{H}$ of a vector $\mathbf{v} \in \mathbb{F}^n$ to be a randomized polynomial of the form $\hat{v}_{\mathbb{H}}(X) = v_{\mathbb{H}}(X) + \nu_{\mathbb{H}}(X)\rho_v(X)$ for a random polynomial ρ_v of degree k. Sometimes, we will not explicitly mention the degree of the randomizer. In this case, the reader should assume that the degree is set to be the minimum degree necessary to keep zero-knowledge of v in the presence of evaluations (on points outside of $\mathbb{H}$) of the polynomial $\hat{v}$.

A type-3 bilinear group $\mathbb{G}$ is a tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2)$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order q, the elements P_1, P_2 are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ 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 $\mathbb{G}_i, i \in \{1, 2, T\}$ are denoted in implicit notation as $[a]_i := aP_i$, where $P_T := e(P_1, P_2)$.

2.1 Commit-and-Prove SNARKs

A commitment scheme is a tuple of algorithm $\mathsf{CS} = (\mathsf{KGen}, \mathsf{Com})$ where the first algorithm samples a commitment key ck and the second algorithm, upon input of the commitment key, a message p and opening material ρ, outputs a commitment c. The basic notions of security for the commitment scheme are (perfect) *hiding* and (computational) *binding*. The former property states that no (unbounded) adversary can distinguish commitments of two different messages when the opening materials are sampled at random from their domain, the latter property states that no (polynomial time) adversary, upon input of the commitment key, can find two different messages and two opening materials that commit to the same commitment.

Following Groth et al. [21], we define a relation $\mathcal{R}$ verifying triple $(\mathsf{pp}; x; w)$. We say that w is a witness to the instance x being in the relation defined by the parameters pp when $(\mathsf{pp}; x; w) \in \mathcal{R}$ (equivalently, we sometimes write $\mathcal{R}(\mathsf{pp}; x; w) = 1$). For example, the parameters pp could be the description of a bilinear group, or additionally contain a commitment key for a commitment scheme or a common reference string. Whenever it is clear of the context, we will write $\mathcal{R}(x; w)$ as a shortcut for $\mathcal{R}(\mathsf{pp}; x; w)$.

Briefly speaking, Commit-and-Prove SNARKs (CP-SNARKs) are zkSNARKs whose relations verify predicates over commitments [8]. Given a commitment scheme CS, we consider relations $\mathcal{R}$ whose instances are of the form $x = ((\mathsf{c}_j)_{j\in[\ell]}, \hat{x})$, where we can un-ambiguously parse the witness $w = ((p_j)_{j\in[\ell]}, (\rho_j)_{j\in[\ell]})$ for some $\ell \in \mathbb{N}$ with $\forall j : p_j$ is in the domain of a commitment scheme CS, and such that there exists a PT relation $\hat{\mathcal{R}}$ such that let $\hat{w} = (p_j)_{j\in[\ell]}$:

$$\mathcal{R}(\mathsf{pp}; x; w) = 1 \iff \hat{\mathcal{R}}(\mathsf{pp}; \hat{x}; \hat{w}) = 1 \wedge \forall j \in [\ell] : \mathsf{c}_j = \mathsf{Com}(\mathsf{ck}, p_j, \rho_j).$$

We refer to a relation $\hat{\mathcal{R}}$ as derived above as a *Commit-and-Prove* (CP) relation. Given a CP-relation $\hat{\mathcal{R}}$ and a commitment scheme CS we can easily derive the *associated* NP-relation $\mathcal{R}$. Instances of NP-relations may contain only commitments, therefore using the notation above, the instances of the associated

CP-relation are empty strings ε, namely, $\hat{\mathcal{R}}$ is a predicate over the committed witness. To avoid cluttering the notation, in these cases, we may omit the (empty) instance and simply write $\hat{\mathcal{R}}(\mathsf{pp}, \hat{w})$.

A CP-SNARK for $\hat{\mathcal{R}}$ and commitment scheme CS is a zkSNARK for the associated relation $\mathcal{R}$ as described above. More in detail, we consider a tuple of algorithms $\mathsf{CP} = (\mathsf{KGen}, \mathsf{Prove}, \mathsf{Verify})$ where:

- $\mathsf{KGen}(\mathsf{ck}) \to \mathsf{srs}$ is a probabilistic algorithm that takes as input a commitment key ck for CS and it outputs $\mathsf{srs} := (\mathsf{ek}, \mathsf{vk}, \mathsf{pp})$, where ek is the evaluation key, vk is the verification key, and pp are the parameters for the relation $\mathcal{R}$ (which include the commitment key ck).
- $\mathsf{Prove}(\mathsf{ek}, x, w) \to \pi$ takes an evaluation key ek, a statement x, and a witness w such that $\mathcal{R}(\mathsf{pp}, x, w)$ holds, and returns a proof π.
- $\mathsf{Verify}(\mathsf{vk}, x, \pi) \to b$ takes a verification key, a statement x, and either accepts ($b = 1$) or rejects ($b = 0$) the proof π.

In some cases, the KGen algorithm would simply (and deterministically) re-parse the commitment key ck information. In these cases we might omit KGen and refer to the CP-SNARK as a tuple of two algorithms.

Zero-Knowledge in the SRS (and RO) model. The zero-knowledge simulator $\mathcal{S}$ of a CP-SNARK is a stateful PPT algorithm that can operate in three modes. $(\mathsf{srs}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0, 1^{\lambda}, d)$ takes care of generating the parameters and the simulation trapdoor (if necessary). $(\pi, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, x)$ simulates the proof for a statement x. $(a, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s)$ takes care of answering random oracle queries. The state $\mathsf{st}_{\mathcal{S}}$ is shared and updated after each operation.

Similarly to [14,19], we define the following wrappers.

Definition 1 (Wrappers for ZK Simulator). *The following algorithms are stateful and share their state* $\mathsf{st} = (\mathsf{st}_{\mathcal{S}}, \mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}})$ *where* $\mathsf{st}_{\mathcal{S}}$ *is initially set to be the empty string, and* $\mathcal{Q}_{\mathsf{sim}}$ *and* $\mathcal{Q}_{\mathsf{RO}}$ *are initially set to be the empty sets.*

- $\mathcal{S}_1(x, w)$ *denotes an oracle that first checks* $(\mathsf{pp}, x, w) \in \mathcal{R}$ *where* pp *is part of* srs *and then runs the first output of* $\mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, x)$.
- $\mathcal{S}_2(s)$ *denotes an oracle that first checks if the query* s *is already present in* $\mathcal{Q}_{\mathsf{RO}}$ *and in case answers accordingly, otherwise it returns the first output* a *of* $\mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s)$. *Additionally, the oracle updates the set* $\mathcal{Q}_{\mathsf{RO}}$ *by adding the tuple* (s, a) *to the set.*

Definition 2 (Zero-Knowledge). *We say that a CP-SNARK* CP *for a CP-relation* $\hat{\mathcal{R}}$ *and commitment scheme* CS *is (perfect) zero-knowledge if there exists a PPT simulator* $\mathcal{S}$ *such that for all adversaries* $\mathcal{A}$ *and for all* $d \in \mathbb{N}$*:*

$$\Pr\left[\begin{array}{l}\mathsf{ck} \leftarrow \mathsf{CS.KGen}(1^{\lambda}, d)\\ \mathsf{srs} \leftarrow \mathsf{CP.KGen}(\mathsf{ck})\\ \mathcal{A}^{\mathsf{Prove}(\mathsf{srs},\cdot,\cdot)}(\mathsf{srs}) = 1\end{array}\right] \approx \Pr\left[\begin{array}{l}(\mathsf{srs}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})\\ \mathcal{A}^{\mathcal{S}_1(\cdot,\cdot)}(\mathsf{srs}) = 1\end{array}\right]$$

Knowledge Soundness. Our definition of knowledge soundness is in the algebraic group model [16]. An algorithm $\mathcal{A}$ is called *algebraic* if for all group elements that $\mathcal{A}$ outputs, it additionally provides the representation relative to all previously received group elements. That is, if elems is the list of group elements that $\mathcal{A}$ has received, then for any group element z in output, the adversary must also provide a vector $\mathbf{r}$ such that $\mathsf{z} = \langle \mathbf{r}, \mathsf{elems} \rangle$. We define the notion of knowledge soundness in the algebraic model.

Definition 3 (Knowledge Soundness in the AGM). *A CP-SNARK is knowledge extractable in the Algebraic Group Model if for any PT algebraic adversary, there exists a PT extractor $\mathcal{E}$ that receives in input the algebraic representations $\mathbf{r}_1, \ldots, \mathbf{r}_l$ of $\mathcal{A}$ and such that:*

$$\Pr\left[\begin{array}{l}\mathsf{ck} \leftarrow \mathsf{CS.KGen}(1^\lambda, d); \mathsf{srs} \leftarrow \mathsf{CP.KGen}(\mathsf{ck}); \\ (x, \pi, \mathbf{r}_1, \ldots, \mathbf{r}_l) \leftarrow \mathcal{A}(\mathsf{srs}); w \leftarrow \mathcal{E}(\mathsf{srs}, \mathbf{r}_1, \ldots, \mathbf{r}_l) \\ \mathsf{Verify}(\mathsf{srs}, x, \pi) \wedge \neg \mathcal{R}(\mathsf{pp}, x, w)\end{array}\right] \leq \mathsf{negl}(\lambda)$$

Indexed Relations and Universal CP-SNARKs. We extend the notion of relations to indexed relations [11]. We define a PT indexed relation $\mathcal{R}$ verifying tuple (pp, ind, x, w). We say that w is a witness to the instance x being in the relation defined by the pp and index ind when $(\mathsf{pp}, ind, x, w) \in \mathcal{R}$ (equivalently, we sometimes write $\mathcal{R}(\mathsf{pp}, ind, x, w) = 1$).

Briefly, we say that a CP-SNARK is *universal* if there exists a deterministic algorithm Derive that takes as input an srs and an index ind, and outputs a specialized reference string $\mathsf{srs}_{ind} = (\mathsf{vk}_{ind}, \mathsf{ek}_{ind})$ where vk_{ind} is a succinct verification key and ek_{ind} is a proving key for such an index. Moreover, we require that the verifier Verify (resp. the P) of an Universal CP-SNARK takes as additional input the specialized verification key vk_{ind} (resp. the specialized ek_{ind}). We refer to Appendix B for more details.

2.2 Extractable Commitment Schemes

An extractable commitment scheme for a domain $\mathcal{D} = \{\mathcal{D}_\lambda\}_\lambda$ is a commitment scheme equipped with a CP-SNARK that proves the knowledge of an opening of the commitments.

Definition 4. *Given a domain $\mathcal{D}$, $\mathsf{CS} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{VerCom})$ is an extractable commitment scheme for the domain $\mathcal{D}$ if there exist two algorithms $\mathsf{Com}', \mathsf{Prove}'$ such that $\mathsf{Com}(\mathsf{ck}, p, \rho)$ executes (1) $\mathsf{c} \leftarrow \mathsf{Com}'(\mathsf{ck}, p, \rho)$ and (2) $\pi \leftarrow \mathsf{Prove}'(\mathsf{ck}, \mathsf{c}, (p, \rho))$ and outputs (c, π), and $(\mathsf{Prove}', \mathsf{VerCom})$ is a CP-SNARK for the commitment scheme $(\mathsf{KGen}, \mathsf{Com}')$ and for the CP-Relation $\hat{\mathcal{R}}_{\mathsf{open}}$ defined below:*

$$\hat{\mathcal{R}}_{\mathsf{open}} = \{\mathsf{pp}; \varepsilon; p : p \in \mathcal{D}_\lambda\}.$$

2.3 Polynomial, Vector and Matrix Commitment Schemes

We use the polynomial commitment scheme of [24] described below:

$\mathsf{KGen}(1^\lambda, d_1, d_2)$ samples a type-3 pairing group with security level λ and outputs commitment key $\mathsf{ck} := ((\left[s^i\right]_1)_{i\in[d_1]}, (\left[s^i\right]_2)_{i\in[d_2]},)$ for random secrets $s \in \mathbb{Z}_q$.
$\mathsf{Com}(\mathsf{ck}, p)$ outputs $[p(s)]_1$.

We notice that the above commitment scheme is not hiding and it is extractable[11] for the domain of polynomial of degree d_1 in the algebraic group model of [16] under the power discrete logarithm assumption (PDL), which informally states that find s is hard given a freshly sampled commitment key, see Definition 12 for details.

The commitment scheme allows for a very efficient CP-SNARK $\Pi_{\mathsf{eval}} = (\mathsf{Prove}_{\mathsf{eval}}, \mathsf{Verify}_{\mathsf{eval}})$ for the CP-relation $\hat{\mathcal{R}}_{\mathsf{eval}} = \{(x, y; p) \colon p(x) = y\}$. In particular, the prover $\mathsf{Prove}_{\mathsf{eval}}$ upon input the SRS ck, an instance $([p(s)]_1, x, y)$ and the witness p, computes the unique polynomial w such that the equation below holds and outputs $[w(s)]_1$ as its proof:

$$p(X) = w(X) \cdot (X - x) + y.$$

On the other hand, the verifier $\mathsf{Verify}_{\mathsf{eval}}$ upon input the SRS ck, an instance (c, x, y) and a proof π, checks $e(\mathsf{c} - [y]_1, [1]_2) = e(\pi, [s - x]_2)$.

Vector and Matrix commitment schemes. From a polynomial commitment scheme, we can define a *vector* commitment. Specifically, let $\mathbb{H}$ be multiplicative subgroup of $\mathbb{F}$ with order N, and let ω_N be a fixed generator of $\mathbb{H}$. We can commit to vector $\mathbf{v}$ by committing to the low degree encoding of v over $\mathbb{H}$. Namely, $[v_{\mathbb{H}}(s)]_1$ is a commitment to $\mathbf{v}$. The commitment key should additionally contain the description of the subgroup $\mathbb{H}$ to allow for verification. Notice that such a commitment scheme is not hiding. However, we can make it hiding by committing to a randomized, low-degree encoding of v over $\mathbb{H}$ instead of its low-degree encoding. We can easily adapt the CP-SNARK for $\mathcal{R}_{\mathsf{eval}}$ to spot-opening of a committed vector.

We define the *vectorization* of a matrix $\mathbf{M} \in \mathbb{F}^{n\times d}$ to be the vector $\bar{\mathbf{m}} \in \mathbb{F}^{n\cdot d}$ which is the concatenation of the rows of $\mathbf{M}$. Namely, for any $i \in [n], j \in [d]$, we define $\bar{\mathbf{m}}_{d\cdot i+j} = \mathbf{M}_{i,j}$. To commit to a matrix $\mathbf{M}$, we commit to its vectorization $\bar{\mathbf{m}}$. Notice that, additionally, the commitment key should contain the values n and d, and the subgroup $\mathbb{H}$ should be of cardinality $n \cdot d$.

3 Zero-Knowledge Matrix Lookup Arguments

Given two vectors $\mathbf{f}, \mathbf{t}$ we say that $\mathbf{f}$ is a *sub-vector of* $\mathbf{t}$ if there exits a (multi) set $K = \{k_1, \ldots, k_n\}$ such that $\mathbf{f}_j = \mathbf{t}_{k_j}$ for any j. We write $\mathbf{f} \prec \mathbf{t}$ to denote that $\mathbf{f}$ is

[11] Technically, we can define a *vacuous* CP-SNARK for opening in the AGM where the prover does nothing and the verifier checks that the commitment is a valid group element.

a sub-vector of $\mathbf{t}$. Notice we diverge from the usual notion of sub-vector. Namely, we assume that a sub-vector $\mathbf{f}$ may contains multiple copies of an element in $\mathbf{t}$ and, moreover, any permutation of $\mathbf{f}$ is a sub-vector of $\mathbf{t}$. We extend the notion of sub-vectors to matrices, we say that a matrix $\mathbf{F} \in \mathbb{F}^{n \times d}$ is a (rows) sub-matrix of a matrix $\mathbf{T} \in \mathbb{F}^{N \times d}$ if $\mathbf{F}$ parsed as a $\mathbb{F}^d$-vector of length n is a sub-vector of $\mathbf{T}$ parsed as a $\mathbb{F}^d$-vector of length N. In other words, $\mathbf{F}$ is a matrix whose rows are also rows in $\mathbf{T}$. Similarly, given a multi set $K = \{k_1, \ldots, k_l\}$ we can define the sub-matrix $\mathbf{F}_{|K}$ as the sub-matrix of $\mathbf{F}$ which j-th row is the row $\mathbf{F}_{k_j}$. Notice that also our notion of sub-matrix is not standard, in fact, besides the differences mentioned for the notion of sub-vector, we consider the special case where the number of columns of $\mathbf{F}$ and $\mathbf{T}$ are the same. This is sufficient for our application, however, for completeness, in Appendix D.1 we consider the more general case where $\mathbf{F}$ may be a selection of a projection of $\mathbf{T}$. We call the latter the rows-columns sub-matrix relationship. We consider the following indexed CP-relation, where we will refer to $\mathbf{T}$ as the table and to $\mathbf{F}$ as the sub-vector (or sub-matrix):

$$\hat{\mathcal{R}}_{\mathsf{zklookup}} := \{\mathsf{pp}; (N, d, n); \varepsilon; (\mathbf{T}, \mathbf{F}) : \mathbf{F} \prec \mathbf{T}, |\mathbf{T}| = N \times d, |\mathbf{F}| = n \times d\}, \quad (1)$$

Previous work focuses on $d = 1$, namely the lookup argument for vector commitments, and where the table $\mathbf{T}$ is public. Moreover, some of the previous work did not focus on zero-knowledge. Namely, previous work focused on (ZK or not) CP-SNARKs for the following CP-relation:

$$\hat{\mathcal{R}}_{\mathsf{lookup}} := \{\mathsf{pp}; (\mathbf{t}, n); \varepsilon; \mathbf{f} : \mathbf{f} \prec \mathbf{t}, |\mathbf{f}| = n\}. \quad (2)$$

A *fully* zero-knowledge lookup argument for a commitment scheme CS is a CP-SNARK for the CP-relation $\hat{\mathcal{R}}_{\mathsf{zklookup}}$ and for the commitment scheme CS. We use the adjective fully zero-knowledge to distinguish our definition from the definition from previous work. State-of-the-art lookup arguments have prover time complexity independent of the length of the table and quasi-linear (or even linear) on the length of the sub-vector. To obtain such a property, all the lookup arguments for arbitrary tables in previous work precompute the table $\mathbf{T}$ producing auxiliary material that is then used during the proving phase. Thus, using the notational framework of Universal SNARK, the precomputation is handled by the Derive algorithm (since $\mathbf{t}$ is in the index).

Definition 5. *A tuple of algorithm* $\mathsf{CP} = (\mathsf{KGen}, \mathsf{Derive}, \mathsf{Prove}, \mathsf{Verify})$ *is a lookup argument for a commitment scheme* CS *if (1)* CP *forms a CP-SNARK for* $\hat{\mathcal{R}}_{\mathsf{lookup}}$ *and* CS*, (2)* Derive *is a* $\mathbb{F}$*-linear function (with respect to the proving key in its output) and the commitment scheme is linearly homomorphic and (3)* Prove *has running time* $\mathsf{poly}(n, \lambda)$*.*

We define an additional algorithm $\mathsf{Preproc}$ to handle our stronger privacy requirement. Similarly to Derive, the algorithm $\mathsf{Preproc}$ just performs an offline preprocessing — both algorithms are necessary *only* for speeding up the prov-

ing and verification algorithms. The difference is that Derive works over *public* information, meanwhile $\mathsf{Preproc}$ works over *private* information[12].

Definition 6. *A tuple of algorithm* $\mathsf{CP} = (\mathsf{KGen}, \mathsf{Derive}, \mathsf{Preproc}, \mathsf{Prove}, \mathsf{Verify})$ *is a fully zero-knowledge lookup argument for a matrix commitment scheme* CS *if (1)* $(\mathsf{KGen}, \mathsf{Derive}, \mathsf{Prove}', \mathsf{Verify})$ *forms a CP-SNARK for* $\hat{\mathcal{R}}_{\mathsf{zklookup}}$ *and* CS *where* Prove' *is the algorithm that upon witness* $(\mathbf{T}, \mathbf{F}, \rho_T, \rho_M)$ *such that* $\mathbf{T}_{|K} = \mathbf{F}$ *first runs* $(\mathsf{aux}_j)_{j\in[N]} \leftarrow \mathsf{Preproc}(\mathsf{srs}, \mathbf{T}, \rho_T)$ *and then runs* Prove *with witness* $(\mathbf{F}, \rho_M, (\mathsf{aux}_j)_{j\in K})$*; (2)* $\mathsf{Preproc}$ *is a* $\mathbb{F}$*-linear function and the commitment scheme is linearly homomorphic and (3)* Prove *has running time* $\mathsf{poly}(nd, \lambda)$*.*

4 Our New Zero-Knowledge Lookup Arguments

In this section, we present our new lookup arguments for KZG-based vector commitments. We let the commitment $\mathsf{c}_{\mathbf{t}}$ and $\mathsf{c}_{\mathbf{f}}$, to the vectors $\mathbf{t}$ and $\mathbf{f}$ respectively, be KZG commitments to randomized low-degree encodings of $\mathbf{t}$ and $\mathbf{f}$. We denote these polynomials $\mathsf{T}(X)$ and $\mathsf{F}(X)$, respectively. Since $\mathbf{t}$ and $\mathbf{f}$ have different sizes, we interpolate them over two multiplicative subgroups of $\mathbb{F}$: $\mathbb{K}$ of order N and $\mathbb{H}$ of order $n \leq N$. In our construction, we need $n \mid N$; however, this usually holds in practice where both n and N are powers of two. Hence, we have

$$\mathsf{T}(X) := \sum_{j=1}^{N} \mathbf{t}_j \lambda_j^{\mathbb{K}}(X) + \rho_{\mathsf{T}} \cdot \nu_{\mathbb{K}}(X), \quad \mathsf{F}(X) := \sum_{i=1}^{n} \mathbf{f}_i \lambda_i^{\mathbb{H}}(X) + \rho_{\mathsf{F}}(X) \cdot \nu_{\mathbb{H}}(X)$$

Above, $\rho_{\mathsf{F}}(X)$ is a random polynomial of degree $< \mathsf{b}_F$ so that $\mathsf{c}_{\mathbf{f}} = [\mathsf{F}(s)]_1$ is perfectly hiding. Furthermore, our lookup arguments work (and are zero-knowledge) for any choice of $\mathsf{b}_F \geq 0$; this property matters whenever the commitment $\mathsf{c}_{\mathbf{f}}$ is generated by other protocols with their own zero-knowledge requirements (e.g., $\mathsf{c}_{\mathbf{f}}$ may come from a SNARK construction where b_F is carefully set to meet the number of leaked evaluations of $\mathsf{F}(X)$ in that protocol). In other words, our lookup arguments achieve zero-knowledge without leaking additional evaluations of $\mathsf{F}(X)$.

On the other hand, if $\rho_{\mathsf{T}} \leftarrow\!\!\$\ \mathbb{F}$ is a random field element, then $\mathsf{c}_{\mathbf{t}} = [\mathsf{T}(s)]_1$ is a perfectly hiding commitment to $\mathbf{t}$. Otherwise, if $\rho_{\mathsf{T}} = 0$, we capture the case of public tables (that is the common use case of lookup arguments).

We use the following lemma from [22].

Lemma 1 (Set inclusion, [22]). *Let* $\mathbb{F}$ *be a field of characteristic* $p > N$*, and suppose that* $(a_i)_{i=1}^N$*,* $(b_i)_{i=1}^N$ *are arbitrary sequences of field elements. Then*

[12] Alternatively, one could define one single algorithm Derive that handles both public and private data, but in this case, one would need to re-define the Universal SNARK's framework to handle zero-knowledge correctly. Our definition instead is only functional as we require that $\mathsf{Preproc}$ and Prove form a two-step prover algorithm for a Universal SNARK.

$\{a_i\} \subseteq \{b_i\}$ as sets (with multiples of values removed), if and only if there exists a sequence $(m_i)_{i=1}$ of field elements from $\mathbb{F}_p \subseteq \mathbb{F}$ such that

$$\sum_{i=1}^{N} \frac{1}{X-a_i} = \sum_{i=1}^{N} \frac{m_i}{X-b_i} \tag{3}$$

in the function field $\mathbb{F}(X)$. Moreover, we have equality of the sets $\{a_i\} = \{b_i\}$, if and only if $m_i \neq 0$, for every $i = 1, \ldots, N$.

Roadmap. For the sake of presentation, we first describe our main lookup argument cq^+, which works for a public table $\mathbf{t}$, thus meeting Definition 5. This protocol is fully described in Fig. 1 and explained in the next section. Next, we discuss an optimized variant, cq^{++}. Finally, in Section 4.2 we show how to obtain the protocol meeting the fully zero-knowledge notion of Definition 6.

4.1 cq^+ Lookup Argument

For ease of exposition, we present our protocol as a public coin interactive argument; as usual, we can compile it into a CP-SNARK using the Fiat-Shamir heuristic.

Setup. We assume a universal $\mathsf{srs} = ((\left[s^j\right]_1)_{j=0}^{N_1}, (\left[s^j\right]_2)_{j=0}^{N_2}))$ for any $N_1 \geq N + \max(\mathsf{b}_F, 1) - 1$ and $N_2 \geq N + \max(\mathsf{b}_F, 1) + 1$, where b_F is the degree of the randomization polynomial $\rho_{\mathsf{F}}(X)$ explained earlier.

Round 1. Our interactive lookup protocol cq^+ starts the same as cq [13]. Namely, based on Lemma 1, the prover computes the multiplicities vector $\mathbf{m}$ such that $\sum_{j=1}^{N} \frac{m_j}{\mathbf{t}_j+X} = \sum_{i=1}^{n} \frac{1}{\mathbf{f}_i+X}$, and sends to the verifier a commitment $[m(s)]_1$ to a randomized low-degree encoding $m(X)$ of $\mathbf{m}$ over $\mathbb{K}$.

Round 2. The verifier sends a random challenge β. At this point, the goal of the prover is to convince the verifier that

$$\sum_{j=1}^{N} \frac{m_j}{\mathbf{t}_j+\beta} = \sum_{i=1}^{n} \frac{1}{\mathbf{f}_i+\beta} \tag{4}$$

which, by Schwartz-Zippel, implies the polynomial identity over $\mathbb{F}[X]$ and thus $\mathbf{f} \prec \mathbf{t}$ by Lemma 1. To this end, the prover commits to randomized low-degree encodings of the two vectors containing the terms of the two sums, i.e.,

$$A(X), B(X) \text{ s.t. } A_j = A(\omega_N^{j-1}) = \frac{m_j}{\mathbf{t}_j+\beta} \quad \text{and} \quad B_i = B(\omega_n^{i-1}) = \frac{1}{\mathbf{f}_i+\beta} \ . \tag{5}$$

In order to prove the well-formedness of $A(X)$ and $B(X)$, as in cq, the prover commits to the polynomials $Q_A(X) = (A(X)(\mathsf{T}(X)+\beta) - m(X))/\nu_{\mathbb{K}}(X)$ and $Q_B(X) = (B(X)(\mathsf{F}(X)+\beta) - 1)/\nu_{\mathbb{H}}(X)$. As we discuss later, we compute a commitment to $Q_A(X)$ using the cached quotients technique of [13] to meet the efficiency requirement (3) of Definition 5.

From this point, our protocol diverges from cq. At this point of the protocol, cq would proceed by applying Aurora's univariate sumcheck on both

$A(X)$ and $B(X)$ to prove the correctness of results $A(0) = \sum_j A(\omega_N^{j-1})$ and $B(0) = \sum_i B(\omega_n^{i-1})$ and then the verifier would check that the results are equal.

In cq^+, we instead apply Aurora's univariate sumcheck on a scaled sum of $A(X)$ and $B(X)$ and prove that the result is zero. More precisely, we define $C(X) := A(X) - \vartheta^{-1}B(X)\mathsf{z}(X)$ where we denote $\vartheta := N/n$ and $\mathsf{z}(X) := \nu_{\mathbb{K}\setminus\mathbb{H}}(X)$ and use the following lemma (see Appendix C for its proof).

Lemma 2. $\sum_{j=0}^{N} A(\omega_N^{j-1}) = \sum_{i=0}^{n} B(\omega_n^{i-1})$ *iff* $\sum_{j=0}^{N} C(\omega_N^{j-1}) = 0$.

The lemma above relies on the observation that the polynomial $\Delta(X) := \vartheta^{-1}B(X)\mathsf{z}(X)$ encodes over $\mathbb{K}$ the same vector encoded by $B(X)$ over $\mathbb{H}$, i.e., $\left(\frac{1}{\mathsf{f}_i+\beta}\right)_i$, but in different positions; while in the rest of positions it encodes zeros. Thus, $\sum_{j=0}^{N} \Delta(\omega_N^{j-1}) = \sum_{i=0}^{n} B(\omega_n^{i-1})$. Moreover, multiplying $B(X)$ by $\mathsf{z}(X)$ gives us for free a low-degree test on $B(X)$.

Thus, towards proving (4), we use Aurora's sumcheck on $C(X)$ to show

$$\exists R_C(X) \in \mathbb{F}_{\leq N-2}[X],\ Q_C(X) \text{ s.t. } C(X) = R_C(X)X + Q_C(X)\nu_{\mathbb{K}}(X)\ . \quad (6)$$

However, we do not send commitments to these two polynomials but use alternative techniques that allow us to obtain both zero knowledge and an efficient degree check on $R_C(X)$. More precisely, to obtain zero-knowledge, we use the sparse ZK sumcheck technique from Lunar [7]: the prover commits to a polynomial $S(X) := R_S X + \rho_S \nu_{\mathbb{K}}(X)$, with the idea that in the next round we perform a sumcheck on $C(X) + \eta^2 S(X)$, for a random challenge η to be chosen by the verifier in the following round. Actually, although for ease of expositions we introduced the use of $S(X)$ here; this polynomial is computed and committed as $[S(s)]_1$ in round 1. In summary, in round 2, the prover sends $[A(s), B(s), Q_B(s)]_1$.

Round 3. The verifier sends random challenges γ, η. In this round, the prover's goal is to show that

$$A(X)(\mathsf{T}(X) + \beta) - m(X) = Q_A(X)\nu_{\mathbb{K}}(X), \quad (7)$$
$$B(X)(\mathsf{F}(X) + \beta) - 1 = Q_B(X)\nu_{\mathbb{H}}(X), \quad (8)$$
$$(A(X) - \vartheta^{-1}B(X)\mathsf{z}(X) + \eta^2 S(X)) - (R_C(X) + \eta^2 R_S)X = Q_C(X)\nu_{\mathbb{K}}(X) \quad (9)$$

To prove equation (7), we use the cached quotient technique of [13] to compute a commitment $[Q_A(s)]_1$ using n scalar group multiplications (see below).

To prove equation (8), notice that we already sent $Q_B(X)$; thus, using a linearization trick and random point evaluations, we set $B_\gamma = B(\gamma)$ and we show $B(X)$ evaluates to B_γ on γ, $D(X) := B_\gamma(\mathsf{F}(X) + \beta) - 1 - Q_B(X)\nu_{\mathbb{H}}(\gamma)$ evaluates at 0 on γ. We batch these claims using the verifier's challenge η. Namely, we send the KZG-evaluation proof $P(X) := ((B(X) - B_\gamma) + \eta D(X))/(X - \gamma)$.

To prove equation (9), we apply a novel idea that allows obtaining, for free, a degree check on $R_C(X)$. We set the polynomial $U(X) = (X^\mu - 1)$ where $\mu = N_1 - N + 2$ and ask the prover to send $R_C^*(X) = R_C(X)U(X)$. To balance this, we multiply the rest of equation (9) by $U(X)$, obtaining

$$(A(X) - \vartheta^{-1}B(X)\mathsf{z}(X) + \eta^2 S(X))U(X) - R_C^*(X)X = Q_C(X)\nu_{\mathbb{K}}(X)U(X) \quad (10)$$

To further optimize this, we batch equations (7) and (10) by using the verifier's random challenge η (and multiplying (7) by $U(X)$), finally obtaining:

$$A(X) \cdot \mathsf{T}(X)U(X) + ((\beta+\eta)A(X) - m(X)) \cdot U(X) \\ - \frac{\eta}{\vartheta} B(X) \cdot \mathsf{z}(X)U(X) - Q(X)\nu_{\mathbb{K}}(X)U(X) = \eta R_C^*(X) \cdot X \quad (11)$$

The idea of this batching is that after multiplying (7) by $U(X)$, both equations aim to prove that the left-hand side is divisible by $\nu_{\mathbb{K}}(X)$ and thus we can send a single quotient polynomial $Q(X) = Q_A(X) + \eta Q_C(X) + \eta^2 \rho_S$.

To summarize, in round 3, the prover sends $[P(s), R_C^*(s), Q(s)]_1$ and B_γ.

Verification. The verifier proceeds as described in Verify of Fig. 1. The verification Item (ii) is a standard technique to check the batched evaluation proof $[P(s)]_1$. The verification Item (i) instead implements the check of Eq. (11) using pairings. Doing this requires the verifier to have in the verification key the $\mathbb{G}_2$ elements $[\mathsf{T}(s)U(s)]_2$ as well as $[U(s), \mathsf{z}(s)U(s), \nu_{\mathbb{K}}(s)U(s)]_2$. Therefore, we let Derive compute all these elements and include them in the verification key.

Prover efficiency. We discuss how the prover algorithm can be implemented with $O(n)$ scalar multiplications in $\mathbb{G}_1$ and $O(n \log n)$ $\mathbb{F}$ operations. First, one can easily see that by preprocessing the computation of the elements $\left[\lambda_j^{\mathbb{K}}(s)\right]_1$ and $[\nu_{\mathbb{K}}(s)]_1$ and by using the n-sparsity of $\mathbf{m}$, it is possible to compute $[m(s), A(s)]_1$ using $2(n+1)$ scalar multiplications. Computing $Q_B(X)$ is the only step that requires time $O(n \log n)$ (in field operations). Computing $[B(s), Q_B(s), P(s)]_1$ requires $\approx 3n$ scalar multiplications.

Computing the commitments $[R_C^*(s)]_1$ and $[Q_A(s)]_1$ with $\approx 2n$ and n scalar multiplications, respectively, can be achieved thanks to the cached quotients and, again, the sparseness of $\mathbf{m}$. Following [13], in Derive for $\mathbf{t}$, we compute and store

$$[Q_j(s)]_1 \text{ where } Q_j(X) := \frac{(\mathsf{T}(X) - \mathbf{t}_j)\lambda_j^{\mathbb{K}}(X)}{\nu_{\mathbb{K}}(X)} .$$

Then, we use this auxiliary input to compute, with $n+1$ scalar multiplications,

$$[Q_A(s)]_1 \leftarrow \textstyle\sum_{m_j \neq 0} A_j \, [Q_j(s)]_1 + [\rho_A(T(s)+\beta) - \rho_m]_1 . \quad (12)$$

The correctness of $Q_A(s)$ is due to

$$\begin{aligned} \textstyle\sum_{j=1}^N A_j Q_j(X) &= \textstyle\sum_{j=1}^N \frac{A_j(\mathsf{T}(X) - \mathbf{t}_j)\lambda_j^{\mathbb{K}}(X)}{\nu_{\mathbb{K}}(X)} \\ &= \textstyle\sum_{j=1}^N \frac{A_j(\mathsf{T}(X)+\beta)\lambda_j^{\mathbb{K}}(X)}{\nu_{\mathbb{K}}(X)} - \sum_{j=1}^N \frac{A_j(\mathbf{t}_j+\beta)\lambda_j^{\mathbb{K}}(X)}{\nu_{\mathbb{K}}(X)} \\ &\overset{(5)}{=} (\mathsf{T}(X)+\beta) \textstyle\sum_{j=1}^N \frac{A_j \lambda_j^{\mathbb{K}}(X)}{\nu_{\mathbb{K}}(X)} - \sum_{j=1}^N \frac{m_j \lambda_j^{\mathbb{K}}(X)}{\nu_{\mathbb{K}}(X)} \\ &= \tfrac{(A(X) - \rho_A \nu_{\mathbb{K}}(X))(\mathsf{T}(X)+\beta) - m(X) + \rho_m \nu_{\mathbb{K}}(X)}{\nu_{\mathbb{K}}(X)} \\ &= Q_A(X) - \rho_A(T(X)+\beta) + \rho_m . \end{aligned}$$

Using a similar technique, in Derive we can precompute $\left[(r_j^{\mathbb{K}}(s))_{j=1}^{N}, (r_i^{\mathbb{H}}(s))_{i=1}^{n}\right]_1$ where $\left\{r_j^{\mathbb{K}}(X) = \frac{\lambda_j^{\mathbb{K}}(X) - \lambda_j^{\mathbb{K}}(0)}{X} U(X)\right\}_{j\in[N]}$, and $\left\{r_i^{\mathbb{H}}(X) = \frac{\lambda_i^{\mathbb{H}}(X)\mathsf{z}(X) - \lambda_i^{\mathbb{H}}(0)}{X} U(X)\right\}_{i\in[n]}$, and use them to compute $[R_C^*(s)]_1$ in $2n$ scalar multiplications.

Thus, the prover's computation is dominated by $8n$ scalar multiplications, which was also the case in cq that did not achieve zero-knowledge and assumed $A(X)$ to be of degree $< N$.

cq^{++}: a variant with a shorter proof. We can further optimize cq^{+} by applying one more batching technique that consists of sending a single group element $[P^*(s)]_1 = [P(s) + R_C^*(s)]_1$ and in merging the two verification equations ((i)) and ((ii)) as follows:

$$\begin{aligned} &e\left([A(s)]_1, [\mathsf{T}(s)U(s)(s-\gamma)]_2\right) \cdot e\left(\left[(\beta+\eta)A(s) - m(s) + \eta^2 S(s)\right]_1, [U(s)(s-\gamma)]_2\right) \cdot \\ &\quad e\left(\tfrac{\eta}{\vartheta}[B(s)]_1, [\mathsf{z}(s)U(s)(s-\gamma)]_2\right)^{-1} \cdot e\left([Q(s)]_1, [\nu_{\mathbb{K}}(s)U(s)(s-\gamma)]_2\right)^{-1} \cdot \\ &\quad e\left(\eta[B(s) + \eta D(s) - B_\gamma]_1, [s]_2\right) = e\left(\eta[P^*(s)]_1, [s(s-\gamma)]_2\right) \ . \end{aligned}$$

This change also requires some small changes. First, we require in the srs to have $N_2 \geq N + \max(\mathsf{b}_F, 1) + 2$. Second, the verification key $\mathsf{vk}_{N,n}$ computed by Derive must include $\left[(s^k U(s), s^k \mathsf{z}(s)U(s), s^k \nu_{\mathbb{K}}(s)U(s))_{k=0}^{1}\right]_2$. Third, the table-dependent verification key for $\mathbf{t}$ should include $\left[(s^k \mathsf{T}(s)U(s))_{k=0}^{1}\right]_2$.

Overall efficiency. Assume that we use a standard curve like BLS12-381, where elements of $\mathbb{G}_1$ (resp., $\mathbb{F}$) are $\mathfrak{g}_1 = 384$ (resp., $\mathfrak{f} = 256$) bits long. Then, in cq^{+}, the communication is $8\mathfrak{g}_1 + 1\mathfrak{f}$ (3328 bits) and in cq^{++}, $7\mathfrak{g}_1 + 1\mathfrak{f}$ (2944 bits). The prover executes $\approx 8n$ scalar multiplications. Verifier has to execute 5 pairings in cq^{+} or 6 in cq^{++}. Importantly, two or three of the pairings are with the standard $\mathbb{G}_2$ element (depending on the variant, $[1, x]_2$ or $\left[1, x, x^2\right]_2$). Hence they can be batched with other pairings in the master protocol and essentially come for free.

If one does not wish ZK, we can remove $[S(s)]_1$ from the argument, and proof size is $7\mathfrak{g}_1 + 1\mathfrak{f}$ (2944 bits) in cq^{+}, and $6\mathfrak{g}_1 + 1\mathfrak{f}$ (2560 bits) in cq^{++}.

To compare, in cq [13] (that does not have ZK), the communication is $8\mathfrak{g}_1 + 3\mathfrak{f}$ (3840 bits), the prover's computation is $\approx 8n$ scalar multiplications, and the verifier has to execute 5 pairings. Hence, even cq^{+} (*with* ZK) has better communication than cq (*without* ZK) while having the same cost in the rest of the parameters.

Security. In the following theorem, we argue the security of cq^{+} (see Appendix C for the proof and the definition of the Power Discrete Logarithm (PDL) assumption); the proof of cq^{++} is very similar.

Theorem 1. *The protocol cq^{+} from Fig. 1 is a lookup argument according to Definition 5. Specifically, cq^{+} is knowledge-sound in the AGM and ROM under the (N_1, N_2)-PDL assumption (see Definition 12), and, furthermore, the protocol is zero-knowledge.*

$\underline{\mathsf{Derive}(\mathsf{srs}, \mathbf{t}, n)}$: // Assume that $|\mathbf{t}| = N$ and $n \mid N$, $\mathsf{srs} = (\left[(s^j)_{j\in[N_1]}\right]_1, \left[(s^j)_{j\in[N_2]}\right]_2)$ for any $N_1, N_2 \geq N + \max(\mathsf{b}_F, 1) - 1$.

Set $\mu = N_1 - N + 2$; define $U(X) := (X^\mu - 1)$, $\vartheta = N/n$, and $\mathsf{z}(X) = \nu_{\mathbb{K}\setminus\mathbb{H}}(X)$;

Define $\mathsf{T}(X) := \sum_{j=1}^{N} = \mathbf{t}_j \lambda_j^{\mathbb{K}}(X)$;

Let $\left\{r_j^{\mathbb{K}}(X) = \frac{\lambda_j^{\mathbb{K}}(X) - \lambda_j^{\mathbb{K}}(0)}{X} U(X)\right\}_{j\in[N]}$, $\left\{r_i^{\mathbb{H}}(X) = \frac{\lambda_i^{\mathbb{H}}(X)\mathsf{z}(X) - \lambda_i^{\mathbb{H}}(0)}{X} U(X)\right\}_{i\in[n]}$,

and $\left\{Q_j(X) = \frac{(\mathsf{T}(X) - \mathbf{t}_j)\lambda_j^{\mathbb{K}}(X)}{\nu_{\mathbb{K}}(X)}\right\}_{j\in[N]}$.

Compute $\mathsf{ek}_{\mathbf{t},n} := \left[(r_j^{\mathbb{K}}(s))_{j=1}^N, (r_i^{\mathbb{H}}(s))_{i=1}^n, U(s), \nu_{\mathbb{K}}(s), s\nu_{\mathbb{K}}(s), (Q_j(s))_{j=1}^N, \mathsf{T}(s)\right]_1$;

Compute $\mathsf{vk}_{\mathbf{t},n} := [1, U(s), \mathsf{z}(s)U(s), \nu_{\mathbb{K}}(s)U(s), \mathsf{T}(s)U(s)]_2$;

Return $(\mathsf{ek}_{\mathbf{t},n}, \mathsf{vk}_{\mathbf{t},n})$.

$\underline{\mathsf{Prove}(\mathsf{ek}_{N,n}, \mathsf{c}_\mathbf{f}, (\mathbf{f}, \rho_\mathsf{F}(X)))}$: //$\mathsf{c}_\mathbf{f} = \left[\sum_i \mathbf{f}_i \lambda_i^{\mathbb{H}}(s) + \rho_\mathsf{F}(s)\nu_{\mathbb{H}}(s)\right]_1$, $deg(\rho_\mathsf{F}) = \mathsf{b}_F$.

Compute $\mathbf{m} = (m_1, \ldots, m_N)$ s.t. $\forall j : \mathbf{t}_j$ appears m_j times in $\mathbf{f}$; samples $\rho_m \leftarrow\$ \mathbb{F}$;

Compute $[m(s)]_1 \leftarrow \sum_{j=1}^N m_j \cdot \left[\lambda_j^{\mathbb{K}}(s)\right]_1 + \rho_m \cdot [\nu_{\mathbb{K}}(s)]_1$; // n scalar mults

Sample $R_S, \rho_S \leftarrow\$ \mathbb{F}$ and compute $[S(s)]_1 \leftarrow R_S \cdot s + \rho_S \cdot \nu_{\mathbb{K}}(s)$;

$\beta \leftarrow \mathsf{RO}(\mathsf{vk}_{N,n} \| (\mathsf{c}_\mathbf{t}, \mathsf{c}_\mathbf{f}) \| [m(s)]_1)$ //Fiat-Shamir challenge.

Sample $\rho_A \leftarrow\$ \mathbb{F}$, $\rho_B(X) \leftarrow\$ \mathbb{F}_{\leq 1}[X]$;

Let $A_j \leftarrow m_j/(\mathbf{t}_j + \beta)\ \forall j = 1, \ldots, N$ and $B_i \leftarrow 1/(\mathbf{f}_i + \beta)\ \forall i = 1, \ldots, n$;

Compute $[A(s)]_1 \leftarrow \sum_{j=1}^N A_j \left[\lambda_j^{\mathbb{K}}(s)\right]_1 + \rho_A \cdot [\nu_{\mathbb{K}}(s)]_1$;

Compute $[B(s)]_1 \leftarrow \sum_{i=1}^n B_i \left[\lambda_i^{\mathbb{H}}(s)\right]_1 + \rho_B(s) \cdot [\nu_{\mathbb{H}}(s)]_1$;

Compute $Q_B(X) \leftarrow (B(X)(\mathsf{F}(X) + \beta) - 1)/\nu_{\mathbb{H}}(X)$ and $[Q_B(s)]_1$;

$(\gamma, \eta) \leftarrow \mathsf{RO}(\beta \| [A(s), B(s), Q_B(s), S(s)]_1)$;//Fiat-Shamir challenge.

Compute $B_\gamma \leftarrow B(\gamma)$, $D(X) \leftarrow B_\gamma \cdot (\mathsf{F}(X) + \beta) - 1 - Q_B(X)\nu_{\mathbb{H}}(\gamma)$;

Compute $P(X) \leftarrow ((B(X) - B(\gamma)) + \eta D(X))/(X - \gamma)$ and $[P(s)]_1$;

Compute $[R_C^*(s)]_1 \leftarrow \sum_{m_j \neq 0} A_j \cdot \left[r_j^{\mathbb{K}}(s)\right]_1 - \vartheta^{-1} \sum_{i=1}^n B_i \cdot \left[r_i^{\mathbb{H}}(s)\right]_1 + \eta R_S \cdot [U(s)]_1$

Compute $[Q_A(s)]_1 \leftarrow \sum_{m_j \neq 0} A_j \cdot [Q_j(s)]_1 + [\rho_A(T(s) + \beta) - \rho_m]_1$;

Compute $[Q_C(s)]_1 \leftarrow \left[\rho_A + \vartheta^{-1}\rho_B(s)\right]_1$;

Compute $[Q(s)]_1 \leftarrow [Q_A(s)]_1 + \eta\,[Q_C(s)]_1 - \eta^2\,[\rho_S]_1$;

Return $\pi = ([m(s), S(s), A(s), B(s), Q_B(s), P(s), R_C^*(s), Q(s)]_1, B_\gamma)$.

$\underline{\mathsf{Verify}(\mathsf{vk}_{\mathbf{t},n}, \mathsf{c}_\mathbf{f}, \pi)}$:

Compute $[D(s)]_1 \leftarrow B_\gamma(\mathsf{c}_\mathbf{f} + [\beta]_1) - [1]_1 - \nu_{\mathbb{H}}(\gamma)\,[Q_B(s)]_1$.

Return 1 if and only if the following holds:

(i) $e([A(s)]_1, \mathsf{c}_\mathbf{t}) \cdot e((\beta + \eta) \cdot [A(s)]_1 - [m(s)]_1 + \eta^2\,[S(s)]_1, [U(s)]_2) \cdot e(\eta/\vartheta \cdot [B(s)]_1, [\mathsf{z}(s)U(s)]_2)^{-1} \cdot e([Q(s)]_1, [\nu_{\mathbb{K}}(s)U(s)]_2)^{-1} = e(\eta \cdot [R_C^*(s)]_1, [x]_2)$,

(ii) $e([B(s)]_1 + \eta\,[D(s)]_1 - [B_\gamma]_1, [1]_2) = e([P(s)]_1, [s - \gamma]_2)$

Fig. 1. Our zero-knowledge lookup argument cq^+.

4.2 Our fully zero-knowledge lookup argument

In this setting we have $\mathsf{T}(X) = \sum_{j=1}^N = \mathbf{t}_j \lambda_j^{\mathbb{K}}(X) + \rho_\mathsf{T} \cdot \nu_{\mathbb{K}}(X)$ where $\rho_\mathsf{T} \leftarrow\$ \mathbb{F}$ and $\mathsf{c}_\mathbf{t} = \left[\mathsf{T}(s)\right]_1$. We need only slight modifications to turn cq^+ to a fully zero-knowledge lookup argument. We refer to the modified lookup argument as zkcq^+, formally described in Fig. 6 in Appendix C.2. First, we defer, from Derive to Preproc, the computation of all the *table-dependent* group elements. Namely, $\mathsf{Preproc}(\mathsf{srs}, \mathbf{t}, \rho_\mathsf{T})$ computes $([Q_j(s))]_1)_{j=1}^N$ and $\tilde{\mathsf{c}}_\mathbf{t} \leftarrow [\mathsf{T}(s)U(s)]_2$. The latter group element is included as part of the proof at proving time by the algorithm Prove. As consequence, Verify needs to additionally run the pairing check $e([1]_1, \tilde{\mathsf{c}}_\mathbf{t}) = e(\mathsf{c}_\mathbf{t}, [U(s)]_2)$ to verify the well-formedness of the commitment

$\mathsf{c_t}$. In the proof of knowledge soundness, this check allows us to ensure that the polynomials extracted from $\mathsf{c_t}$ and $\tilde{\mathsf{c}}_\mathbf{t}$ are of the form $\mathsf{T}^*(X)$ and $\mathsf{T}^*(X)U(X)$ for some $\mathsf{T}^*(X)$; thus, after verifying this we can apply virtually the same proof of Theorem 1.

5 Our Matrix Lookup Argument

We show a compiler from a fully zero-knowledge vector lookup argument for KZG-based vector commitment to a fully zero-knowledge matrix lookup for the (succinct) KZG-based matrix commitment from Section 2.3. The same construction applies for lookup argument as in Definition 5.

5.1 The Straw Man Solution

An alternative approach to commit to a matrix is to one-by-one vector commit to its columns. The obvious shortcoming is that the commitment scheme is not succinct in the number of columns. Nonetheless, this approach already results in a matrix lookup argument (under the assumption that the vector commitment is linearly homomorphic). In particular, consider the lookup argument that hashes together the columns $\mathbf{t}_j$ of the table $\mathbf{T}$ and the columns $\mathbf{f}_j$ of the sub-matrix $\mathbf{F}$ using a random challenge ρ computing vectors

$$\mathbf{t}^* = \sum_j \mathbf{t}_j \rho^{j-1} \qquad\qquad \mathbf{f}^* = \sum_j \mathbf{f}_j \rho^{j-1}.$$

Notice that by Schwartz-Zippel lemma we that $\mathbf{f}^* \prec \mathbf{t}^*$ implies $\mathbf{F} \prec \mathbf{T}$ with overwhelming probability. Thus, we could run a vector lookup argument over $(\mathbf{f}^*, \mathbf{t}^*)$, thanks to the linear homomorphic property of the commitment scheme the verifier can compute commitments to $\mathbf{f}^*$ and $\mathbf{t}^*$ and verify the proof. Notice the prover time complexity is $\mathsf{poly}(n, d, \lambda)$ thanks to the $\mathbb{F}$-linearity of the pre-computation algorithm. However, the verification time is linear in in the number of columns. We show in the next section how to restore succinct verification time and commitment size.

5.2 Our scheme

In Fig. 2 we describe our scheme $\mathsf{mtx}[\mathsf{CP}]$ that runs internally a lookup argument CP for KZG-based vector commitment scheme. The proof of the following theorem is in Appendix D. In the description of the scheme, we let $\mathbb{K}$ (resp. $\mathbb{H}$) be a multiplicative subgroup of $\mathbb{F}$ of order $N \cdot d$ (resp. of order $n \cdot d$), we let $\omega := \omega_{n \cdot d}$ be the fixed generator for $\mathbb{H}$ and we consider the following matrices and polynomial:

1. the matrix $\mathbf{R} \in \mathbb{F}^{N \times d}$ where $R_{i,j} = i$,
2. for any k the matrix $\mathbf{C}^{(k)} \in \mathbb{F}^{k \times d}$ where $C_{i,j} = j$.
3. Let $\nu_{\bar{\mathbb{H}}}(X)$ be the vanishing polynomial of $\bar{\mathbb{H}} = \{\omega^{d \cdot i + j} : j \in [1, d-1], i \in [n]\}$.

$\underline{\mathsf{Derive}(\mathsf{srs}, N, d, n)}$:

Let $\bar{\mathbf{t}}, \bar{\mathbf{f}}, \bar{\mathbf{r}}_N, \bar{\mathbf{c}}_N$ and $\bar{\mathbf{c}}_n$ be vectorizations of the matrices $\mathbf{T}, \mathbf{F}, \mathbf{R}, \mathbf{C}^{(N)}$ and $\mathbf{C}^{(n)}$.
Compute $\mathsf{c}_{r,N} \leftarrow \mathsf{Com}(\mathsf{ck}, \bar{\mathbf{r}}_N)$, $\mathsf{c}_{c,N} \leftarrow \mathsf{Com}(\mathsf{ck}, \bar{\mathbf{c}}_N)$ and $\mathsf{c}_{c,n} \leftarrow \mathsf{Com}(\bar{\mathbf{c}}_n)$.
Compute $(\mathsf{ek}', \mathsf{vk}') \leftarrow \mathsf{CP.Derive}(\mathsf{srs}, Nd, nd)$.
Return $(\mathsf{ek}', \mathsf{vk}_n)$ where $\mathsf{vk}_n = (\mathsf{c}_{r,N}, \mathsf{c}_{c,N}, \mathsf{c}_{c,n}, [\nu_{\bar{\mathbb{H}}}(s)]_2, \mathsf{vk}')$.

$\underline{\mathsf{Preproc}(\mathsf{srs}, \mathbf{T})}$:

Compute $(\mathsf{aux}_{T,j})_{j\in[Nd]} \leftarrow \mathsf{CP.Preproc}(\mathsf{srs}, \bar{\mathbf{t}})$,
$(\mathsf{aux}_{R,j})_{j\in[Nd]} \leftarrow \mathsf{CP.Preproc}(\mathsf{srs}, \bar{\mathbf{r}}_N)$,
$(\mathsf{aux}_{C,j})_{j\in[Nd]} \leftarrow \mathsf{CP.Preproc}(\mathsf{srs}, \bar{\mathbf{c}}_N)$.
Let $\mathsf{aux}_i = (\mathsf{aux}_{T,di+j}, \mathsf{aux}_{R,di+j}, \mathsf{aux}_{C,di+j})_{j\in[d]}$.
Return $(\mathsf{aux}_i)_{i\in[N]}$.

$\underline{\mathsf{Prove}(\mathsf{ek}, (\mathsf{c}_\mathbf{T}, \mathsf{c}_\mathbf{F}), \mathbf{F}, (\mathsf{aux}_j)_{j\in K})}$: //$\mathbf{T}_{|K} = \mathbf{F}$, $K = \{k_1, \ldots, k_n\}$.

Let $\mathbf{S}$ be s.t. $\mathbf{S}_{i,j} = k_i$ for $i \in [n], j \in [d]$.
Let $\sigma(X)$ be the randomized low-degree encoding over $\mathbb{H} = \langle\omega\rangle$ of the vectorization of $\mathbf{S}$.
Compute $w(X)$ such that $\sigma(\omega \cdot X) - \sigma(X) = w(X) \cdot \nu_{\bar{\mathbb{H}}}(X)$.
$(\rho, \zeta) \leftarrow \mathsf{RO}(\mathsf{vk}_n \| (\mathsf{c}_\mathbf{T}, \mathsf{c}_\mathbf{F}) \| (\mathsf{c}_{R,n}, \mathsf{c}_{R',n}, \mathsf{c}_w))$. //Fiat-Shamir challenge.
Compute $z \leftarrow \sigma(\omega \cdot \zeta)$.
Compute proofs π_R and $\pi_{R'}$ for $\hat{\mathcal{R}}_{\mathsf{eval}}(\omega \cdot \zeta, z; \sigma(X)) = 1$ and $\hat{\mathcal{R}}_{\mathsf{eval}}(\zeta, z; \sigma(\omega \cdot X)) = 1$;
Let π^* proof for $\hat{\mathcal{R}}_{\mathsf{zklookup}}((N \cdot d, n \cdot d); \varepsilon; (\bar{\mathbf{t}}^*, \bar{\mathbf{f}}^*)) = 1$ where

$$\bar{\mathbf{t}}^* = \bar{\mathbf{t}} + \rho \cdot \bar{\mathbf{c}}_N + \rho^2 \cdot \bar{\mathbf{r}}_N \qquad \bar{\mathbf{f}}^* = \bar{\mathbf{f}} + \rho \cdot \bar{\mathbf{c}}_n + \rho^2 \cdot \bar{\boldsymbol{\sigma}} \tag{13}$$

Return $\pi = ([\sigma(s)]_1, [\sigma(\omega \cdot s)]_1, [w(s)]_1, \pi_R, \pi_{R'}, \pi^*, z)$.

$\underline{\mathsf{Verify}(\mathsf{vk}_n, (\mathsf{c}_\mathbf{T}, \mathsf{c}_\mathbf{F}), \pi)}$:

Parse the proof $\pi = (\mathsf{c}_{R,n}, \mathsf{c}_{R',n}, \mathsf{c}_w, \pi_R, \pi_{R'}, \pi^*, z)$.
$(\rho, \zeta) \leftarrow \mathsf{RO}(\mathsf{vk}_n \| (\mathsf{c}_\mathbf{T}, \mathsf{c}_\mathbf{F}) \| (\mathsf{c}_{R,n}, \mathsf{c}_{R',n}, \mathsf{c}_w))$. //Fiat-Shamir challenge.
Compute $\mathsf{c}^*_\mathbf{T} \leftarrow \mathsf{c}_\mathbf{T} + \rho \mathsf{c}_{c,N} + \rho^2 \mathsf{c}_{r,N}$ and $\mathsf{c}^*_\mathbf{F} \leftarrow \mathsf{c}_\mathbf{F} + \rho \mathsf{c}_{c,n} + \rho^2 \mathsf{c}_{R,n}$.
Return 1 if the following checks hold (else 0):

(i) $\mathsf{Verify}_{\mathsf{eval}}(\mathsf{ck}, (\mathsf{c}_{R,n}, \omega \cdot \zeta, z)) = 1$,
(ii) $\mathsf{Verify}_{\mathsf{eval}}(\mathsf{ck}, (\mathsf{c}_{R',n}, \zeta, z)) = 1$,
(iii) $e(\mathsf{c}_{R',n} - \mathsf{c}_{R,n}, [1]_2) = e(\mathsf{c}_w, [\nu_{\bar{\mathsf{k}}}(s)]_2)$,
(iv) $\mathsf{CP.Verify}(\mathsf{srs}, \mathsf{vk}', (\mathsf{c}^*_\mathbf{T}, \mathsf{c}^*_\mathbf{F}), \pi^*) = 1$.

Fig. 2. Our Matrix Lookup Argument $\mathsf{mtx}[\mathsf{CP}]$.

Theorem 2. *The lookup argument* $\mathsf{mtx}[\mathsf{CP}]$ *defined in Fig. 2 is knowledge-sound in the AGM and ROM under the* $(N \cdot d, N \cdot d)$*-PDL assumption and assuming that* CP *is knowledge-sound. Furthermore, the protocol is zero-knowledge assuming* CP *is zero-knowledge.*

A row-column Matrix Lookup Argument. In Appendix D.1 we consider the rows-columns sub-matrix relation where $\mathbf{F} \prec \mathbf{T}$ if and only if there exist (multi)sets $R = \{r_1, \ldots, r_n\}$ and $C = \{c_1, \ldots, c_d\}$ with $\mathbf{F}_{i,j} = \mathbf{T}_{r_i,c_j}$, and give an *rows-columns* matrix-lookup argument system $\mathsf{mtx}^*[\mathsf{CP}]$ for such a relation. Briefly, the main difference with the scheme in this section is that we commit to an additional vector $\bar{\boldsymbol{\sigma}}^C$ which is the concatenation of the vector $(c_1, \ldots, c_d)$ for n times, prove in zero-knowledge its tensor structure, and show that $\bar{\mathbf{f}}^* = \bar{\mathbf{f}} + \rho \cdot \bar{\boldsymbol{\sigma}}^C + \rho^2 \cdot \bar{\boldsymbol{\sigma}}$ is a sub-vector of $\bar{\mathbf{t}}^*$.

Table 1. Summary of efficiency of our constructions for matrix lookups. The relation being considered is parametrized as follows: a table matrix of size $N \times d$; the submatrix being looked up is of size $n \times d$. $\mathbb{P}$ is the cost of one pairing. Proof size includes commitment to the witness.

Scheme	Preprocessing	Proof size	Time (P)	Time (V)
$\mathsf{mtx}^{\mathsf{longprf}}[\mathsf{zkcq}^+]$ (Section 5.1)	$O(dN \log N)\mathbb{F}, \mathbb{G}$	$(d+9)\mathfrak{g}_1 + 1\mathfrak{f}$	$O(nd)\mathbb{G}_1 + O(nd \log n)\mathbb{F}$	$d\mathbb{G}_1 + 7\mathbb{P}$
$\mathsf{mtx}[\mathsf{zkcq}^+]$ (Fig. 2)	$O(dN \log dN)\mathbb{F}, \mathbb{G}$	$16\mathfrak{g}_1 + 2\mathfrak{f}$	$O(nd)\mathbb{G}_1 + O(nd \log(nd))\mathbb{F}$	$13\mathbb{P}$
[?]	$O(dN \log dN)\mathbb{F}, \mathbb{G}$	$20\mathfrak{g}_1 + 6\mathfrak{f}$	$O(nd \log nd)\mathbb{G}_1 + O(nd \log(nd))\mathbb{F}$	$23\mathbb{P}$

5.3 Concrete Efficiency

In Table 1 we describe the complexity of proving a matrix lookup in a table T described by a matrix of size $N \times d$. The size of the submatrix we are looking up in the larger table is $n \times d$.

In Appendix F we describe a breakdown of efficiency measurements for our fully zero-knowledge construction ($\mathsf{mtx}[\mathsf{zkcq}^+]$). Our naive scheme, the one derived from the observations in Section 5.1, and the scheme in Appendix D.1 have efficiency analyses which follow similarly. The values for [12] are taken directly from the paper, the number of pairing in verification by a simple inspection of the protocol, the extra $\log nd\mathbb{G}_1$ factor in the prover time complexity arises from their sub-protocol adapted from [36].

6 Zero-Knowledge Decision Tree Statistics

A decision tree is an algorithm that, upon an input, performs a finite sequence of adaptive queries on the input and eventually outputs a value. Concretely, we consider binary decision trees where the inputs are vectors in $[B]^d$ for natural numbers d and B, where the queries are comparisons and the outputs (often called the labels) are in $[B]$. We let N_{tot} be the number of nodes in a decision tree T, and we index the root node with 1. A binary tree with N_{tot} nodes and where each node has either zero children or exactly two children, has $N_{\mathsf{leaf}} := (N_{\mathsf{tot}} + 1)/2$ leaf nodes, and the remaining $N_{\mathsf{int}} = N_{\mathsf{tot}} - N_{\mathsf{leaf}}$ nodes are called internal nodes (including the root node). We index the internal nodes of the decision tree with numbers in $[N_{\mathsf{int}}]$. The computation of a decision tree T upon input $\mathbf{x}$, denoted as $\mathsf{T}(\mathbf{x})$, consists of a traversal of the tree from the root node to a leaf. During the traversal, the computation fetches, from each internal node i, a threshold t_i and a feature index $e_i \in [d]$. If $x_{e_i} < t_i$, the computation continues recurring on the left child of node i, and otherwise, to the right child. Once reaches a leaf, the computation outputs the label v_i assigned to the leaf i as the final output.

Therefore, seen as a data structure, a decision tree T is made by a binary tree (namely, the *structure* of the tree), by the values d_i, t_i for each internal node i, and by the label v_i for each leaf node i. We refer to this encoding of a tree as the *standard encoding*. We define $\mathcal{T}_{N_{\mathsf{tot}},B,d}$ to be the set of decision trees with N_{tot} nodes that maps vector in $[B]^d$ to the co-domain $\mathbb{F}$.

Quasi-Complete Decision Tree. We define the notion of *quasi-complete* decision tree. The difference with a standard tree is that during the traversal, the computation fetches from each internal node i two vectors $\mathbf{E}_i$ and $\mathbf{T}_i$, we call the vector $\mathbf{E}_i \in \{0,1\}^d$ the feature vector associated to the node i and vector $\mathbf{T}_i \in [B]^d$ the threshold vector associated to the node i. The computation continues recurring on the left child of node i if $\forall j \in [d] : \mathbf{E}_{i,j} = 1 \Rightarrow x_j < \mathbf{T}_{i,j}$, on to the right child of the node i if $\forall j \in [d] : \mathbf{E}_{i,j} = 1 \Rightarrow x_j \geq \mathbf{T}_{i,j}$, or outputs $\perp$ if neither of the two conditions holds. The pseudo-code of the evaluation of a quasi-complete decision tree is in Fig. 8 in Appendix E.

Similarly to decision trees, we define $\mathcal{T}^*_{N_{\mathsf{tot}},B,d}$ to be the set of quasi-complete decision trees with N_{tot} nodes that maps feature vector in $[B]^d$ to the co-domain $\mathbb{F}$. Notice that when for any node j the (row) vector $\mathbf{E}_j$ is an elementary vector (namely with only one position set to 1) then the quasi-complete decision tree is indeed a standard decision tree thus $\mathcal{T}_{N_{\mathsf{tot}},B,d} \subset \mathcal{T}^*_{N_{\mathsf{tot}},B,d}$.

The class of quasi-complete decision trees defines a correct but not complete computational model. In fact, every input is either correctly labelled to one label or to the error message $\perp$. Being a more general class of computation than standard decision trees, it is easier to decide whether a data structure is a quasi-complete decision tree than to decide if it is a standard decision tree. This allows for faster prover time. On the other hand, an adversary that commits to (strictly) quasi-complete decision tree (namely, a decision tree in $\mathcal{T}^*_{N_{\mathsf{tot}},B,d} \setminus \mathcal{T}_{N_{\mathsf{tot}},B,d}$) cannot prove contradicting statements, in particular, we require that it cannot prove any statistics on an input $\mathbf{x}$ whenever $\mathsf{T}(\mathbf{x}) = \perp$.

6.1 Security Model

We consider the scenario where a *model producer* commits to a decision tree T, the model producer can delegate the computation of statistics on a set of data points and predictions over T to a server, a user can obtain such statistics. Informally, we require integrity of the computation, namely the statistics are correctly computed over the set of data points and predictions over the committed decision tree T, and privacy, namely the user does not learn anything more than the validity of such statistics.

We consider an adversarial model where either the model producer and the server can be corrupted, or the user is corrupted. Previous work considered only the case where the model producer is honest [38] (and either the server or user are corrupted). Notice that a corrupted model producer could commit to a *useless/bogus* decision tree. Unfortunately, we cannot do anything to prevent that. On the other hand, we would like to prevent the corrupted model producer and corrupted server can convince the user of the validity of *incoherent* statistics. For example, an attacker should not be able to convince the user that simultaneously $\mathsf{T}(\mathbf{x}) = 1$ and $\mathsf{T}(\mathbf{x}) = 0$ for a data point $\mathbf{x}$.

To formalize such property, we use the notion of knowledge soundness for argument systems. In particular, we require that whenever the verifier is convinced (w.r.t. a commitment c) of the statistic over a set of data points, there

must exist an extractor that outputs an opening of the commitment to a decision tree T where such a statistic over such data is correct. Notice the commitment to the decision tree is binding. Thus we must obtain coherent statistics over many queries on the same committed decision tree. To optimize the efficiency of the statistic evaluations, we split in two parts the generation of a valid commitment from the evaluation of a proof for a given tuple statistic/data points.

Definition 7. *Let $\mathcal{S}$ be an arbitrary set of tuples (S, m) such that $S : [B]^m \to \{0,1\}^*$ and $m \in \mathbb{N}$ where S is an efficiently computable function (a statistic). A (commit-and-prove)* decision-tree-statistic argument *for a set of statistics $\mathcal{S}$ is a tuple $\mathsf{zkDT} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{VerCom}, \mathsf{Derive}, \mathsf{Prove}, \mathsf{Verify})$ where:*

(i) $\mathsf{CS}_{DT} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{VerCom})$ *define an extractable commitment scheme for the domain $\mathcal{T}^*$ of (quasi-complete) decision tree. In particular,* KGen *takes in input a natural number N_{tot} the maximum number of nodes, and the natural numbers B and d, besides the security parameter and generates a commitment key for the set $\mathcal{T}^*_{B,d,N_{\mathsf{tot}}}$.*

(ii) $\mathsf{CP}_{DT} = (\mathsf{Derive}, \mathsf{Prove}, \mathsf{Verify})$ *define a Universal CP-SNARK for the indexed CP-relation $\hat{\mathcal{R}}_{\mathsf{DTstat}}$ defined below.*

$$\hat{\mathcal{R}}_{\mathsf{DTstat}} = \left\{ \mathsf{pp}; (S,m); y, (\mathbf{x}_j)_{j\in[m]}); \mathsf{T} : \begin{array}{c} y = S(\mathsf{T}(\mathbf{x}_1), \ldots, \mathsf{T}(\mathbf{x}_m)), \\ \forall i : \mathsf{T}(\mathbf{x}_i) \neq \perp, \quad (S,m) \in \mathcal{S} \end{array} \right\}.$$

6.2 The Extended Encoding of Decision Trees

We introduce an alternative encoding of a decision tree as a data structure before presenting our zero-knowledge decision-tree statistics argument. We follow the work of Chen *et al.* [10]. In particular, we define a d-dimensional *box* as a tuple of vectors in $[B+1]^d$, where the first vector defines the *left bounds* and the second vector defines the *right bounds*. We say that a vector $\mathbf{x} \in [B]^d$ *is contained* in a box $(\mathbf{b}^{\leftarrow}, \mathbf{b}^{\rightarrow})$ if $\mathbf{b}^{\leftarrow} \leq \mathbf{x} < \mathbf{b}^{\rightarrow}$. We can assign to each node of a decision tree a $d-$dimensional box. In particular, we denote with $(\mathbf{N}^{\leftarrow}_i, \mathbf{N}^{\rightarrow}_i)$ the box assigned to the i-th node in the tree and with $\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}$ the tuple of matrices of all the boxes of a decision tree (mapping the i-th row to the box of i-th node).

We can associate a (quasi-complete) decision tree to a tuple of matrices, below we define such a relation:

Definition 8. *Given a quasi-complete decision tree T with N_{tot} nodes and given matrices $\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}$, we say that $(\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow})$ is a* boxes-encoding *of T if*

1. *$\mathbf{N}^{\leftarrow}_1 = \mathbf{0}$ and $\mathbf{N}^{\rightarrow}_1 = \mathbf{B} + \mathbf{1}$, where $\mathbf{0}$ (resp. $\mathbf{1}$ and $\mathbf{B}$) is the vector of all 0 (resp. of all 1 and of all B).*
2. *Let $p \in [N_{\mathsf{int}}]$ be the index of a node and let l and r respectively be the indexes of the left child and right child of the node with index p.*

$$\mathbf{N}^{\leftarrow}_l - \mathbf{N}^{\leftarrow}_p = \mathbf{0} \qquad \mathbf{N}^{\rightarrow}_r - \mathbf{N}^{\rightarrow}_p = \mathbf{0} \tag{14}$$

$$\mathbf{E}_p \circ (\mathbf{N}^{\rightarrow}_l - \mathbf{T}_p) = \mathbf{0} \qquad \mathbf{E}_p \circ (\mathbf{N}^{\leftarrow}_r - \mathbf{T}_p) = \mathbf{0} \tag{15}$$

$$(\mathbf{1} - \mathbf{E}_p) \circ (\mathbf{N}^{\rightarrow}_l - \mathbf{N}^{\rightarrow}_p) = \mathbf{0} \qquad (1 - \mathbf{E}_p) \circ (\mathbf{N}^{\leftarrow}_r - \mathbf{N}^{\leftarrow}_p) = \mathbf{0} \tag{16}$$

The computation, through a boxes-encoding, of a decision tree $\mathsf{T}(\mathbf{x})$ consists in finding the index k of the leaf whose box contains $\mathbf{x}$ and outputs the label associated with such a leaf. For a quasi-complete decision tree, such an index k might not exist. We formalize this in the next definition and prove such a computational equivalence in the next lemma whose proof is in Appendix E.

Definition 9. *Let T be a quasi-complete decision tree with N_{tot} nodes (with domain $[B]^d$) and $(\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow})$ be a boxes-encoding of T. For any $\mathbf{x} \in [B]^d$, if $\mathbf{x}$ is contained in the box of a leaf of T define the index of the leaf as $k_{\mathsf{T}}(\mathbf{x})$ such that $\mathbf{x}$ is contained in $(\mathbf{N}^{\leftarrow}_{k_{\mathsf{T}}(\mathbf{x})}, \mathbf{N}^{\rightarrow}_{k_{\mathsf{T}}(\mathbf{x})})$ else $k_{\mathsf{T}}(\mathbf{x})$ is set to $\perp$.*

Whenever it is clear from the context, we will omit the subscript T and write $k(\mathbf{x})$ to refer to such an index.

Lemma 3. *Let T be a quasi-complete decision tree with N_{tot} nodes and $(\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow})$ be a boxes-encoding of T. Let $\mathbf{v}$ be the vector of the labels assigned to the leaf nodes of T, namely for any $i \in [N_{\mathsf{int}} + 1, N_{\mathsf{tot}}]$, we have v_i as the label assigned to the i-th leaf. For any $\mathbf{x} \in [B]^d$, $\mathsf{T}(\mathbf{x}) = v_{k(\mathbf{x})}$ or $\mathsf{T}(\mathbf{x}) = \perp$.*

As corollary of the above lemma, we have that the boxes of leaf do not overlap because no vector $\mathbf{x}$ can be contained in more than one of the boxes of the leaves.

Before giving the next definition, we set some notation: given a decision tree, we say that node p splits at coordinate $i^* \in [d]$ if i^* is a coordinate where p's left and right child boundaries are different, namely, $\mathbf{N}^{\rightarrow}_{p,i} \neq \mathbf{N}^{\rightarrow}_{\ell,i}$ and $\mathbf{N}^{\leftarrow}_{p,i} \neq \mathbf{N}^{\leftarrow}_{r,i}$ where ℓ and r are the left and right child of p. We are ready to describe our (more redundant but ZKP-friendly) encoding of a quasi-complete decision tree.

Definition 10. *Let T be a quasi-complete decision tree with N_{tot} nodes. Let $\mathcal{T} = (\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}, \mathbf{v}, \mathbf{L}, \mathbf{R}, \mathbf{E})$ be a tuple of matrices (described below). We say that $\mathcal{T}$ is an extended encoding of T if the following conditions hold:*

(i) $(\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow})$ *is a boxes-encoding of* T*;*
(ii) $\mathbf{v}$ *is the vector of the labels assigned to the leaf nodes of* T*;*
(iii) $\mathbf{L}$ *(resp.* $\mathbf{R}$*) is the* $N_{\mathsf{int}} \times N_{\mathsf{tot}}$ *bit matrix whose* p*-th row is the elementary vector* $\mathbf{e}^{\mathsf{T}}_{\ell}$ *(resp.* $\mathbf{e}^{\mathsf{T}}_{r}$*) if* ℓ *is the left (resp.* r *is the right) child of node* p*'s in* T*,*
(iv) $\mathbf{E} \in \{0,1\}^{N_{\mathsf{int}} \times d}$ *is the bit matrix such that its* p*-th row and* i *column is* 1 *iff the node* p *splits at coordinate* i*.*

Let Encode *be the algorithm that, given a quasi-complete decision tree* T*, computes the extended encoding of* T*.*

Let the matrices $\mathbf{P}^{\leftarrow}, \mathbf{P}^{\rightarrow} \in \mathbb{F}^{N_{\mathsf{int}} \times d}$ describe the boxing encodings of the internal nodes, and $\mathbf{F}^{\leftarrow}, \mathbf{F}^{\rightarrow} \in \mathbb{F}^{N_{\mathsf{leaf}} \times d}$ describe the boxing encodings of the leaves. Thus:

$$\mathbf{N}^{\leftarrow} = \begin{pmatrix} \mathbf{P}^{\leftarrow} \\ = \\ \mathbf{F}^{\leftarrow} \end{pmatrix} \text{ and } \mathbf{N}^{\rightarrow} = \begin{pmatrix} \mathbf{P}^{\rightarrow} \\ = \\ \mathbf{F}^{\rightarrow} \end{pmatrix}.$$

The function Encode in Definition 10 is injective but not surjective. In the next lemma (whose proof is in Appendix E), we give sufficient conditions for belonging in the image of Encode.

Lemma 4. *Consider a tuple* $(\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}, \mathbf{L}, \mathbf{R}, \mathbf{E}, \mathbf{v})$ *such that the following constraints hold:*

a) The following equations hold:

$$\mathbf{N}^{\leftarrow}_1 = \mathbf{0}, \mathbf{N}^{\rightarrow}_1 = \mathbf{B} + \mathbf{1}, \tag{17}$$

$$\mathbf{L} \cdot \mathbf{N}^{\leftarrow} = \mathbf{P}^{\leftarrow}, \mathbf{R} \cdot \mathbf{N}^{\rightarrow} = \mathbf{P}^{\rightarrow}, \tag{18}$$

$$\mathbf{E} \circ (\mathbf{L} \cdot \mathbf{N}^{\rightarrow} - \mathbf{R} \cdot \mathbf{N}^{\leftarrow}) = \mathbf{0} \tag{19}$$

$$(\mathbf{1} - \mathbf{E}) \circ (\mathbf{P}^{\leftarrow} - \mathbf{R} \cdot \mathbf{N}^{\leftarrow}) = \mathbf{0}, \qquad (\mathbf{1} - \mathbf{E}) \circ (\mathbf{P}^{\rightarrow} - \mathbf{L} \cdot \mathbf{N}^{\rightarrow}) = \mathbf{0} \tag{20}$$

b) All the boxes are not empty. Namely, for all i, j *we have* $\mathbf{N}^{\leftarrow}_{i,j} < \mathbf{N}^{\rightarrow}_{i,j}$.

c) The matrix $\begin{pmatrix} \mathbf{L} \\ \hline\hline \mathbf{R} \end{pmatrix}$ *is a (row) permutation of the (squared) matrix* $(\mathbf{0} \| \mathbf{I}_{N_{\mathsf{tot}}-1})$ *(the matrix whose rows are the row vectors* $(\mathbf{e}_i)_{i \in [2, N_{\mathsf{tot}}]}$ *of length* N_{tot}*).*

Then there exists a quasi-complete decision tree T *with* N_{tot} *nodes such that* $\mathsf{Encode}(\mathsf{T}) = (\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}, \mathbf{L}, \mathbf{R}, \mathbf{E}, \mathbf{v})$.

6.3 Extractable Commitment to Decision Trees

In a nutshell our commitment procedure on input a decision tree computes the encoding described in Section 6.2, then it commits to the matrices $\mathbf{F}^{\leftarrow}, \mathbf{F}^{\rightarrow}$ and $\mathbf{v}$ and prove in zero-knowledge the constraints from Lemma 4. We can implement the latter zero-knowledge proof using a general-purpose R1CS circuit describing the constraints of the lemma, however, the size of the circuit would be $O(dN^2_{\mathsf{tot}})$, in fact, we would need to commit to the remaining matrices $\mathbf{P}^{\leftarrow}, \mathbf{P}^{\rightarrow}, \mathbf{L}, \mathbf{R}$ and $\mathbf{E}$ and we would need already $O(dN^2_{\mathsf{tot}})$ multiplication gates for Eq. (18). We show how to remove the quadratic dependency from the number of total nodes. The main idea is to notice that $\mathbf{L}$ and $\mathbf{R}$ have sparsity linear in N_{tot}, thus we can use techniques from [37] to commit to such sparse matrices and then prove in zero-knowledge that the constraints in Item c) of Lemma 4 hold for the committed matrices. The remaining constraints can be proved in $O(dN_{\mathsf{tot}} \log(dN_{\mathsf{tot}}))$.

The building blocks. Consider the following (indexed) CP-relations:

$$\hat{\mathcal{R}}_{\mathsf{lin}} = \{\mathsf{pp}; \varepsilon; (\mathbf{M}, \mathbf{N}, \mathbf{R}) : \mathbf{M} \cdot \mathbf{N} = \mathbf{R}\} \tag{21}$$

$$\hat{\mathcal{R}}_{\mathsf{had}} = \{\mathsf{pp}; \varepsilon; (\mathbf{M}, \mathbf{N}) : \mathbf{M} \circ \mathbf{N} = \mathbf{0}\} \tag{22}$$

$$\hat{\mathcal{R}}_{\mathsf{perm}} = \left\{\mathsf{pp}; (N, i(X)); \varepsilon; p(X) : \exists \pi, \forall j \in [N] : i(\pi(\omega^j)) = p(\omega^j)\right\} \tag{23}$$

$$\hat{\mathcal{R}}_{\mathsf{shift}} = \left\{\mathsf{pp}; S; \varepsilon; (\mathbf{v}, \mathbf{u}) : \mathbf{v}_i = \mathbf{u}_{(i+S \pmod{|\mathbf{u}|})}\right\} \tag{24}$$

$$\hat{\mathcal{R}}_{\mathsf{range}} = \left\{\mathsf{pp}; (B, n, d); \varepsilon; \mathbf{X} : \mathbf{X} \in [B]^{n \times d}\right\} \tag{25}$$

$$\hat{\mathcal{R}}_{\mathsf{sm}} = \left\{\mathsf{pp} : K; \varepsilon; \mathbf{M} : \mathbf{M}_{|K} = \mathbf{0}\right\} \tag{26}$$

Our scheme uses CP-SNARKs for all the relations above as building blocks. The first three relations are standard, and CP-SNARKs for them can be found in the related work. Given a CP-SNARK for $\hat{\mathcal{R}}_{\mathsf{lin}}$, we can define a CP-SNARK for

$\hat{\mathcal{R}}_{\mathsf{shift}}$ in fact that the shifting operator can be described through a linear transformation. The latter linear transformation can be public, thus the underlying CP-SNARK (for $\hat{\mathcal{R}}_{\mathsf{lin}}$) does not need to be zero-knowledge w.r.t. the first matrix $\mathbf{M}$, in particular, a commitment to such a matrix could be part of the index polynomials. A CP-SNARK for $\hat{\mathcal{R}}_{\mathsf{range}}$ can be realized using our lookup argument and considering the table $\mathbf{b} = (j)_{j\in[B]}$ and proving that the vectorization $\bar{\mathbf{x}}$ of $\mathbf{X}$ is such that $\bar{\mathbf{x}} \prec \mathbf{b}$. Finally, a CP-SNARK for $\hat{\mathcal{R}}_{\mathsf{sm}}$ can be easily realized by committing to a matrix $\bar{\mathbf{T}}$ such that $\bar{\mathbf{T}}_K = \mathbf{T}$ and 0 everywhere else and to the vanishing polynomial in ν_K in $\mathbb{G}_2$ as part of the index. At proving time, the prover returns as proof a commitment to the quotient polynomial q such that $f'(X) = q(X) \cdot \nu_K(X)$ where $f'(X)$ is the polynomial associated to the matrix $\mathbf{M} - \bar{T}$. At verification time the verifier checks $e(\mathsf{c}_{\mathbf{M}} - \mathsf{c}_{\bar{\mathbf{T}}}, [1]_2) = e(\pi, [\nu_K]_2)$.

For the CP-SNARK for $\hat{\mathcal{R}}_{\mathsf{lin}}$, we require two different commitment schemes, one for the first matrix and one for the other two. In particular, we consider an alternative way to commit to matrices following the work of [31,37]. Let $\mathbf{M}$ be a *basic matrix*, namely a matrix whose rows are elementary vectors. Let $\mathbb{H}$ be any fixed subgroup with $|\mathbb{H}| \geq N_{\mathsf{tot}}$ [13] of $\mathbb{F}$ with generator ω. For any basic matrix $\mathbf{M} \in \{0,1\}^{n\times k}$ and $n, k \in \mathbb{N}$, let $col_{\mathbf{M}}(X)$ be the (low-degree) polynomial such that $col_{\mathbf{M}}(\omega^i) = \omega^j$ where the i-th row of $\mathbf{M}$ is the vector $\mathbf{e}_j^\top$ (notice that $col_{\mathbf{M}}$ is the LDE of the vector whose i-th element is the value ω^j). We define the sparse (hiding) commitment of a matrix $\mathbf{M}$ as a (hiding) polynomial commitment of $col_{\mathbf{M}}$. Namely, we define:

$$\mathsf{sparseCom}(\mathsf{ck}, \mathbf{M}, \rho) := \mathsf{Com}(\mathsf{ck}, col_{\mathbf{M}}, \rho).$$

Notice that, by the above definition, a sparse commitment to a basic matrix $\mathbf{M}$ has a dual interpretation (as a sparse matrix or as a vector col).

Let $\mathsf{CP}_{\mathsf{lin}}$ be a CP-SNARK for the $\hat{\mathcal{R}}_{\mathsf{lin}}$ relation where the first matrix is committed using $\mathsf{sparseCom}$ while the other matrices are committed with the matrix commitment scheme from Section 2.3. An instantiation of such a scheme can be found for the matrix-times-vector case (namely, $\mathbf{N} \in \mathbb{F}^{n\times 1}$) in Baloo by [37] (see Sections 5.2, 5.3 and 5.4 of the paper). We show a generalization to matrix-times-matrix case in Appendix E.4. We write $\boxed{\mathbf{M}}$ to underline that the matrix $\mathbf{M}$ is committed with a sparse matrix commitment. For example, we can write $(\mathsf{pp}, \varepsilon; \boxed{\mathbf{M}}, \mathbf{N}, \mathbf{R}) \in \hat{\mathcal{R}}_{\mathsf{lin}}$ to identify the statement that there are commitments $\mathsf{c}_M, \mathsf{c}_N, \mathsf{c}_R$ where the first is a sparse matrix commitment and that open to $\mathbf{M}, \mathbf{N}$ and $\mathbf{R}$ with $\mathbf{M} \cdot \mathbf{N} = \mathbf{R}$.

Let $\mathsf{CP}_{\mathsf{had}}$ be a CP-SNARK for the $\hat{\mathcal{R}}_{\mathsf{had}}$ relation where all the matrices are committed using the commitment scheme from Section 2.3. Notice that a CP-SNARK for our matrix commitment scheme for such a CP-relation derives directly from CP-SNARK for vector commitment. Finally, let $\mathsf{CP}_{\mathsf{perm}}$ be a CP-SNARK for the CP-relation $\hat{\mathcal{R}}_{\mathsf{perm}}$. The permutation argument of Plonk [18] is a CP-SNARK for such a relation.

[13] Alternatively, we can consider the same subgroup used for the matrix commitment and thus $|\mathbb{H}| = N_{\mathsf{tot}} \cdot d$.

The Extractable Commitment to Decision Tree. We are ready to define our extractable commitment scheme for the domain of quasi-complete decision trees. The main idea is, as part of the proof of opening, to commit to the matrices $\mathbf{L}$ and $\mathbf{R}$ through sparse commitments to basic matrices and then prove the linear relations from Lemma 4 in zero-knowledge with a complexity that is linear in the sparsity of the matrices and the dimension d. The additional constraints on the two matrices $\mathbf{L}$ and $\mathbf{R}$ are proved using the permutation argument.

$\underline{\mathsf{KGen}(1^\lambda, (N_{\mathsf{tot}}, B, d))}$:

Sample a type-3 pairing group pp_G with security level λ.
Set $\mathsf{ck}' \leftarrow (\mathsf{pp}_G, ([s^i]_1)_{i\in[N_1]}, ([s^i]_2)_{i\in[N_2]}$ for random secrets $s \leftarrow\!\$\, \mathbb{Z}_q$.
Compute $\mathsf{srs}_{\mathsf{sm},1} \leftarrow \mathsf{CP}_{\mathsf{sm}}.\mathsf{Derive}(\mathsf{ck}', [N_{\mathsf{int}}])$, $\mathsf{srs}_{\mathsf{sm},2} \leftarrow \mathsf{CP}_{\mathsf{sm}}.\mathsf{Derive}(\mathsf{ck}', [N_{\mathsf{int}}, N_{\mathsf{tot}}])$ and $\mathsf{srs}_{\mathsf{sm},3} \leftarrow \mathsf{CP}_{\mathsf{sm}}.\mathsf{Derive}(\mathsf{ck}', \{1\})$.
Compute $\mathsf{srs}_{\mathsf{perm}} \leftarrow \mathsf{CP}_{\mathsf{perm}}.\mathsf{Derive}(\mathsf{ck}', (N_{\mathsf{tot}} - 1, id))$.
Compute $\mathsf{srs}_{\mathsf{range}} \leftarrow \mathsf{CP}_{\mathsf{range}}.\mathsf{Derive}(\mathsf{ck}', (B, N_{\mathsf{tot}}, d))$.
Compute $\mathsf{srs}_{\mathsf{shift}} \leftarrow \mathsf{CP}_{\mathsf{range}}.\mathsf{Derive}(\mathsf{ck}', N_{\mathsf{int}})$.
Return $\mathsf{ck} := (\mathsf{ck}', [b(s)]_1, \mathsf{srs}_{\mathsf{perm}}, \mathsf{srs}_{\mathsf{range}}, \mathsf{srs}_{\mathsf{shift}}, (\mathsf{srs}_{\mathsf{sm},j})_{j\in[3]})$.

$\underline{\mathsf{Com}(\mathsf{ck}, \mathsf{T}, \rho_{\mathsf{T}})}$:

Compute $(\mathbf{L}, \mathbf{R}, \mathbf{E}, \mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}, \mathbf{v}) \leftarrow \mathsf{Encode}(\mathsf{T})$, parses ρ_{T} as $(\rho_v, \rho_{\leftarrow}, \rho_{\rightarrow})$.
$\mathsf{c}_v \leftarrow \mathsf{Com}(\mathsf{ck}, \mathbf{v}, \rho_v)$.
$\mathsf{c}_{\leftarrow} \leftarrow \mathsf{Com}(\mathsf{ck}, \bar{\mathbf{F}}_{\leftarrow}, \rho_{\leftarrow})$, $\mathsf{c}_{\rightarrow} \leftarrow \mathsf{Com}(\mathsf{ck}, \bar{\mathbf{F}}_{\rightarrow}, \rho_{\rightarrow})$, $\mathsf{c}'_{\leftarrow} \leftarrow \mathsf{Com}(\mathsf{ck}, \bar{\mathbf{P}}_{\leftarrow}, \rho_{\leftarrow})$, $\mathsf{c}'_{\rightarrow} \leftarrow \mathsf{Com}(\mathsf{ck}, \bar{\mathbf{P}}_{\rightarrow}, \rho_{\rightarrow})$.
$\mathsf{c}_{ln} \leftarrow \mathsf{Com}(\mathsf{ck}, \bar{\mathbf{L}} \cdot \mathbf{N}^{\rightarrow},)$, $\mathsf{c}_{rn} \leftarrow \mathsf{Com}(\mathsf{ck}, \bar{\mathbf{R}} \cdot \mathbf{N}^{\leftarrow})$ and $\mathsf{c}_E \leftarrow \mathsf{Com}(\mathsf{ck},)$.
$\mathsf{c}_L \leftarrow \mathsf{sparseCom}(\mathsf{ck}, \bar{\mathbf{L}})$, $\mathsf{c}_R \leftarrow \mathsf{sparseCom}(\mathsf{ck}, \bar{\mathbf{R}})$ and $\mathsf{c}'_R \leftarrow \mathsf{sparseCom}(\underline{\mathbf{R}})$.
Let $col_{\bar{\mathbf{L}}}, col_{\bar{\mathbf{R}}}$ and $col_{\underline{\mathbf{R}}}$ be the underlying polynomials.
Prove the following statements, let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_{16})$ be the proofs.

$$\begin{aligned}
\pi_1, \ldots, \pi_4 &: && (\boxed{\bar{\mathbf{L}}}, \mathbf{N}^{\leftarrow}, \bar{\mathbf{P}}_{\leftarrow}),\ (\boxed{\bar{\mathbf{L}}}, \mathbf{N}^{\rightarrow}, \bar{\mathbf{L}} \cdot \mathbf{N}^{\rightarrow}),\ (\boxed{\bar{\mathbf{R}}}, \mathbf{N}^{\rightarrow}, \bar{\mathbf{P}}_{\rightarrow}),\ (\boxed{\bar{\mathbf{R}}}, \mathbf{N}^{\leftarrow}, \bar{\mathbf{R}} \cdot \mathbf{N}^{\leftarrow}) \in \hat{\mathcal{R}}_{\mathsf{lin}},\\
\pi_5, \pi_6, \pi_7 &: && (\bar{\mathbf{E}}, \bar{\mathbf{L}} \cdot \mathbf{N}^{\rightarrow} - \bar{\mathbf{R}} \cdot \mathbf{N}^{\leftarrow}),\ (\mathbf{1} - \bar{\mathbf{E}}, \mathbf{P}^{\leftarrow} - \bar{\mathbf{R}} \cdot \mathbf{N}^{\leftarrow}),\ (\mathbf{1} - \bar{\mathbf{E}}, \mathbf{P}^{\rightarrow} - \bar{\mathbf{L}} \cdot \mathbf{N}^{\rightarrow}) \in \hat{\mathcal{R}}_{\mathsf{had}},\\
\pi_8, \pi_9 &: && ((B, N_{\mathsf{tot}}, d); \mathbf{N}^{\rightarrow} - \mathbf{N}^{\leftarrow} - \mathbf{1}) \in \hat{\mathcal{R}}_{\mathsf{range}},\quad (N_{\mathsf{tot}} - 1, id; col_{\bar{\mathbf{L}}}(X) + col_{\underline{\mathbf{R}}}(X)) \in \hat{\mathcal{R}}_{\mathsf{perm}},\\
\pi_{10} &: && (N_{\mathsf{int}}, col_{\bar{\mathbf{R}}}, col_{\underline{\mathbf{R}}}) \in \hat{\mathcal{R}}_{\mathsf{shift}},\\
\pi_{11}, \ldots, \pi_{16} &: && ([N_{\mathsf{int}}]; \bar{\mathbf{F}}_{\leftarrow}), ([N_{\mathsf{int}}]; \bar{\mathbf{F}}_{\rightarrow}), ((N_{\mathsf{int}}, N_{\mathsf{tot}}]; \bar{\mathbf{P}}_{\leftarrow}), ((N_{\mathsf{int}}, N_{\mathsf{tot}}]; \bar{\mathbf{P}}_{\rightarrow}), (\{1\}; \bar{\mathbf{P}}_{\leftarrow}), (\{1\}; \bar{\mathbf{P}}_{\rightarrow} - \mathbf{B}) \in \hat{\mathcal{R}}_{\mathsf{sm}}.
\end{aligned}$$

Return $(\mathsf{c}_{\leftarrow}, \mathsf{c}_{\rightarrow}, \mathsf{c}_v), \pi$ where $\pi = (\mathsf{c}'_{\leftarrow}, \mathsf{c}'_{\rightarrow}, \mathsf{c}_{ln}, \mathsf{c}_{rn}, \mathsf{c}_E, \mathsf{c}_L, \mathsf{c}_R, \mathsf{c}'_R, \boldsymbol{\pi})$.

$\underline{\mathsf{Verify}(\mathsf{ck}, \mathsf{c}_{\mathsf{T}})}$:

Let $\mathsf{c}_{\mathsf{T}} = (\mathsf{c}_{\leftarrow}, \mathsf{c}_{\rightarrow}, \mathsf{c}_v, \pi)$ and parse π. Let $\mathsf{c}_{N,\leftarrow} \leftarrow \mathsf{c}_{\leftarrow} + \mathsf{c}'_{\leftarrow}$ and $\mathsf{c}_{N,\rightarrow} \leftarrow \mathsf{c}_{\rightarrow} + \mathsf{c}'_{\rightarrow}$.
1. Verify $\pi_1, \pi_2, \pi_3, \pi_4$ w.r.t. $(\mathsf{c}_L, \mathsf{c}_{N,\leftarrow}, \mathsf{c}'_{\leftarrow}), (\mathsf{c}_L, \mathsf{c}_{N,\rightarrow}, \mathsf{c}_{ln}), (\mathsf{c}_R, \mathsf{c}_{N,\rightarrow}, \mathsf{c}'_{\rightarrow}), (\mathsf{c}_R, \mathsf{c}_{N,\leftarrow}, \mathsf{c}_{rn})$.
2. Verify π_5, π_6, π_7 w.r.t. $(\mathsf{c}_E, \mathsf{c}_{ln} - \mathsf{c}_{rn}), ([1]_1 - \mathsf{c}_E, \mathsf{c}'_{\leftarrow} - \mathsf{c}_{rn}), ([1]_1 - \mathsf{c}_E, \mathsf{c}'_{\rightarrow} - \mathsf{c}_{ln})$.
3. Verify π_8, π_9 w.r.t. $((B, N_{\mathsf{tot}}, d); \mathsf{c}_{N,\rightarrow} - \mathsf{c}_{N,\leftarrow} - [1]_1)$ and $(N_{\mathsf{tot}} - 1, id; \mathsf{c}_L + \mathsf{c}'_R)$.
4. Verify π_{10} w.r.t. $(N_{\mathsf{int}}; (\mathsf{c}_R, \mathsf{c}'_R))$.
5. Verify $\pi_{11}, \ldots, \pi_{16}$ w.r.t. $([N_{\mathsf{int}}], \mathsf{c}_{\leftarrow}), ([N_{\mathsf{int}}], \mathsf{c}_{\rightarrow}), ((N_{\mathsf{int}}, N_{\mathsf{tot}}], \mathsf{c}'_{\leftarrow}), ((N_{\mathsf{int}}, N_{\mathsf{tot}}], \mathsf{c}'_{\rightarrow}), (\{1\}, \mathsf{c}'_{\leftarrow}), (\{1\}, \mathsf{c}'_{\rightarrow} - [b(s)]_1)$.

Fig. 3. Our extractable commitment CS_{DT}. The value $N_1 \geq N_{\mathsf{tot}} \cdot d$, N_1 and N_2 are sufficiently big to allow instantiation of the SRSs of building-block CP-SNARKs.

To help readability we list below some shortcuts used in the description of the protocol.

$$\bar{\mathbf{F}}_{\leftarrow} := \begin{pmatrix} \mathbf{0} \\ = \\ \mathbf{F}^{\leftarrow} \end{pmatrix},\ \bar{\mathbf{F}}_{\rightarrow} := \begin{pmatrix} \mathbf{0} \\ = \\ \mathbf{F}^{\rightarrow} \end{pmatrix},\ \bar{\mathbf{P}}_{\leftarrow} := \begin{pmatrix} \mathbf{P}^{\leftarrow} \\ = \\ \mathbf{0} \end{pmatrix},\ \bar{\mathbf{P}}_{\rightarrow} := \begin{pmatrix} \mathbf{P}^{\rightarrow} \\ = \\ \mathbf{0} \end{pmatrix},$$
$$\bar{\mathbf{L}} := \begin{pmatrix} \mathbf{L} \\ = \\ \mathbf{0} \end{pmatrix},\ \bar{\mathbf{R}} := \begin{pmatrix} \mathbf{R} \\ = \\ \mathbf{0} \end{pmatrix},\ \underline{\mathbf{R}} := \begin{pmatrix} \mathbf{0} \\ = \\ \mathbf{R} \end{pmatrix},\ \bar{\mathbf{E}} := \begin{pmatrix} \mathbf{E} \\ = \\ \mathbf{0} \end{pmatrix}$$

The set I corresponds to the coordinates of the first N_{int} rows of matrix commitment, the set I' corresponds to the coordinates of the first row. The padding for the matrices make them all to have N_{tot} rows. Moreover, we let $\mathbf{B}$ the matrix whose first row is the vector $(B+1,\ldots,B+1)$ and the remaining rows are set to $\mathbf{0}$, and we let $b(X)$ be the LDE of the vectorization of such a matrix. This polynomial can be computed in $O(d\log d)$ operations, however, for simplicity, we commit to the polynomial at key-generation phase. We let id be the low-degree polynomial that evaluates $id(\omega^i)=\omega^{i+1}$ for $i\in[N_{\mathsf{tot}}-1]$ (equivalently, the commitment $[id(s)]_1$ is a sparse-matrix commitment to the matrix $(\mathbf{0}\|\mathbf{I}_{N_{\mathsf{tot}}-1})$).

Theorem 3. *The commitment scheme CS_{DT} defined in Fig. 3 is hiding and it is an extractable commitment scheme for the domain $\{\mathcal{T}^*_{N_{\mathsf{tot}},B,d}\}_{N_{\mathsf{tot}},d,B}$ in the AGM and assuming the building blocks are knowledge-sound and zero-knowledge.*

Efficiency. The extractable commitment in this section has constant proof size when the CP-SNARK for $\hat{\mathcal{R}}_{\mathsf{lin}}$ is instantiated with the building block described in Appendix E.4. Its proving time is $O(dN_{\mathsf{tot}}\log(dN_{\mathsf{tot}}))$ when applied to a decision tree with d features and N_{tot} nodes. Notice that N_{tot} is usually at least one order of magnitude larger than d.

6.4 CP-SNARK for Statistics on Decision Trees

The building blocks are the following CP-SNARKs for the matrix commitment scheme in Section 2.3.

1. Let $\mathsf{CP}_{\mathsf{lookup}*}$ be a CP-SNARK for the indexed CP-relation:

$$\hat{\mathcal{R}}_{\mathsf{lookup}*}=\left\{\mathsf{pp};(N,d,n);\varepsilon;(\mathbf{T}_j)_{j\in[m]},(\mathbf{F}_j)_{j\in[m]}:\begin{array}{c}(\mathbf{F}_1\|\ldots\|\mathbf{F}_m)\prec(\mathbf{T}_1\|\ldots\|\mathbf{T}_m)\\ \forall j:|\mathbf{T}_j|=N\times d,|\mathbf{F}_j|=n\times d\end{array}\right\}$$

2. Let $\mathsf{CP}_{\mathsf{range}}$ be a CP-SNARK for the indexed CP-relation $\hat{\mathcal{R}}_{\mathsf{range}}$ in Eq. (25).
3. Let $\mathsf{CP}_{\mathsf{stat}}$ be a CP-SNARK for the following indexed CP-relation:

$$\hat{\mathcal{R}}_{\mathsf{stat}}=\{\mathsf{pp},(S,m);y;\mathbf{v}:S(\mathbf{v})=y\wedge|\mathbf{v}|=m\}$$

It is not hard to see that we can define a CP-SNARK for $\hat{\mathcal{R}}_{\mathsf{lookup}*}$ on top of our compiler from Section 5. Namely, we batch together the matrices $\mathbf{T}_j$ and the matrices $\mathbf{F}_j$ using a random challenge, as described in the *straw-man solution* in Section 5, and then we run our matrix lookup argument.

Putting together Theorems 3 and 4, we have a corollary that the above scheme CP_{DT} and the commitment scheme CS_{DT} from the previous section define a decision-tree statistic argument.

Theorem 4. *If the building blocks have the properties stated in Items 1 to 3 then $\mathsf{CP}_{DT}=(\mathsf{Derive},\mathsf{Prove},\mathsf{Verify})$ define an Universal CP-SNARK for the indexed CP-relation $\hat{\mathcal{R}}_{\mathsf{DTstat}}$.*

$\underline{\mathsf{Derive}(\mathsf{srs}, (S,m))}$**:**

Compute $\mathsf{srs}_{(S,m)} \leftarrow \mathsf{CP}_{\mathsf{stat}}.\mathsf{Derive}(\mathsf{ck}, (S,m))$, $\mathsf{srs}_m \leftarrow \mathsf{CP}_{\mathsf{lookup}*}.\mathsf{Derive}(\mathsf{ck}, m)$.
Compute $\mathsf{srs}_{(B,m,d)} \leftarrow \mathsf{CP}_{\mathsf{range}}.\mathsf{Derive}(\mathsf{ck}, (B,m,d))$ with values B, d contained in srs.
Return the specialized SRSs.

$\underline{\mathsf{Prove}(\mathsf{srs}, (\mathsf{c}_\mathsf{T}, y, (\mathbf{x}_j)_{j\in[m]}), (\mathsf{T}, \rho_\mathsf{T}))}$**:**

Parse $\mathsf{c}_\mathsf{T} = (\mathsf{c}_v, \mathsf{c}_{\overleftarrow{\cdot}}, \mathsf{c}_{\overrightarrow{\cdot}})$ and $\rho_\mathsf{T} = (\rho_v, \rho_{\overleftarrow{\cdot}}, \rho_{\overrightarrow{\cdot}})$.
Let $k_i = k_\mathsf{T}(\mathbf{x}_i)$ and $K = \{k_1, \ldots, k_m\}$, where $k_\mathsf{T}(\cdot)$ as defined in Definition 9.
Compute matrix commitments $\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3$ to the matrices $(\bar{\mathbf{F}}_{\overleftarrow{\cdot}})_{|K}, (\bar{\mathbf{F}}_{\overrightarrow{\cdot}})_{|K}, \mathbf{v}_{|K}$.
Compute a proof π_{zklookup} that

$$(m; \varepsilon; ((\bar{\mathbf{F}}_{\overleftarrow{\cdot}})_{|K}, (\bar{\mathbf{F}}_{\overrightarrow{\cdot}})_{|K}, \mathbf{v}_{|K}), (\bar{\mathbf{F}}_{\overleftarrow{\cdot}}, \bar{\mathbf{F}}_{\overrightarrow{\cdot}}, \mathbf{v})) \in \hat{\mathcal{R}}_{\mathsf{lookup}*}.$$

Compute a (not hiding) commitment to the matrix $\mathbf{X}$ whose rows are the vectors $(\mathbf{x}_j)_{j\in[m]}$.
Compute proofs $\pi^{\overleftarrow{\cdot}}_{\mathsf{range}}$ and $\pi^{\overrightarrow{\cdot}}_{\mathsf{range}}$ for the following two statements:

$$((B,m,d); \mathbf{X} - (\bar{\mathbf{F}}_{\overleftarrow{\cdot}})_{|K}) \in \hat{\mathcal{R}}_{\mathsf{range}}, \qquad ((B,m,d); (\bar{\mathbf{F}}_{\overrightarrow{\cdot}})_{|K} - \mathbf{X} - \mathbf{1}) \in \hat{\mathcal{R}}_{\mathsf{range}}.$$

Compute a proof π_{stat} that $((S,m); y; \mathbf{v}_{|K})) \in \hat{\mathcal{R}}_{\mathsf{stat}}$.
Return $(\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \pi_{\mathsf{zklookup}}, \pi^{\overleftarrow{\cdot}}_{\mathsf{range}}, \pi^{\overrightarrow{\cdot}}_{\mathsf{range}}, \pi_{\mathsf{stat}})$.

$\underline{\mathsf{Verify}(\mathsf{srs}, \mathsf{vk}_{(S,m)}, (\mathsf{c}_\mathsf{T}, y, (\mathbf{x}_j)_{j\in[m]}), \pi_\mathsf{T})}$**:**

Parse the proof $\pi_\mathsf{T} = (\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \pi_{\mathsf{zklookup}}, \pi^{\overleftarrow{\cdot}}_{\mathsf{range}}, \pi^{\overrightarrow{\cdot}}_{\mathsf{range}}, \pi_{\mathsf{stat}})$.
Compute $\mathsf{c}_\mathbf{X} \leftarrow \mathsf{Com}(\mathsf{ck}, \mathbf{X})$ where $\mathbf{X}$ is the matrix whose rows are the vectors $(\mathbf{x}_j)_{j\in[m]}$.
Return 1 if the following statements hold (else 0):

1. $\mathsf{CP}_{\mathsf{lookup}*}.\mathsf{Verify}(\mathsf{srs}, \mathsf{vk}_m, (\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \mathsf{c}_{\overleftarrow{\cdot}}, \mathsf{c}_{\overrightarrow{\cdot}}, \mathsf{c}_v), \pi_{\mathsf{zklookup}}) = 1$
2. $\mathsf{CP}_{\mathsf{range}}.\mathsf{Verify}(\mathsf{srs}, \mathsf{vk}_{(B,m,d)}, (\mathsf{c}_\mathbf{X} - \mathsf{c}_{\overleftarrow{\cdot}}), \pi^{\overleftarrow{\cdot}}_{\mathsf{range}}) = 1$ and
   $\mathsf{CP}_{\mathsf{range}}.\mathsf{Verify}(\mathsf{srs}, \mathsf{vk}_{(B,m,d)}, (\mathsf{c}_{\overrightarrow{\cdot}} - \mathsf{c}_\mathbf{X} - [1]_1), \pi^{\overrightarrow{\cdot}}_{\mathsf{range}}) = 1$
3. $\mathsf{CP}_{\mathsf{stat}}.\mathsf{Verify}(\mathsf{srs}, \mathsf{vk}_{(S,m)}, (\mathsf{c}_3, y, (\mathbf{x}_j)_{j\in[m]}), \pi_{\mathsf{stat}}) = 1$
4. $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_\mathsf{T}) = 1$.

Fig. 4. Our CP-SNARK CP_{DT}.

6.5 Efficiency and Concrete Instantiations

We discuss how to instantiate our scheme above through commit-and-prove SNARKs (lookups and general-purpose). The whole resulting system has a universal trusted setup.

- $\mathsf{CP}_{\mathsf{lookup}*}$ can be instantiated with our construction $\mathsf{mtx}[\mathsf{zkcq}^+]$ from Section 3;
- $\mathsf{CP}_{\mathsf{range}}$ can be implemented through a (vector) lookup in a table of size B where the subvector being looked up is of size m. The idea is to consider the table $\mathbf{b} = (j)_{j\in[B]}$ and prove, through a lookup argument, that that $\bar{\mathbf{x}} \prec \mathbf{b}$ where $\bar{\mathbf{x}}$ is the vectorization of $\mathbf{X}$.
- $\mathsf{CP}_{\mathsf{stat}}$ can be implemented through a general-purpose commit-and-prove SNARK, such as [2,7]. For concreteness, and to minimize proof size, in the remainder of this document, we consider the proof scheme CP-LunarLite from [7] (Section 9.4).

We can provide an upper bound on the total proof size for the instantiations above to $20\mathbb{G}_1$ elements per each of the proof above (this is a loose upper

Table 2. Comparison between our solution in Section 6.4 and Section 6.5 and [38] for zero-knowledge decision tree accuracy. Parameters are d (number of attributes), m (size of sample), $|\mathcal{H}|$ is the cost of hash function invocation (such as SHA256); $|\mathcal{H}_{\text{circ}}|$ is the cost of a hash function invocation as a circuit; $\mathbb{P}$ is the cost of one pairing. Notation $\tilde{O}(f)$ refers to $O(f \log f)$. This table does not include the one-time cost of preprocessing for the prover (see Table 1. for concrete costs). On the other hand, notice that the asymptotics in the row for our construction account for just the commitment algorithm and the *extractability* proof. The asymptotics reported for [38] are actually a lower bound and do not include some additional factors in their complexity, such as tree height. Dominated factors, such as B and k (input and output size of decision tree respectively), are also not included in the asymptotics.

Scheme	Commit Time	Prover Time	Verifier Time	Proof Size
[38]	$O(N_{\text{tot}})\|\mathcal{H}\|$	$\tilde{O}(md + N_{\text{tot}} \log m + N_{\text{tot}}\|\mathcal{H}_{\text{circ}}\|)\mathbb{F}$	$O(md)\mathbb{F}$	$O(\log^2(md))\mathfrak{f}$
Our solution	$\tilde{O}(dN_{\text{tot}})(\mathbb{G} + \mathbb{F})$	$\tilde{O}(md)(\mathbb{G} + \mathbb{F})$	$O(m)\mathbb{G} + O(1)\mathbb{P}$	$O(1)(\mathfrak{g}_1 + \mathfrak{f})$

bound)—see Table 1 in this work, Table 1 and Section 9.4 in [7]. (we use the fact that the size of field elements is at most that of $\mathbb{G}_1$ elements in this estimate) On a concrete curve like BLS12-381 this yields a total proof size of *at most* approximately 3.84KB (this is a generous lower bound). For comparison, the proof size in [38] is of the order of hundreds of kilobytes.

Decision Tree Accuracy. In the specific case of proving decision tree accuracy we prove that a decision tree is able to correctly estimate a specific fraction of a given data sample. Namely we consider the statistic that upon input $(v_j)_{j\in[m]}, (y_j)_{j\in[m]}$ computes $\sum_j \mathsf{eq}_k(v_j, y_j)/m$, $v_j = \mathsf{T}(\mathbf{x}_j)$ for $j \in [m]$ where $k \in \mathbb{N}$ is a small constant and eq_k is the function returning 1 when its two arguments, of size k, are equal[14]; otherwise it returns 0. Thanks to Theorem 4 this can be reduced to a CP-SNARK for the following relation[15]:

$$\mathcal{R}_{\mathsf{acc}} = \left\{ (m, k); \left((y_j)_{j\in[m]}, n^*\right); (v_j)_{j\in[m]} : n^* = \sum_j \mathsf{eq}_k(v_j, y_j) \right\}$$

Even with an R1CS-based (Rank-1 Constraint System) general purpose SNARK, the relation above can be implemented very efficiently. For example, an upper bound on a naive implementation is $\approx m \cdot k$ (with a very small multiplicative constant). This number accounts for implementing the equality predicate (bit decomposition and bit equality checks, done m times) and a sum of m bits (which can be described with a single constraint row in an R1CS). For representative values of m and k—respectively 2000 and 10 (see Figure 5 in [38])—this roughly corresponds to 20K constraints which results in an additional proving time of only tens of milliseconds.

[14] In typical applications of decision trees the labels are integer values belonging to a small domains, for example, either booleans or bytes.

[15] Here expressed as a sum instead of a fraction. Since the size of the sample is public this is equivalent.

Further details on estimates. We consider parameters from the largest dataset considered in [38], namely *Forest Covertype* from the UCI machine learning repository[16]. The approximated parameters are (accurate up to 5%):

- Number of nodes in the tree: $N_{\mathsf{tot}} \approx 1K$
- Number of attributes: $d \approx 50$
- Number of data samples considered in accuracy relation: $m \approx 2K$

Within these parameters (and for other parameters of interest), our proving time is dominated by multiscalar exponentiations of size $\approx 500K$ (this number is derived by dm multiplied by a small constant which depends on the number of invocations of the lookup arguments under the hood in our constructions). The prover performs roughly 30 exponentiations of this size (this is a constant and a loose upper bound). Our verification time is dominated by pairings (in the ballpark of 20). A pairing can be typically computed within $4 - 5$ms (often even faster if applying batching techniques).

Our estimates show improvements of almost one order of magnitude for proving time and two orders of magnitude for verification time. Our prover runs in the order of a few seconds (well within the half minute); our verifier in the order of 100ms. The construction in [38] has a prover running in the order of minutes (2-5m) and a verifier running in the order of 10s[17].

References

1. Ali, R.E., So, J., Avestimehr, A.S.: On polynomial approximations for privacy-preserving and verifiable relu networks (2021)
2. Aranha, D.F., Bennedsen, E.M., Campanelli, M., Ganesh, C., Orlandi, C., Takahashi, A.: ECLIPSE: Enhanced compiling method for pedersen-committed zk-SNARK engines. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022, Part I. LNCS, vol. 13177, pp. 584–614. Springer, Heidelberg (Mar 2022). https://doi.org/10.1007/978-3-030-97121-2˙21
3. Arun, A., Setty, S., Thaler, J.: Jolt: Snarks for virtual machines via lookups. Cryptology ePrint Archive, Paper 2023/1217 (2023), https://eprint.iacr.org/2023/1217, https://eprint.iacr.org/2023/1217
4. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: Verifying program executions succinctly and in zero knowledge. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 90–108. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1˙6
5. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 103–128. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2˙4
6. Bootle, J., Cerulli, A., Groth, J., Jakobsen, S.K., Maller, M.: Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part I. LNCS, vol. 11272, pp. 595–626. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03326-2˙20

[16] http://archive.ics.uci.edu/ml

[17] These estimates refer to running times on an AWS EC2 c5.9xlarge. This architecture is comparable to the one used in [38].

7. Campanelli, M., Faonio, A., Fiore, D., Querol, A., Rodríguez, H.: Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part III. LNCS, vol. 13092, pp. 3–33. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92078-4˙1
8. Campanelli, M., Fiore, D., Querol, A.: LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2075–2092. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3339820
9. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In: EUROCRYPT 2023, Part II. pp. 499–530. LNCS, Springer, Heidelberg (Jun 2023). https://doi.org/10.1007/978-3-031-30617-4˙17
10. Chen, H., Zhang, H., Si, S., Li, Y., Boning, D.S., Hsieh, C.: Robustness verification of tree-based models. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) NeurIPS 2019. pp. 12317–12328. Curran Associates, Inc., Red Hook, NY, USA (December 2019), https://proceedings.neurips.cc/paper/2019/hash/cd9508fdaa5c1390e9cc329001cf1459-Abstract.html
11. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., Ward, N.P.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 738–768. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45721-1˙26
12. Choudhuri, A.R., Garg, S., Goel, A., Sekar, S., Sinha, R.: Sublonk: Sublinear prover plonk. Cryptology ePrint Archive, Paper 2023/902 (2023), https://eprint.iacr.org/2023/902
13. Eagen, L., Fiore, D., Gabizon, A.: cq: Cached quotients for fast lookups. Cryptology ePrint Archive, Report 2022/1763 (2022), https://eprint.iacr.org/2022/1763
14. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat-Shamir transform. In: Galbraith, S.D., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 60–79. Springer, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34931-7˙5
15. Feng, B., Qin, L., Zhang, Z., Ding, Y., Chu, S.: ZEN: An optimizing compiler for verifiable, zero-knowledge neural network inferences. Cryptology ePrint Archive, Report 2021/087 (2021), https://eprint.iacr.org/2021/087
16. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0˙2
17. Gabizon, A., Williamson, Z.J.: plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Report 2020/315 (2020), https://eprint.iacr.org/2020/315
18. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019), https://eprint.iacr.org/2019/953
19. Ganesh, C., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 397–426. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3˙14

20. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5˙11
21. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 698–728. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96878-0˙24
22. Haböck, U.: Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive, Report 2022/1530 (2022), https://eprint.iacr.org/2022/1530
23. Kang, D., Hashimoto, T., Stoica, I., Sun, Y.: Scaling up trustless dnn inference with zero-knowledge proofs (2022)
24. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (Dec 2010). https://doi.org/10.1007/978-3-642-17373-8˙11
25. Lee, S., Ko, H., Kim, J., Oh, H.: vCNN: Verifiable convolutional neural network. Cryptology ePrint Archive, Report 2020/584 (2020), https://eprint.iacr.org/2020/584
26. Lipmaa, H.: Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 169–189. Springer, Heidelberg (Mar 2012). https://doi.org/10.1007/978-3-642-28914-9˙10
27. Lipmaa, H., Siim, J., Zajac, M.: Counting vampires: From univariate sumcheck to updatable ZK-SNARK. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part II. LNCS, vol. 13792, pp. 249–278. Springer, Heidelberg (Dec 2022). https://doi.org/10.1007/978-3-031-22966-4˙9
28. Liu, T., Xie, X., Zhang, Y.: zkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 2968–2985. ACM Press (Nov 2021). https://doi.org/10.1145/3460120.3485379
29. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2111–2128. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3339817
30. Posen, J., Kattis, A.A.: Caulk+: Table-independent lookup arguments. Cryptology ePrint Archive, Report 2022/957 (2022), https://eprint.iacr.org/2022/957
31. Ràfols, C., Zapico, A.: An algebraic framework for universal and updatable SNARKs. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 774–804. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0˙27
32. Setty, S.: Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 704–737. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56877-1˙25
33. Setty, S., Thaler, J., Wahby, R.: Unlocking the lookup singularity with lasso. Cryptology ePrint Archive, Paper 2023/1216 (2023), https://eprint.iacr.org/2023/1216, https://eprint.iacr.org/2023/1216
34. Wang, H., Hoang, T.: ezdps: An efficient and zero-knowledge machine learning inference pipeline. PoPETs **2023**(2), 430–448 (2023). https://doi.org/10.56553/popets-2023-0061

35. Weng, J., Weng, J., Tang, G., Yang, A., Li, M., Liu, J.: pvcnn: Privacy-preserving and verifiable convolutional neural network testing. IEEE Trans. Inf. Forensics Secur. **18**, 2218–2233 (2023). https://doi.org/10.1109/TIFS.2023.3262932
36. Zapico, A., Buterin, V., Khovratovich, D., Maller, M., Nitulescu, A., Simkin, M.: Caulk: Lookup arguments in sublinear time. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 3121–3134. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3560646
37. Zapico, A., Gabizon, A., Khovratovich, D., Maller, M., Ràfols, C.: Baloo: Nearly optimal lookup arguments. Cryptology ePrint Archive, Report 2022/1565 (2022), https://eprint.iacr.org/2022/1565
38. Zhang, J., Fang, Z., Zhang, Y., Song, D.: Zero knowledge proofs for decision tree predictions and accuracy. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 2039–2053. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3417278

Supplementary Material

A On Applying Other Backends to [38]

Here we elaborate on why applying different backends (e.g., Groth16 [20] or Marlin [11]) does not substantially change our comparison with [38]. This is particularly true for proving time, for which the main bottleneck of the approach in [38] lies in the number of constraints, which stems from the essence of their "tree-visiting" approach. This results in dependence on the size of the hash function among other metrics, as discussed in the introduction.

We verify this claim by running [38] on Groth16—one of the most optimized, highly succinct proof systems—instead of Aurora (the main backend originally described [38]) and using their original code[18]. When running it over Groth16 we observe that the scheme in [38] runs in ≈ 3 minutes. If one wanted to use a universal setup scheme (e.g., Marlin) these numbers would be at least 4x as large. As a comparison, we recall that the ballpark of our scheme is 15-30s (pg30). This results in approximately an order of magnitude difference between our scheme and Marlin applied to [38]. Note that this comparison is apple-to-apple: it is on the same architecture, on the parameters referred at 31, and using the original code for [38].

The above observations are for proving time. For other dimensions—verification and proof size—applying a scheme like Marlin to [38] would obtain numbers close to ours. For proof size these numbers may be marginally better than ours.

B Additional Material on Section 2

Definition 11 (Universal CP-SNARK). *A universal CP-SNARK for an indexed relation $\mathcal{R}$ is a tuple of algorithms $\Pi = (\mathsf{KGen}, \mathsf{Derive}, \mathsf{Prove}, \mathsf{Verify})$ where*

- Derive *is a deterministic algorithm that takes as input an* srs *(which includes relation parameters* pp*) produced by* KGen*, an index ind, and outputs specialized SRS* $\mathsf{srs}_{ind} = (\mathsf{ek}_{ind}, \mathsf{vk}_{ind})$*. The length of* vk_{ind} *is* $\mathsf{poly}(\lambda, \log |ind|)$*.*
- *Consider the relation $\mathcal{R}'$ such that $\mathcal{R}'(\mathsf{pp}, (ind, x), w) \iff \mathcal{R}(\mathsf{pp}, ind, x, w)$, the tuple of algorithms $(\mathsf{KGen}, \mathsf{Prove}, \mathsf{Verify}')$ is an argument system for the relation $\mathcal{R}'$ and Verify' is the algorithm that upon input* srs*, instance (ind, x) and a proof π, first runs* Derive *on* srs *and index ind, then runs* $\mathsf{Verify}(\mathsf{vk}_{ind}, x, \pi)$*.*

C Additional Material on Section 4

Lemma 2. $\sum_{j=0}^{N} A(\omega_N^{j-1}) = \sum_{i=0}^{n} B(\omega_n^{i-1})$ *iff* $\sum_{j=0}^{N} C(\omega_N^{j-1}) = 0$.

Proof. Recall that we denote $C(X) := A(X) - \frac{1}{\vartheta} B(X)\mathsf{z}(X)$, $\vartheta := N/n$ and $\mathsf{z}(X) := \nu_{\mathbb{K}\setminus\mathbb{H}}(X)$. Define

$$\Delta(X) := A(X) - C(X) = \tfrac{1}{\vartheta} B(X)\mathsf{z}(X) \ .$$

[18] Available at https://github.com/TAMUCrypto/ZKDT_release.

For any $j \in [N]$, $\Delta(\omega_N^{j-1}) = \frac{1}{\vartheta} B(\omega_N^{j-1}) \mathsf{z}(\omega_N^{j-1})$. Now, $\mathsf{z}(\omega_N^{j-1}) = 0$ when $\omega_N^{j-1} \notin \mathbb{H}$ (i.e., $\vartheta \nmid j-1$) and $\mathsf{z}(\omega_N^{j-1}) = \vartheta$ otherwise. Hence,

$$\Delta(\omega_N^{j-1}) = \begin{cases} 0 \ , & \omega_N^{j-1} \notin \mathbb{H} \quad (\text{i.e., } \vartheta \nmid j-1) \ , \\ B(\omega_N^{j-1}) \ , & \omega_N^{j-1} \in \mathbb{H} \quad (\text{i.e., } \vartheta \mid j-1) \ . \end{cases}$$

Writing $j - 1 = (i-1)\vartheta$ in the last case, we get that when $\vartheta \mid j-1$,

$$\Delta(\omega_N^{j-1}) = B(\omega_N^{j-1}) = B(\omega_N^{(i-1)\vartheta}) = B(\omega_n^{i-1}) = B_i \ .$$

Thus,

$$\Delta(\omega_N^{j-1}) = \begin{cases} 0 \ , & \vartheta \nmid (j-1) \ , \\ B(\omega_n^{i-1}) \ , & j-1 = (i-1)\vartheta \ . \end{cases}$$

Hence, the "low-degree part" of $\Delta(X)$ only depends on the values of $B(X)$ on the subgroup $\mathbb{H}$ and hence, we do not have to perform a low-degree test on $B(X)$. Moreover, $\sum_{j=0}^{N} \Delta(\omega_N^{j-1}) = \sum_{i=0}^{n} B(\omega_n^{i-1})$. Thus, $\sum_{j=0}^{N} C(\omega_N^{j-1}) = \sum_{j=0}^{N} A(\omega_N^{j-1}) - \sum_{i=0}^{n} B(\omega_n^{i-1})$. This proves the claim. □

C.1 Security Proofs of cq^+

Knowledge-Soundness. We prove knowledge-soundness in the AGM under the standard PDL assumption. We give a complete proof for cq^+; the proof for cq^{++} follows from that and the known polynomial commitment batching lemmas.

Definition 12 (Power Discrete Logarithm [26]). *Let $d_1(\lambda), d_2(\lambda) \in \mathsf{poly}(\lambda)$. A bilinear group generator $\mathsf{GroupGen}$ is (d_1, d_2)-*PDL (Power Discrete Logarithm) secure *if for any non-uniform PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{GroupGen},\mathcal{A}}(\lambda) :=$*

$$\Pr\left[\mathsf{pp} \leftarrow \mathsf{GroupGen}(1^\lambda); s \leftarrow\$ \mathbb{F}^* : \mathcal{A}\left(\mathsf{pp}, \left[(s^i)_{i=0}^{d_1}\right]_1, \left[(s^i)_{i=0}^{d_2}\right]_2\right) = s \right] = \mathsf{negl}(\lambda) \ .$$

Theorem 5. *Assume $U(X) \nmid X$. Let $N_1 \geq N + b - 1$ and $N_2 \geq N + 1$. Then the interactive protocol cq^+ from Fig. 1 is knowledge-sound in the AGM under the (N_1, N_2)-PDL assumption.*

Proof. Let $\mathcal{A}$ be an algebraic knowledge-soundness adversary that, after interacting with the honest verifier, outputs the following group elements:

$$[m(s), A(s), B(s), Q_B(s), S(s), P(s), R_C^*(s), Q(s)]_1 \ ,$$

with explanations, showing that the outputs are evaluations of polynomials $m, A, B, Q_B, S, P, R_C^*, Q$ that all have degree $\leq N_1$. Moreover, $m(X)$ and $S(X)$ do not depend on β, γ, η, while $(A(X), B(X), Q_B(X))$ depend on β, and $(P(X), R_C^*(X), Q(X))$ depend on β, γ, η. The adversary also returns one field element B_γ (the claimed value of $B(\gamma)$).

As usual in the AGM proofs, one has two cases: an information-theoretic and a computational case (a reduction to PDL). One writes down a polynomial form of the verifier's equations (two equations V_1 and V_2 in the current case), such that the verifier accepts iff $V_1(s) = V_2(s) = 0$. Here, the coefficients of V_1 and V_2 can be computed from the outputs of the extractor. In the information-theoretic case, we consider the possibility that both V_1 and V_2 are zero polynomials and show that $\mathcal{A}$ must have been honest. In the computational case, we analyze the case that either $V_1(X)$ or V_2 is a non-zero polynomial, but the verifier still accepts, i.e., $V_1(s) = V_2(s) = 0$. We then construct a reduction to the security of PDL.

Information-theoretic case. By the second verification equation in Fig. 1,

$$V_2(X) := B(X) + \eta D(X) - B_\gamma - P(X) \cdot (X - \gamma)$$

is a zero polynomial. Thus,

$$(X - \gamma) \mid (B(X) + \eta D(X) - B_\gamma) \ .$$

Since neither $B(X)$ or $D(X)$ depends on η, we get by the Schwartz-Zippel lemma that $(X-\gamma) \mid (B(X)-B_\gamma)$ and $(X-\gamma) \mid D(X)$. Hence, $B(\gamma) = B_\gamma$ and $D(\gamma) = 0$. Recalling that $D(X) = B(\gamma)(\mathsf{F}(X) + \beta) - 1 - Q_B(X)\nu_{\mathbb{H}}(\gamma)$, we get

$$B(\gamma)(\mathsf{F}(\gamma) + \beta) - 1 = Q_B(\gamma)\nu_{\mathbb{H}}(\gamma) \ .$$

Since neither $B(X)$ or $Q_B(X)$ depends on γ, by the Schwartz-Zippel lemma,

$$B(X)(\mathsf{F}(X) + \beta) - 1 = Q_B(X)\nu_{\mathbb{H}}(X)$$

as a polynomial. Matching the left and right-hand sides for the values of $X = \omega_n^{i-1}$, we get

$$B_i = \frac{1}{\mathbf{f}_i + \beta} \ .$$

Next, from the first verification equation, we get that the following polynomial $V_1(X)$ is a zero polynomial:

$$V_1(X) := \begin{pmatrix} A(X)\mathsf{T}(X) + (\beta + \eta)A(X) - m(X) - \\ \frac{\eta}{\vartheta} B(X)\mathsf{z}(X) + \eta^2 S(X) - Q(X)\nu_{\mathbb{K}}(X) \end{pmatrix} \cdot U(X) - \eta R_C^*(X)X \ .$$

Since $U(X) \nmid X$, we get that $U(X) \mid R_C^*(X)$. Since $\deg U(X) = \mu = N_1 - N + 2$, $R_C^*(X) = R_C(X)U(X)$ for some polynomial $R_C(X)$ of degree $\leq N - 2$. Thus,

$$\begin{aligned} A(X)\mathsf{T}(X) + (\beta + \eta)A(X) - m(X) - \tfrac{\eta}{\vartheta} B(X)\mathsf{z}(X) + \eta^2 S(X) \\ = \eta R_C(X)X + Q(X)\nu_{\mathbb{K}}(X) \ . \end{aligned}$$

Recalling that $C(X) = A(X) - \frac{1}{\vartheta} B(X)\mathsf{z}(X)$, we get

$$A(X)(\mathsf{T}(X) + \beta) - m(X) + \eta C(X) + \eta^2 S(X) = \eta R_C(X)X + Q(X)\nu_{\mathbb{K}}(X) \ .$$

Here, only $Q(X) = Q(X, \eta)$ and $R_C(X) = R_C(X, \eta)$ depend on η (and γ) while other polynomials do not. Let Y be the indeterminate corresponding to η, and let us write Q and R_C as bivariate polynomials in X and Y. By applying the Schwartz-Zippel lemma again,

$$A(X)(\mathsf{T}(X)+\beta)-m(X)+YC(X)+Y^2S(X) = YR_C(X,Y)X+Q(X,Y)\nu_{\mathbb{K}}(X)$$

as a polynomial.

Setting $Y = 0$, we get

$$A(X)(\mathsf{T}(X) + \beta) - m(X) = Q(X, 0)\nu_{\mathbb{K}}(X) \ .$$

Considering the values of $X = \omega_N^{j-1}$ again, we get

$$A_j = \frac{m_j}{\mathbf{t}_j + \beta} \ .$$

On the other hand, define $\widehat{Q}(X) := (Q(X, Y) - Q(X, 0))/Y$ and $\widehat{r}_C(X) := (R_C(X, Y) - R_C(X, 0))/Y$. Thus,

$$\begin{aligned} &A(X)(\mathsf{T}(X) + \beta) - m(X) + YC(X) + Y^2S(X) \\ &\quad = Y\left(\widehat{r}_C(X, Y)Y + R_C(X, 0)\right) \cdot X + \left(\widehat{Q}(X, Y)Y + Q(X, 0)\right) \cdot \nu_{\mathbb{K}}(X) \ . \end{aligned}$$

Ignoring addends that do not depend on Y, we get

$$C(X) + YS(X) = (\widehat{r}_C(X, Y)Y + R_C(X, 0)) \cdot X + \widehat{Q}(X, Y)\nu_{\mathbb{K}}(X) \ \textit{enspace.}$$

Replace now Y by any constant, say $Y = 0$. Thus,

$$C(X) = R_C(X, 0)X + \widehat{Q}(X, 0)\nu_{\mathbb{K}}(X) \ .$$

Since $\deg_X R_C(X, Y) \leq N - 2$, by the Aurora's sumcheck,

$$\sum_{j=1}^{N} C_j = 0 \ .$$

By Lemma 2,

$$\sum_{j=1}^{N} A_j = \sum_{i=1}^{n} B_i \ .$$

Next, we already expressed A_j and B_i as $m_j/(\mathbf{t}_j + \beta)$ and $1/(\mathbf{f}_j + \beta)$. Thus,

$$\sum_{j=1}^{N} \frac{m_j}{\mathbf{t}_j + \beta} = \sum_{i=1}^{n} \frac{1}{\mathbf{f}_j + \beta} \ .$$

Since m_j does not depend on β, we can apply the Schwartz-Zippel lemma, obtaining

$$\sum_{j=1}^{N} \frac{m_j}{\mathbf{t}_j + X} = \sum_{i=1}^{n} \frac{1}{\mathbf{f}_i + X} \ .$$

By Haböck's lemma (Lemma 1), this means that $\{\mathbf{f}_i\} \subseteq \{\mathbf{t}_j\}$. Hence, the adversary is honest. This proves the information-theoretical case.

Computational case. In this case, one of V_1 and V_2 is a non-zero polynomial but $V_1(s) = V_2(s) = 0$. W.l.o.g., assume $V_1 \neq 0$ (since $\deg V_1 > \deg V_2$, this results in a stronger PDL assumption). Note that $\deg V_1 \leq N_1 + N + (N_1 - N + 2) = 2(N_1 + 1)$. We construct the following (N_1, N_2)-PDL adversary $\mathcal{B}$. $\mathcal{B}\left(\left[(s^i)_{i=0}^{N_1}\right]_1, \left[(s^i)_{i=0}^{N_2}\right]_2\right)$ uses its as the SRS for $\mathcal{A}$. $\mathcal{B}$ then invokes $\mathcal{A}$, playing the honest verifier and obtaining a protocol transcript. Since $\mathcal{A}$ is algebraic, it returns explanations, i.e., polynomials $m, \ldots, Q$. Given these polynomials, $\mathcal{B}$ can compute all coefficients of V_1. Hence, $\mathcal{B}$ has a known non-zero polynomial V_1 of degree $2(N_1+1)$, such that $V_1(s) = 0$. $\mathcal{B}$ now uses a standard probabilistic polynomial-time root-finding algorithm over finite fields to obtain all roots of V_1, and tests which root equals s by using its input. Hence, $\mathcal{B}$ can compute s and thus break (N_1, N_2)-PDL.

Analyzing both cases finishes the proof. □

Zero-Knowledge. We prove the zero-knowledge property of cq^+; the proof for cq^{++} is nearly identical and is omitted.

Theorem 6. *The interactive protocol* cq^+ *from Fig. 1 is honest-verifier zero-knowledge.*

Proof. To prove the theorem we first describe the simulator and then argue why its simulation is indistinguishable from a honestly generated proof.

We present the simulator in Fig. 5. What Sim does is to generate the commitments $[m(s), A(s), B(s)]_1$ as in the prover algorithm but by setting $A_j = B_i = m_j = 0$. Next, it samples the remaining elements following the correct distribution that makes the verification equations accept.

Let us argue about each element, output by the simulator:

- $m(s)$ (resp. $A(s)$) is masked by random value ρ_m (resp. ρ_A); this perfectly hides $m(X)$ (resp. $A(X)$) since only one evaluation of it (at $X = s$) is known.
- $B(s)$ is masked by a degree-1 random polynomial; this perfectly hides $B(X)$ since only two evaluations of it (at $X = s$ and $X = \gamma$) are known.
- $Q_B(s)$ is computed as in the interactive protocol.
- $S(s)$ is computed as in the interactive protocol.
- $P(s)$ is computed as in the interactive protocol.
- $R_C^*(s)$ is computed as in the interactive protocol in the case $A_j = B_i = 0$. Note that the pair $(S(s), R_C^*(s))$ is uniformly random due to the choice of R_S and ρ_S.
- $Q(s)$ is chosen so that it makes the verifier accept. To check it, let us rewrite the first verification equation in cq^{++} but in discrete logarithms:

$$A(s)\mathsf{T}(s)U(s) + \left((\beta+\eta)A(s) - m(s) + \eta^2 S(s)\right)U(s) - \frac{\eta}{\vartheta}B(s)\mathsf{z}(s)U(s) - Q(s)\nu_{\mathbb{K}}(s)U(s) = \eta R_C^*(s)s$$

$\mathsf{Sim}(s, (N, n), (\mathsf{c_t}, \mathsf{c_f}), [\mathsf{T}(s)]_1)$:

1. $\rho_m \leftarrow\!\$\ \mathbb{F}$; $m(s) \leftarrow \rho_m \nu_{\mathbb{K}}(s)$;
2. $R_S, \rho_S \leftarrow\!\$\ \mathbb{F}$; $S(X) \leftarrow X R_S(X) + \rho_S \nu_{\mathbb{K}}(X)$;
3. Send $[m(s), S(s)]_1$;

---

4. Obtain β;
5. $\rho_A \leftarrow\!\$\ \mathbb{F}$; $A(s) \leftarrow \rho_A \nu_{\mathbb{K}}(s)$;
6. $\rho_B(X) \leftarrow\!\$\ \mathbb{F}_{\leq 1}[X]$; $B(s) \leftarrow \rho_B(s) \nu_{\mathbb{K}}(s)$;
7. $[Q_B(s)]_1 \leftarrow (B(s)(\mathsf{c_f} + \beta [1]_1) - 1)/\nu_{\mathbb{H}}(s)$;
8. Send $[A(s), B(s), Q_B(s)]_1$;

---

9. Obtain γ, η;
10. $[D(s)]_1 \leftarrow B(\gamma)(\mathsf{c_f} + \beta [1]_1) - 1 - \nu_{\mathbb{H}}(\gamma) \cdot [Q_B(s)]_1$;
11. // Note that $D(s) = \left(B(\gamma) - \frac{B(s)\nu_{\mathbb{H}}(\gamma)}{\nu_{\mathbb{H}}(s)}\right)(\mathsf{F}(s) + \beta) - 1 + \frac{\nu_{\mathbb{H}}(\gamma)}{\nu_{\mathbb{H}}(s)}$;
    // Replacing s with γ, we get that $D(\gamma) = 0$;
    Claim $D_\gamma \leftarrow 0$;
    Claim $B_\gamma \leftarrow B(\gamma) = \rho_B(\gamma)\nu_{\mathbb{K}}(\gamma)$;
12. $[P(s)]_1 \leftarrow \frac{1}{s-\gamma}((B(s) - B(\gamma))[1]_1 + \eta [D(s)]_1)$;
13. $R^*_C(s) \leftarrow \eta \cdot R_S \cdot U(s)$;
14. Choose $Q(s)$ that makes the verifier accept:
    $[Q(s)]_1 \leftarrow \rho_A(s)(\beta + \eta + [\mathsf{T}(s)]_1) - \frac{\eta}{\vartheta}\rho_B(s) - \rho_m(s) + \eta^2 \rho_S(s)$;
15. Send $[P(s), R^*_C(s), Q(s)]_1, B_\gamma$.

Fig. 5. Simulator of cq^+

Writing in simulator chosen $A(X)$, $B(X)$, $m(X)$, $S(X)$, and $R^*_C(X)$, this is equivalent to

$$\begin{aligned}\rho_A(s)\nu_{\mathbb{K}}(s)\mathsf{T}(s)U(s) + \\ \left((\beta+\eta)\rho_A(s)\nu_{\mathbb{K}}(s) - \rho_m(s)\nu_{\mathbb{K}}(s) + \eta^2\left(sR_S(s) + \nu_{\mathbb{K}}(s)\rho_S(s)\right)\right)U(s) - \\ \frac{\eta}{\vartheta}\rho_B(s)\nu_{\mathbb{K}}(s)U(s) - Q(s)\nu_{\mathbb{K}}(s)U(s) = \eta^2 R_S(s)U(s)s\end{aligned}$$

Cancelling $\eta^2 R_S(s)U(s)s$ and dividing the rest by $\nu_{\mathbb{K}}(s)U(X)$, it is equivalent to

$$Q(s) = (\beta + \eta + \mathsf{T}(s))\rho_A(s) - \frac{\eta}{\vartheta}\rho_B(s) - \rho_m(s) + \eta^2\rho_S(s) \ .$$

C.2 Our fully zero-knowledge lookup argument

We describe the protocol zkcq^+ in Fig. 6. We remark that $\mathsf{Preproc}$ computes the value $\tilde{\mathsf{c}}_\mathbf{t}$ which is included in the proof. This is just syntatic sugar, and it is only necessary for matching the syntax of CP-SNARK. In practical implementations, the value $\tilde{\mathsf{c}}_\mathbf{t}$ could be posted together with $\mathsf{c_t}$, and their well-formedness could verified only once.

$\underline{\mathsf{Derive}(\mathsf{srs}, N, n)}$: // Assume that $n \mid N$, $\mathsf{srs} = (\left[(s^j)_{j\in[N_1]}\right]_1, \left[(s^j)_{j\in[N_2]}\right]_2)$ for any $N_1 \geq N + \max(\mathsf{b}_F, 1) - 1$ and $N_2 \geq N + \max(\mathsf{b}_F, 1) + 1$.

Set $\mu = N_1 - N + 2$; define $U(X) := (X^\mu - 1)$, $\vartheta = N/n$, and $\mathsf{z}(X) = \nu_{\mathbb{K}\setminus\mathbb{H}}(X)$;

Let $\left\{r_j^{\mathbb{K}}(X) = \frac{\lambda_j^{\mathbb{K}}(X) - \lambda_j^{\mathbb{K}}(0)}{X} U(X)\right\}_{j\in[N]}$, and $\left\{r_i^{\mathbb{H}}(X) = \frac{\lambda_i^{\mathbb{H}}(X)\mathsf{z}(X) - \lambda_i^{\mathbb{H}}(0)}{X} U(X)\right\}_{i\in[n]}$;

Compute $\mathsf{ek}_{N,n} := \left[(r_j^{\mathbb{K}}(s))_{j=1}^N, (r_i^{\mathbb{H}}(s))_{i=1}^n, U(s), \nu_{\mathbb{K}}(s), s\nu_{\mathbb{K}}(s), (x^i)_{i=0}^{n+\mathsf{b}_F-1}\right]_1$

Compute $\mathsf{vk}_{N,n} := [1, U(s), \mathsf{z}(s)U(s), \nu_{\mathbb{K}}(s)U(s)]_2$

Return $(\mathsf{ek}_{N,n}, \mathsf{vk}_{N,n})$.

$\underline{\mathsf{Preproc}(\mathsf{srs}, \mathbf{t})}$:

Define $\mathsf{T}(X) := \sum_{j=1}^N = \mathbf{t}_j \lambda_j^{\mathbb{K}}(X) + \rho_{\mathsf{T}} \cdot \nu_{\mathbb{K}}(X)$, and compute $\tilde{\mathsf{c}}_{\mathbf{t}} \leftarrow [\mathsf{T}(s)U(s)]_2$;

Compute $\mathsf{aux}_{\mathbf{t}} := (\mathbf{t}, \left[(Q_j(s))_{j=1}^N, \mathsf{T}(s)\right]_1)$;

Return $(\mathsf{aux}_{\mathbf{t}}, \tilde{\mathsf{c}}_{\mathbf{t}})$.

$\underline{\mathsf{Prove}(\mathsf{ek}_{N,n}, (\mathsf{c}_{\mathbf{t}}, \mathsf{c}_{\mathbf{f}}), \mathsf{aux}_{\mathbf{t}}, (\mathbf{f}, \rho_{\mathsf{F}}(X)))}$: //$\mathsf{c}_{\mathbf{f}} = \left[\sum_i \mathbf{f}_i \lambda_i^{\mathbb{H}}(s) + \rho_{\mathsf{F}}(s)\nu_{\mathbb{H}}(s)\right]_1$, $deg(\rho_{\mathsf{F}}) = \mathsf{b}_F$.

Compute $\mathbf{m} = (m_1, \ldots, m_N)$ s.t. $\forall j : \mathbf{t}_j$ appears m_j times in $\mathbf{f}$; samples $\rho_m \leftarrow\$ \mathbb{F}$;

Compute $[m(s)]_1 \leftarrow \sum_{j=1}^N m_j \cdot \left[\lambda_j^{\mathbb{K}}(s)\right]_1 + \rho_m \cdot [\nu_{\mathbb{K}}(s)]_1$; // n scalar mults

Sample $R_S, \rho_S \leftarrow\$ \mathbb{F}$ and compute $[S(s)]_1 \leftarrow R_S \cdot s + \rho_S \cdot \nu_{\mathbb{K}}(s)$;

$\beta \leftarrow \mathsf{RO}(\mathsf{vk}_{N,n} \| (\mathsf{c}_{\mathbf{t}}, \mathsf{c}_{\mathbf{f}}) \| [m(s)]_1)$ //Fiat-Shamir challenge.

Sample $\rho_A \leftarrow\$ \mathbb{F}$, $\rho_B(X) \leftarrow\$ \mathbb{F}_{\leq 1}[X]$;

Let $A_j \leftarrow m_j/(\mathbf{t}_j + \beta)\ \forall j = 1, \ldots, N$ and $B_i \leftarrow 1/(\mathbf{f}_i + \beta)\ \forall i = 1, \ldots, n$;

Compute $[A(s)]_1 \leftarrow \sum_{j=1}^N A_j \left[\lambda_j^{\mathbb{K}}(s)\right]_1 + \rho_A \cdot [\nu_{\mathbb{K}}(s)]_1$;

Compute $[B(s)]_1 \leftarrow \sum_{i=1}^n B_i \left[\lambda_i^{\mathbb{H}}(s)\right]_1 + \rho_B(s) \cdot [\nu_{\mathbb{H}}(s)]_1$;

Compute $Q_B(X) \leftarrow (B(X)(\mathsf{F}(X) + \beta) - 1)/\nu_{\mathbb{H}}(X)$ and $[Q_B(s)]_1$;

$(\gamma, \eta) \leftarrow \mathsf{RO}(\beta \| [A(s), B(s), Q_B(s), S(s)]_1)$;//Fiat-Shamir challenge.

Compute $B_\gamma \leftarrow B(\gamma)$, $D(X) \leftarrow B_\gamma \cdot (\mathsf{F}(X) + \beta) - 1 - Q_B(X)\nu_{\mathbb{H}}(\gamma)$;

Compute $P(X) \leftarrow ((B(X) - B(\gamma)) + \eta D(X))/(X - \gamma)$ and $[P(s)]_1$;

Compute $[R_C^*(s)]_1 \leftarrow \sum_{m_j \neq 0} A_j \cdot \left[r_j^{\mathbb{K}}(s)\right]_1 - \vartheta^{-1} \sum_{i=1}^n B_i \cdot \left[r_i^{\mathbb{H}}(s)\right]_1 + \eta R_S \cdot [U(s)]_1$

Compute $[Q_A(s)]_1 \leftarrow \sum_{m_j \neq 0} A_j \cdot [Q_j(s)]_1 + [\rho_A(T(s) + \beta) - \rho_m]_1$;

Compute $[Q_C(s)]_1 \leftarrow \left[\rho_A + \vartheta^{-1}\rho_B(s)\right]_1$;

Compute $[Q(s)]_1 \leftarrow [Q_A(s)]_1 + \eta\,[Q_C(s)]_1 - \eta^2\,[\rho_S]_1$;

Return $\pi = (\tilde{\mathsf{c}}_{\mathbf{t}}, [m(s), S(s), A(s), B(s), Q_B(s), P(s), R_C^*(s), Q(s)]_1, B_\gamma)$.

$\underline{\mathsf{Verify}(\mathsf{vk}_{N,n}, (\mathsf{c}_{\mathbf{t}}, \mathsf{c}_{\mathbf{f}}), \pi)}$:

Compute $[D(s)]_1 \leftarrow B_\gamma(\mathsf{c}_{\mathbf{f}} + [\beta]_1) - [1]_1 - \nu_{\mathbb{H}}(\gamma)\,[Q_B(s)]_1$.

Return 1 if and only if the following holds:

(i) $e([A(s)]_1, \mathsf{c}_{\mathbf{t}}) \cdot e((\beta + \eta) \cdot [A(s)]_1 - [m(s)]_1 + \eta^2\,[S(s)]_1, [U(s)]_2) \cdot e(\eta/\vartheta \cdot [B(s)]_1, [\mathsf{z}(s)U(s)]_2)^{-1} \cdot e([Q(s)]_1, [\nu_{\mathbb{K}}(s)U(s)]_2)^{-1} = e(\eta \cdot [R_C^*(s)]_1, [x]_2)$,

(ii) $e([B(s)]_1 + \eta\,[D(s)]_1 - [B_\gamma]_1, [1]_2) = e([P(s)]_1, [s - \gamma]_2)$

(iii) $e([1]_1, \tilde{\mathsf{c}}_{\mathbf{t}}) = e(\mathsf{c}_{\mathbf{t}}, [U(s)]_2)$

Fig. 6. Our fully zero-knowledge lookup argument zkcq^+.

D Additional Material on Section 5

Theorem 2. *The lookup argument* $\mathsf{mtx}[\mathsf{CP}]$ *defined in Fig. 2 is knowledge-sound in the AGM and ROM under the* $(N \cdot d, N \cdot d)$*-PDL assumption and assuming that* CP *is knowledge-sound. Furthermore, the protocol is zero-knowledge assuming* CP *is zero-knowledge.*

Proof. Notice that the vectorization operator is linear, moreover that $\mathsf{Preproc}$ applies a linear function to the precomputation through CP of the vectorizations of the matrices $\mathbf{T}, \mathbf{R}, \mathbf{C}^{(N)}$. Thus the $\mathsf{Preproc}$ is $\mathbb{F}$-linear when $\mathsf{Preproc}'$ is $\mathbb{F}$-linear.

The algorithm Prove makes $O(n \log n)$ field operations and group multiplications and additions to compute $\sigma, w, \pi_R, \pi_{R'}$ and makes $O(n)$ field operations to compute z. The proofs π_R and $\pi_{R'}$ take $O(n \log n)$ field operations and group multiplications and additions. Notice that Prove does not need to compute $\bar{\mathbf{t}}^*, \bar{\mathbf{f}}^*$ to compute the proof π^*. In fact, to compute such a proof, the prover needs only:

$$(\mathsf{aux}^*_j)_{j \in K} = (\mathsf{aux}_{M,j})_{j \in K} + \rho \cdot (\mathsf{aux}_{C,j})_{j \in K} + \rho^2 \cdot (\mathsf{aux}_{R,j})_{j \in K},$$

which, thanks to the $\mathbb{F}$-linearity of $\mathsf{CP.Preproc}$, are valid auxiliary information for $(\bar{\mathbf{t}}^*, \bar{\mathbf{f}}^*)$ and then it runs the prover of CP whose running time is $\mathsf{poly}(n \cdot d, \lambda)$.

Completeness is almost straightforward. Let $K = \{k_1, \ldots, k_n\}$ be such that $\mathbf{T}_{|K} = \mathbf{F}$ notice that for any i and for any j we have $\mathbf{T}_{k_i,j} + \rho j + \rho^2 k_i = \mathbf{F}_{i,j} + \rho j + \rho^2 \mathbf{S}_{i,j}$. In fact, $\mathbf{T}_{k_i,j} = \mathbf{F}_{i,j}$ by the hypothesis for completeness and $\mathbf{S}_{i,j} = k_i$ by inspection of the prover's algorithm. Moreover, we have that $\sigma(\omega X) - \sigma(X)$ evaluated in ω^{di+j} is equal to $\mathbf{S}_{i,j+1} - \mathbf{S}_{i,j} = 0$ for $j \in \{1, d-1\}$ and $\mathbf{S}_{i+1,0} - \mathbf{S}_{i,d}$ otherwise. Thus the polynomial $\sigma(\omega X) - \sigma(X)$ is divisible by $\nu_{\bar{\mathbb{H}}}(X)$.

As for zero-knowledge, by randomizing σ by summing $\nu_{\mathbb{K}}(X) \cdot r(X)$ where r is a random polynomial with $\deg(r) = 2$, we have that the values $\sigma(s), \sigma(\zeta), \sigma(\omega \cdot s)$ are uniformly distributed. In particular, one could sample random value z and random group elements for $\mathsf{c}_{R,n}, \mathsf{c}_{R',n}$ and use the zero-knowledge simulator of the proof of evaluation and the zero-knowledge simulator of CP.

We focus on knowledge soundness in the AGM. Notice that the prover and verifier algorithm define a one-round public-coin protocol where we applied the Fiat-Shamir transform.

The adversary outputs valid representations for the proof elements that we can parse as polynomials. Let $\tilde{\sigma}_1(X)$ and $\tilde{\sigma}_2(X)$ be the polynomials underlying the commitments $\mathsf{c}_{R,n}$ and $\mathsf{c}_{R',n}$ notice that by the verification equations in Items (i) and (ii) there exist two polynomials W_1 and W_2 such that:

$$\tilde{\sigma}_1(X) - z = W_1(X)(X - \omega \cdot \zeta) \tag{27}$$
$$\tilde{\sigma}_2(X) - z = W_2(X)(X - \zeta) \tag{28}$$

By change of variable we have $\tilde{\sigma}_1(\omega X) - z = W_1(\omega X)(\omega \cdot X - \omega \cdot \zeta) = \omega W_1(\omega X)(X - \zeta)$, and thus $\sigma_1(\omega X) - \tilde{\sigma}_2(X)$ is divisible by $(X - \zeta)$. Since ζ is sampled uniformly at random after $\tilde{\sigma}_1$ and $\tilde{\sigma}_2$ are defined, by the Swartz-Zippel Lemma we have that $\tilde{\sigma}_1(\omega X) = \tilde{\sigma}_2(X)$.

By the pairing equation in Item (iii) we have

$$\sigma_1(\omega X) - \sigma_1(X) = W_3(X) \nu_{\bar{\mathbb{H}}}(X)$$

and thus for any i, j with $j \in [1, d-1]$ we have $\sigma_1(\omega^{d\cdot i+j+1}) = \sigma_1(\omega^{d\cdot i+j})$. The latter implies that there exists a multi-set[19] of indexes $K = \{k_1, \ldots, k_n\}$ such that $\sigma_1(\omega^{d\cdot i+j}) = k_i$.

Finally, consider the vectors $\hat{\mathbf{t}}, \hat{\mathbf{f}}$ with elements in $\mathbb{F}^3$ such that $\hat{\mathbf{t}}_{d\cdot i+j} = (\mathbf{T}_{i,j}, \mathbf{C}^{(N)}_{i,j}, \mathbf{R}_{i,j})$ and $\hat{\mathbf{f}}_{d\cdot i+j} = (\mathbf{F}_{i,j}, \mathbf{C}^{(n)}_{i,j}, \sigma_1(\omega^{d\cdot i+j}))$. Notice that $\hat{\mathbf{f}} \prec \hat{\mathbf{t}}$ if and only if $\mathbf{T}_{|K} = \mathbf{F}$. Moreover, the Swartz-Zippel Lemma implies that the family of hash functions with signature $\mathbb{F}^3 \to \mathbb{F}$ that maps (x_0, x_1, x_2) to $\sum \rho^i x_i$ with key ρ sampled uniformly at random over $\mathbb{F}$ form a universal hash function family, namely for any $\mathbf{x}, \mathbf{x}'$ such that $\mathbf{x} \neq \mathbf{x}'$ $\Pr[h(\mathbf{x}) = h(\mathbf{x}')] = 1/q$. Notice that, the vectors $\hat{\mathbf{t}}$ and $\hat{\mathbf{f}}$ are fully defined before ρ is sampled. Thus by union bound over all the tuples of coordinates of $\hat{\mathbf{f}}$ and $\hat{\mathbf{t}}$ and by the universal hash property, we have that with $1 - \Omega(nd/q)$ probability $\bar{\mathbf{f}}^* \prec \bar{\mathbf{t}}^*$ implies that $\hat{\mathbf{f}} \prec \hat{\mathbf{t}}$.

D.1 Rows-Columns Matrix Lookup

In this section we consider a more general notion of the sub-matrix relation, to which we refer as (rows-columns) sub-matrix relation, where $\mathbf{F} \prec \mathbf{T}$ with $\mathbf{F} \in \mathbb{F}^{n\times d}$, $\mathbf{T} \in \mathbb{F}^{N\times D}$ and $N, D, n, d \in \mathbb{N}$ holds true if and only if there exist (multi)sets $R = \{r_1, \ldots, r_n\}$ and $C = \{c_1, \ldots, c_d\}$ with $\mathbf{F}_{i,j} = \mathbf{T}_{r_i,c_j}$ for any i, j. Similar to the notion of rows sub-matrix, we define $\mathbf{T}_{|R\times C}$ be the sub-matrix of $\mathbf{T}$ such that $(\mathbf{T}_{|R\times C})_{i,j} = \mathbf{T}_{r_i,c_j}$ for any i, j. We realize an argument system for the relation:

$$\hat{\mathcal{R}}'_{\mathsf{zklookup}} = \left\{ \mathsf{pp}; (N, D, n, d); \varepsilon; \mathbf{T}, \mathbf{F} : \begin{array}{c} \mathbf{F} \prec \mathbf{T} \\ |\mathbf{T}_j| = N \times D, |\mathbf{F}| = n \times d \end{array} \right\}$$

Notice that we are not zero-knowledge neither with respect of the number of columns nor the number of rows of the sub-matrix $\mathbf{F}$.

In Fig. 7 we describe our second scheme $\mathsf{mtx}^*[\mathsf{CP}]$ for rows-columns matrix lookup, namely a matrix lookup argument w.r.t. the more general relation described above, the that runs internally a lookup argument CP for KZG-based vector commitment scheme. In the description of the scheme, we let $\mathbb{K}$ (resp. $\mathbb{H}$) be a multiplicative subgroup of $\mathbb{F}$ of order $N \cdot D$ (resp. of order $n \cdot d$), we let $\omega := \omega_{n\cdot d}$ be the fixed generator for $\mathbb{H}$ and we consider the following matrices and polynomial:

1. the matrix $\mathbf{R} \in \mathbb{F}^{N\times d}$ where $R_{i,j} = i$,
2. for any k the matrix $\mathbf{C}^{(k)} \in \mathbb{F}^{k\times d}$ where $C_{i,j} = j$.
3. Let $\nu_R(X)$ be the vanishing polynomial of the set $\mathbb{H}_R = \{\omega^{d\cdot i+j} : j \in [1, d-1], i \in [n]\}$.
4. Let $\nu_{\mathbb{H}}(X)$ be the vanishing polynomial of the multiplicative subgroup $\mathbb{H}$.

Theorem 7. *The lookup argument* $\mathsf{mtx}^*[\mathsf{CP}]$ *defined in Fig. 7 is knowledge-sound in the AGM and ROM under the* $(N \cdot d, N \cdot d)$*-PDL assumption and assuming that* CP *is knowledge-sound. Furthermore, the protocol is zero-knowledge assuming* CP *is zero-knowledge.*

[19] K is a multiset because there might exist i, k such that $k_i = k_j$.

$\underline{\mathsf{Derive}(\mathsf{srs}, N, d, n)\textbf{:}}$

Let $\bar{\mathbf{t}}, \bar{\mathbf{f}}, \bar{\mathbf{r}}_N, \bar{\mathbf{c}}_N$ and $\bar{\mathbf{c}}_n$ be vectorizations of the matrices $\mathbf{T}, \mathbf{F}, \mathbf{R}, \mathbf{C}^{(N)}$ and $\mathbf{C}^{(n)}$.
Compute $\mathsf{c}_{r,N} \leftarrow \mathsf{Com}(\mathsf{ck}, \bar{\mathbf{r}}_N)$ and $\mathsf{c}_{c,N} \leftarrow \mathsf{Com}(\mathsf{ck}, \bar{\mathbf{c}}_N)$.
Compute $(\mathsf{ek}', \mathsf{vk}') \leftarrow \mathsf{CP.Derive}(\mathsf{srs}, Nd, nd)$.
Return $(\mathsf{ek}', \mathsf{vk}_n)$ where $\mathsf{vk}_n = (\mathsf{c}_{r,N}, \mathsf{c}_{c,N}, [\nu_R(s)]_2, \mathsf{vk}')$.

$\underline{\mathsf{Preproc}(\mathsf{srs}, \mathbf{T})\textbf{:}}$

Compute $(\mathsf{aux}_{T,j})_{j\in[Nd]} \leftarrow \mathsf{CP.Preproc}(\mathsf{srs}, \bar{\mathbf{t}})$,
$(\mathsf{aux}_{R,j})_{j\in[Nd]} \leftarrow \mathsf{CP.Preproc}(\mathsf{srs}, \bar{\mathbf{r}}_N)$,
$(\mathsf{aux}_{C,j})_{j\in[Nd]} \leftarrow \mathsf{CP.Preproc}(\mathsf{srs}, \bar{\mathbf{c}}_N)$.
Let $\mathsf{aux}_i = (\mathsf{aux}_{T,di+j}, \mathsf{aux}_{R,di+j}, \mathsf{aux}_{C,di+j})_{j\in[d]}$.
Return $(\mathsf{aux}_i)_{i\in[N]}$.

$\underline{\mathsf{Prove}(\mathsf{ek}, (\mathsf{c}_\mathbf{T}, \mathsf{c}_\mathbf{F}), \mathbf{F}, (\mathsf{aux}_j)_{j\in K})\textbf{:}}$ //$\mathbf{T}_{|R\times C} = \mathbf{F}$, $R = \{r_1, \ldots, r_n\}$, $C = \{c_1, \ldots, c_d\}$.

Let $\mathbf{S}^R$ be s.t. $\mathbf{S}^R_{i,j} = r_i$ for $i\in[n], j\in[d]$. Let $\mathbf{S}^C$ be s.t $\mathbf{S}^C_{i,j} = c_j$ for $i\in[n], j\in[d]$.
Let $\sigma^R(X)$ (resp. $\sigma^C(X)$) be a randomized LDE over $\mathbb{H}$ of the vectorization of $\mathbf{S}^R$ (resp. $\mathbf{S}^C$).
Compute $w^R(X)$ such that $\sigma^R(\omega \cdot X) - \sigma^R(X) = w^R(X) \cdot \nu_R(X)$.
Compute $w^C(X)$ such that $\sigma^C(\omega^d \cdot X) - \sigma^C(X) = w^C(X) \cdot \nu_{\mathbb{H}}(X)$.
$(\rho, \zeta) \leftarrow \mathsf{RO}(\mathsf{vk}_n \| (\mathsf{c}_\mathbf{T}, \mathsf{c}_\mathbf{F}) \| (\mathsf{c}_{R,n}, \mathsf{c}_{R',n}, \mathsf{c}_w))$. //Fiat-Shamir challenge.
Compute $z^R \leftarrow \sigma(\omega \cdot \zeta), z^C \leftarrow \sigma(\omega^d \cdot \zeta)$.
Compute proofs π_R and $\pi_{R'}$ for $\hat{\mathcal{R}}_{\mathsf{eval}}(\omega \cdot \zeta, z^R; \sigma^R(X)) = 1$ and $\hat{\mathcal{R}}_{\mathsf{eval}}(\zeta, z^R; \sigma(\omega \cdot X)) = 1$;
Compute proofs π_C and $\pi_{C'}$ for $\hat{\mathcal{R}}_{\mathsf{eval}}(\omega^d \cdot \zeta, z^C; \sigma^C(X)) = 1$ and $\hat{\mathcal{R}}_{\mathsf{eval}}(\zeta, z^C; \sigma(\omega^d \cdot X)) = 1$;
Let π^* proof for $\hat{\mathcal{R}}_{\mathsf{zklookup}}((N \cdot d, n \cdot d); \varepsilon; (\bar{\mathbf{t}}^*, \bar{\mathbf{f}}^*)) = 1$ where

$$\bar{\mathbf{t}}^* = \bar{\mathbf{t}} + \rho \cdot \bar{\mathbf{c}}_N + \rho^2 \cdot \bar{\mathbf{r}}_N \qquad \bar{\mathbf{f}}^* = \bar{\mathbf{f}} + \rho \cdot \bar{\boldsymbol{\sigma}}^C + \rho^2 \cdot \bar{\boldsymbol{\sigma}}^R \tag{29}$$

Return $\pi = ([\sigma(s)]_1, [\sigma(\omega \cdot s)]_1, [w(s)]_1, \pi_R, \pi_{R'}, \pi^*, z)$.

$\underline{\mathsf{Verify}(\mathsf{vk}_n, (\mathsf{c}_\mathbf{T}, \mathsf{c}_\mathbf{F}), \pi)\textbf{:}}$

Parse the proof $\pi = (\mathsf{c}_{R,n}, \mathsf{c}_{R',n}, \mathsf{c}_w, \pi_R, \pi_{R'}, \pi^*, z)$.
$(\rho, \zeta) \leftarrow \mathsf{RO}(\mathsf{vk}_n \| (\mathsf{c}_\mathbf{T}, \mathsf{c}_\mathbf{F}) \| (\mathsf{c}_{R,n}, \mathsf{c}_{R',n}, \mathsf{c}_w))$. //Fiat-Shamir challenge.
Compute $\mathsf{c}^*_\mathbf{T} \leftarrow \mathsf{c}_\mathbf{T} + \rho \mathsf{c}_{c,N} + \rho^2 \mathsf{c}_{r,N}$ and $\mathsf{c}^*_\mathbf{F} \leftarrow \mathsf{c}_\mathbf{F} + \rho \mathsf{c}_{c,n} + \rho^2 \mathsf{c}_{R,n}$.
Return 1 if the following checks hold (else 0):

(i) $\mathsf{Verify}_{\mathsf{eval}}(\mathsf{ck}, (\mathsf{c}_{R,n}, \omega \cdot \zeta, z)) = 1$,
(ii) $\mathsf{Verify}_{\mathsf{eval}}(\mathsf{ck}, (\mathsf{c}_{R',n}, \zeta, z)) = 1$,
(iii) $e(\mathsf{c}_{R',n} - \mathsf{c}_{R,n}, [1]_2) = e(\mathsf{c}_w, [\nu_{\bar{\mathbb{k}}}(s)]_2)$,
(iv) $\mathsf{CP.Verify}(\mathsf{srs}, \mathsf{vk}', (\mathsf{c}^*_\mathbf{T}, \mathsf{c}^*_\mathbf{F}), \pi^*) = 1$.

Fig. 7. Our rows-columns Matrix Lookup Argument $\mathsf{mtx}^*[\mathsf{CP}]$.

Procedure $\mathsf{T}(x)$:

$n \leftarrow 1$ // root node
while n is not a leaf **do**
  fetch $\mathbf{E}_n, \mathbf{T}_n$
  if $\forall j \in [d] : \mathbf{E}_{n,j} = 1 \Rightarrow x_j < \mathbf{T}_{n,j}$ **then** $n \leftarrow$ left child of n
  elseif $\forall j \in [d] : \mathbf{E}_{n,j} = 1 \Rightarrow x_j \geq \mathbf{T}_{n,j}$ **then** $n \leftarrow$ right child of n
  else return $\perp$
fetch v_n **from** $\mathbf{v}$ // Vector of all the labels
return v_n

Fig. 8. The pseudo-code of an evaluation of a quasi-complete decision tree.

The proof of the theorem is very similar to the proof of Theorem 2, the only difference is that the prover additionally shows that $\sigma^C(X)$ has a well-defined tensor-product structure that we prove in the next lemma.

Lemma 5. *Let $\mathbb{H} = \langle\omega\rangle$ be a multiplicative subgroup of $\mathbb{F}$ of order $n \cdot d$ with $d \geq 1$, there exists $\mathbf{c}$ of length d such that $\sigma^C(X)$ is the LDE of $\mathbf{c} \otimes \mathbf{1}$ (over $\mathbb{H}$) if and only if $\sigma^C(\omega^d \cdot X) - \sigma^C(X) \equiv 0 \mod \nu_{\mathbb{H}}(X)$.*

Proof. The first implication is easy. In fact, let $\sigma^C(X)$ be the LDE of $\mathbf{c} \otimes \mathbf{1}$ then $\sigma^C(\omega^{(i+1)\cdot d+j \mod nd}) = \sigma^C(\omega^{i\cdot d+j \mod nd})$ for any i and $j \in [d]$. We can prove the other direction by induction.

Let $\nu_i(X)$ be the vanishing polynomial of the set $\mathbb{H}_i = \{\omega^j : 0 \leq j < i \cdot d\}$, we can show that if $\bar{\sigma}(X) := \sigma^C(\omega^d \cdot X) - \sigma^C(X) \equiv 0 \mod \nu_i(X)$ and $i \leq n$ then $\exists \mathbf{c}, \mathbf{d}$ such that $\omega^C(X)$ is the LDE of $(\mathbf{c} \otimes \mathbf{1}_i \| \mathbf{d})$ and $\mathbf{1}_i$ has length i.

- For $i = 1$ the statement is trivially true because $\forall j \in [d] : \sigma^C(\omega^{d+j}) = \sigma^C(\omega^j)$ thus there exists $\mathbf{c}, \mathbf{d}$ such that σ^C is the LDE of $\mathbf{c} \| \mathbf{c} \| \mathbf{d}$.
- For $n \geq i > 1$ we have that $\bar{\sigma}(X) \equiv 0 \mod \nu_i(X)$ implies $\bar{\sigma}(X) \equiv 0 \mod \nu_{i-1}(X)$, thus σ^C is the LDE of $\mathbf{c} \otimes \mathbf{1}_{i-1} \| \mathbf{d}$. We need to show that $\mathbf{d} = \mathbf{c} \| \mathbf{d}'$ for some vector $\mathbf{d}'$. Notice that $\sigma^C(\omega^d \cdot \omega^{(i-1)\cdot d+j}) - \sigma^C(\omega^{(i-1)\cdot d+j}) = 0$ for $j \in [d]$ thus, if $i < n$, the first d coordinates of $\mathbf{d}$ agree with the last d coordinate of $\mathbf{c} \otimes \mathbf{1}_{i-1}$, which means that $\mathbf{d} = \mathbf{c} \| \mathbf{d}'$ for some $\mathbf{d}'$, if $i = n$ the first d coordinates of $\mathbf{d}$ agree with the first d coordinates of $\mathbf{c} \otimes \mathbf{1}_{n-1}$, which means that $\mathbf{d} = \mathbf{c}$.

E Additional Material on Section 6

E.1 The Extended Encoding of Decision Trees

Lemma 3. *Let T be a quasi-complete decision tree with N_{tot} nodes and $(\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow})$ be a boxes-encoding of T. Let $\mathbf{v}$ be the vector of the labels assigned to the leaf nodes of T, namely for any $i \in [N_{\mathsf{int}} + 1, N_{\mathsf{tot}}]$, we have v_i as the label assigned to the i-th leaf. For any $\mathbf{x} \in [B]^d$, $\mathsf{T}(\mathbf{x}) = v_{k(\mathbf{x})}$ or $\mathsf{T}(\mathbf{x}) = \perp$.*

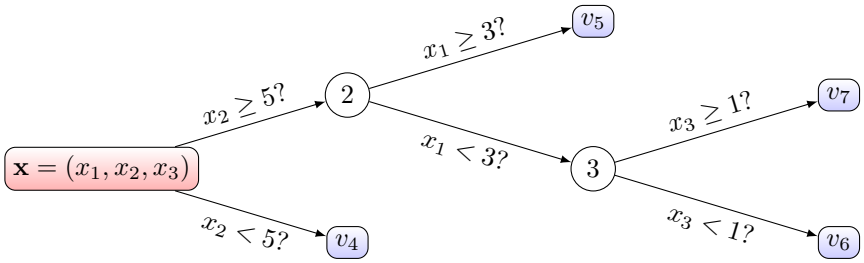

Fig. 9. Example of decision tree ($d = 3, N_{\mathsf{int}} = 3, N_{\mathsf{tot}} = 7$). $\mathbf{x}$ is the input to the tree. Internal nodes are marked by their (circled) index only, otherwise the subscript in the labels (v_i-s) marks their index. The root is implicitly indexed as node 1.

Proof. Let $n_1, \ldots, n_s$ be the nodes traversed by the computation of $\mathsf{T}(\mathbf{x})$, we prove that $\mathbf{x}$ is contained in $(\mathbf{N}^{\leftarrow}_{n_j}, \mathbf{N}^{\rightarrow}_{n_j})$ for any j or $T(\mathbf{x}) = \perp$. Notice that n_1 is the root, namely $n_1 = 1$, and by (1) of Definition 8 we clearly have $\mathbf{x}$ is contained in $(\mathbf{N}^{\leftarrow}_1, \mathbf{N}^{\rightarrow}_1)$. Moreover, assume that at the i-th step, $\mathbf{x}$ is contained in $(\mathbf{N}^{\leftarrow}_{n_i}, \mathbf{N}^{\rightarrow}_{n_i})$. If $\forall j \in [d] : \mathbf{E}_{n_i,j} = 1 \Rightarrow x_j < \mathbf{T}_{n_i,j}$ then n_{i+1} is the left child of n_i and because of Eqs. (14) and (15) we have that $\mathbf{x}$ is contained in $(\mathbf{N}^{\leftarrow}_{n_{i+1}}, \mathbf{N}^{\rightarrow}_{n_{i+1}})$. Similarly, if $\forall j \in [d] : \mathbf{E}_{n_i,j} = 1 \Rightarrow x_e \geq \mathbf{T}_{n_i,j}$, then n_{i+1} is the right child of n_i and because of Eqs. (14) and (16) we have that $\mathbf{x}$ is contained in $(\mathbf{N}^{\leftarrow}_{n_{i+1}}, \mathbf{N}^{\rightarrow}_{n_{i+1}})$. Otherwise, we have $\mathsf{T}(\mathbf{x}) = \perp$.

Because of Eqs. (15) and (16) the boxes of the left and right children are disjoint (namely, there isn't any $\mathbf{x}$ that is contained in both boxes). Thus by induction on the structure of the tree, the set of the boxes of all the leaves are pair-wise disjointed. This implies that if $\mathsf{T}(\mathbf{x}) \neq \perp$ then $k(\mathbf{x})$ is uniquely defined. Thus n_s is equal to $k(\mathbf{x})$. □

Lemma 4. *Consider a tuple* $(\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}, \mathbf{L}, \mathbf{R}, \mathbf{E}, \mathbf{v})$ *such that the following constraints hold:*

a) The following equations hold:

$$\mathbf{N}^{\leftarrow}_1 = \mathbf{0}, \mathbf{N}^{\rightarrow}_1 = \mathbf{B} + \mathbf{1}, \tag{17}$$

$$\mathbf{L} \cdot \mathbf{N}^{\leftarrow} = \mathbf{P}^{\leftarrow}, \mathbf{R} \cdot \mathbf{N}^{\rightarrow} = \mathbf{P}^{\rightarrow}, \tag{18}$$

$$\mathbf{E} \circ (\mathbf{L} \cdot \mathbf{N}^{\rightarrow} - \mathbf{R} \cdot \mathbf{N}^{\leftarrow}) = \mathbf{0} \tag{19}$$

$$(\mathbf{1} - \mathbf{E}) \circ (\mathbf{P}^{\leftarrow} - \mathbf{R} \cdot \mathbf{N}^{\leftarrow}) = \mathbf{0}, \qquad (\mathbf{1} - \mathbf{E}) \circ (\mathbf{P}^{\rightarrow} - \mathbf{L} \cdot \mathbf{N}^{\rightarrow}) = \mathbf{0} \tag{20}$$

b) All the boxes are not empty. Namely, for all i, j we have $\mathbf{N}^{\leftarrow}_{i,j} < \mathbf{N}^{\rightarrow}_{i,j}$.

c) The matrix $\begin{pmatrix} \mathbf{L} \\ \mathbf{R} \end{pmatrix}$ is a (row) permutation of the (squared) matrix $(\mathbf{0} \| \mathbf{I}_{N_{\mathsf{tot}}-1})$ (the matrix whose rows are the row vectors $(\mathbf{e}_i)_{i \in [2, N_{\mathsf{tot}}]}$ of length N_{tot}).

Then there exists a quasi-complete decision tree T with N_{tot} nodes such that $\mathsf{Encode}(\mathsf{T}) = (\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}, \mathbf{L}, \mathbf{R}, \mathbf{E}, \mathbf{v})$.

$$\mathbf{L} = \overbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}}^{N_{\mathsf{tot}}} \Big\} N_{\mathsf{int}} \quad \mathbf{R} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{N}^{\leftarrow} = \overbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 0 \\ 3 & 5 & 0 \\ 0 & 5 & 0 \\ 0 & 5 & 1 \end{bmatrix}}^{d} \Big\} N_{\mathsf{tot}} \quad \mathbf{N}^{\rightarrow} = \begin{bmatrix} B+1 & B+1 & B+1 \\ B+1 & B+1 & B+1 \\ 3 & B+1 & B+1 \\ B+1 & 5 & B+1 \\ B+1 & B+1 & B+1 \\ 3 & B+1 & 1 \\ 3 & B+1 & B+1 \end{bmatrix} \quad \mathbf{E} = \overbrace{\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{d} \Big\} N_{\mathsf{int}}$$

Fig. 10. Matrix examples for decision tree in Fig. 9. Notice that the boundaries in $\mathbf{N}^{\leftarrow}$ and $\mathbf{N}^{\rightarrow}$ consist of a half-open interval that is greater or equal to $\mathbf{N}^{\leftarrow}$ and less than $\mathbf{N}^{\rightarrow}$ (e.g. $0 \leq x_1 < 3$ for node 3). Notice how the Hadamard product of a row of $\mathbf{1} - \mathbf{E}$ can be used to show two sibling boxes are the same except in one coordinate, e.g. for nodes 6 and 7 we use the third row of $\mathbf{1} - \mathbf{E}$ (node 3 is their parent) to show that it should hold $(\mathbf{1} - \mathbf{E})_3 \circ (\mathbf{N}^{\leftarrow}{}_6 - \mathbf{N}^{\leftarrow}{}_7) = \mathbf{0}$ and $(\mathbf{1} - \mathbf{E})_3 \circ (\mathbf{N}^{\rightarrow}{}_6 - \mathbf{N}^{\rightarrow}{}_7) = \mathbf{0}$. Also, for the split coordinate, the right bound of the left child should equal the left bound of the right child and equal the threshold of their parent. For example, for node 2, the right bound of the left child (node 3) for x_1 is $(\mathbf{E}_2 \circ (\mathbf{L} \cdot \mathbf{N}^{\rightarrow})_2)_1 = 3$. Similarly, for the right child, we have $(\mathbf{E}_2 \circ (\mathbf{R} \cdot \mathbf{N}^{\leftarrow})_2)_1 = 3$. We have the threshold for node 2 and x_1 is $\mathbf{T}_{2,1} = 3$.

Proof. First, notice that the constraint in Eq. (17) is necessary by definition of box-encoding of a tree.

From Item c), for any $p \in [N_{\mathsf{int}}]$, $\mathbf{L}_p$ and $\mathbf{R}_p$ are elementary vectors, so there exists unique l, r such that $\mathbf{L}_{i,l} = 1, \mathbf{R}_{i,r} = 1$. Since $N_{\mathsf{tot}} - 1 = 2N_{\mathsf{int}}$ and $\mathcal{L} \cup \mathcal{R} = [2, N_{\mathsf{tot}}]$, the rows of the matrix $\frac{\mathbf{L}}{\mathbf{R}}$ are linearly independent. As a result, all internal nodes have one left child and one right child.

We define a procedure that, upon the input data structure $(\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}, \mathbf{L}, \mathbf{R}, \mathbf{E}, \mathbf{v})$ such that the hypothesis of the lemma holds, computes an (alleged) quasi-complete decision tree. We then show that the latter is indeed a quasi-complete decision tree, namely, that (I) the resulting (indirect) graph is acyclic and the number of edges in the graph is $N_{\mathsf{tot}} - 1$ (thus, it is a tree), moreover, the out-degree of any of the nodes is either 2 or 0 (thus it is a binary tree), and (II) for each internal node p, the procedure defines the feature vector $\mathbf{E}_p \in \{0, 1\}^d$ and threshold vector $\mathbf{T}_p \in [B]^d$.

– Start with a fully disconnected graph with N_{tot} nodes. For any $p \in [N_{\mathsf{int}}]$, let l, r be the unique indexes such that $\mathbf{L}_{p,l} = 1$ and $\mathbf{R}_{p,r} = 1$. Add the direct edge (p, l) and (p, r) to the direct graph.
– Set $\mathbf{T} \leftarrow \mathbf{E} \circ (\mathbf{L} \cdot \mathbf{N}^{\rightarrow})$. For any p, the p-th row of $\mathbf{T}$ and $\mathbf{E}$ are the feature and threshold vectors for p.
– Associate to the leaves the labels $\mathbf{v}$. Namely, for any $i > N_{\mathsf{int}}$, the i-th node gets assigned the label v_i.

We notice that for any p, $\mathbf{E}_p, \mathbf{T}_p$ are well defined. In fact

$$\mathbf{E}_p \circ (\mathbf{N}^{\rightarrow}_l - \mathbf{T}_p) = \mathbf{E}_p \circ (\mathbf{N}^{\rightarrow}_l - (\mathbf{L} \cdot \mathbf{N}^{\rightarrow})_p) = \mathbf{0}$$

where the first equation comes by definition of $\mathbf{T}_p$ and the second by definition of $\mathbf{L}$. Notice that, by hypothesis of the lemma, we have $\mathbf{T}_p = \mathbf{R} \cdot \mathbf{N}^{\leftarrow}$. Thus, with the same derivation as above, we have $\mathbf{E}_p \circ (\mathbf{N}^{\leftarrow}_r - \mathbf{T}_p) = 0$. By definition of $\mathbf{L}$ (resp. $\mathbf{R}$) and Eq. (20) we readily derive that Eq. (16) holds for $\mathbf{E}$. Thus we have then proved condition (II).

We can focus on proving condition (I). First notice, it is easy to check that the out-degree of any of the nodes is at most 2, by definition of the procedure described above. Notice that the number of edges added by the procedure is the added number of rows in $\mathbf{L}$ and $\mathbf{R}$, namely $N_{\mathsf{tot}} - 1 = 2N_{\mathsf{int}}$. Thus we need to prove that the procedure did not add twice the same edge. This could only happen if there is a node $p \in [N_{\mathsf{int}}]$ such that for the same child node i, $\mathbf{L}_{p,i} = \mathbf{R}_{p,i} = 1$. However, we have proved that all internal nodes have one left and one right child. Given that the number of elements in $(\mathbf{L}, \mathbf{R})$ is $N_{\mathsf{tot}} - 1$, we have that any child node i can not serve as a child node more than one time. In other words, for any child node i, we have the only $p \in [N_{\mathsf{int}}]$ such that $\mathbf{L}_{p,i} = 1$ **or** $\mathbf{R}_{p,i} = 1$.

Define $P_i := \sum_j |\mathbf{N}^{\rightarrow}_{i,j} - \mathbf{N}^{\leftarrow}_{i,j}|$ as the "potential" associated to the box $(\mathbf{N}^{\leftarrow}_i, \mathbf{N}^{\rightarrow}_i)$ over the integers. For any $p \in [N_{\mathsf{int}}]$ with $\mathbf{L}_{p,l} = 1$ and $\mathbf{R}_{p,r} = 1$, we have $P_p > P_l$ and $P_p > P_r$. For the former, namely $P_p > P_l$, notice that because of Eqs. (18) to (20) and Item b), for any e_i with all $\mathbf{E}_{p,e_i} = 1$, we have it holds that $\mathbf{N}^{\leftarrow}_{p,e_i} = \mathbf{N}^{\leftarrow}_{l,e_i} < \mathbf{N}^{\rightarrow}_{l,e_i} < \mathbf{N}^{\rightarrow}_{r,e_i} = \mathbf{N}^{\rightarrow}_{p,e_i}$, and thus $\mathbf{N}^{\rightarrow}_{p,e_i} - \mathbf{N}^{\leftarrow}_{p,e_i} > \mathbf{N}^{\rightarrow}_{l,e_i} - \mathbf{N}^{\leftarrow}_{l,e_i}$ while, $\mathbf{N}^{\rightarrow}_{p,e_j} - \mathbf{N}^{\leftarrow}_{p,e_j} = \mathbf{N}^{\rightarrow}_{l,e_j} - \mathbf{N}^{\leftarrow}_{l,e_j}$ for the other indexes e_j with all $\mathbf{E}_{p,e_j} = 0$. The latter, namely that $P_p > P_r$, follows similarly. Assume there exists a cycle in (the indirect generalization of) the graph produced by the procedure described above, and let $(j_1, \ldots, j_k = j_1)$ be such a cycle. First, notice that because of the constraint in Item c), every node has an in-degree at most 1 in the direct graph. Thus, if a cycle exists in the indirect graph, there is a cycle in the direct graph as well. Moreover, by construction, all the edges in the graph are of the form (p, l) or (p, r) for $p \in [N_{\mathsf{int}}]$. We have for any k that $P_{j_k} < P_{j_{k+1}}$ and thus $P_{j_1} < P_{j_1}$ which reaches a contradiction. Thus there are no cycles in the graph.

E.2 Extractable Commitment to Decision Trees

Theorem 3. *The commitment scheme CS_{DT} defined in Fig. 3 is hiding and it is an extractable commitment scheme for the domain $\{\mathcal{T}^*_{N_{\mathsf{tot}},B,d}\}_{N_{\mathsf{tot}},d,B}$ in the AGM and assuming the building blocks are knowledge-sound and zero-knowledge.*

Proof (Sketch.). We can prove hiding by relying on the hiding of the matrix commitment scheme and the zero-knowledge of the underlying CP-SNARKs[20].

By definition of extractable commitment we can interpret the commitment function as a first sub-procedure that generates a (binding) commitment and second procedure that generates a proof. By inspection of the algorithm, we can divide the commitment function above in this way. We need to prove knowledge soundness in the AGM for the derived CP-SNARK.

The extractor, using the algebraic representations, extract polynomials from the commitments $\mathsf{c}_{\leftarrow}, \mathsf{c}_{\rightarrow}, \mathsf{c}_v$, the commitments $\mathsf{c}'_{\leftarrow}, \mathsf{c}'_{\rightarrow}$, the sparse commitments $\mathsf{c}_L, \mathsf{c}_R$ and the commitment c_E. From such polynomials, the extractor can derive matrices $(\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}, \mathbf{L}, \mathbf{R}, \mathbf{E}, \mathbf{v})$. In particular, the matrix $\mathbf{N}^{\leftarrow}$ is defined as the sum of the (padded) matrices $\mathbf{F}^{\leftarrow}$ extracted from $\mathsf{c}_{\leftarrow}$ and $\mathbf{P}^{\leftarrow}$ extracted from $\mathsf{c}'_{\leftarrow}$ (similarly for $\mathbf{N}^{\rightarrow}$). We show that the constraints of Lemma 4 hold, thus the extractor can compute a valid quasi-complete decision tree T from the extracted extended encoding.

The validity of the proofs $\pi_{11}, \ldots, \pi_{14}$ enforce that $\mathbf{N}^{\leftarrow}$ (resp. $\mathbf{N}^{\rightarrow}$) stacks the matrix $\mathbf{P}^{\leftarrow}$ on to of $\mathbf{F}^{\leftarrow}$ (resp. $\mathbf{P}^{\rightarrow}$ on top of $\mathbf{F}^{\rightarrow}$). Moreover, the validity of the proofs π_{15}, π_{16} enforce the constraint Eq. (17), as otherwise we would either break the knowledge soundness of $\mathsf{CP}_{\mathsf{sm}}$ or the binding property of the matrix commitment scheme.

The validity of the proofs π_1 and π_3 implies that Eq. (18) indeed holds, as otherwise we would either break the knowledge soundness of $\mathsf{CP}_{\mathsf{lin}}$ or the binding property of the matrix commitment scheme.

The validity of the proof π_2 (resp. π_4) implies that the commitments c_{ln} (resp. c_{rn}) open to the matrix $\bar{\mathbf{L}} \cdot \mathbf{N}^{\rightarrow}$ (resp. $\bar{\mathbf{R}} \cdot \mathbf{N}^{\leftarrow}$), this coupled with the validity of the proof π_5 imply the constraints in Eqs. (19) and (20). Again, we can formally prove this by a first reduction to the biding property of the matrix commitment (showing that the knowledge extractors for π_2, π_4 and π_5 should output the same matrices as computed by the algebraic representations), and then to the knowledge soundness of $\mathsf{CP}_{\mathsf{lin}}$ and $\mathsf{CP}_{\mathsf{had}}$.

The validity of the proof π_8 implies Item b) in a straightforward manner. The validity of the proofs π_9 and π_{10} and the definition of the polynomial $id(X)$ by the KGen algorithm imply Item c). In particular set $c'(X) \stackrel{\text{def}}{=} col_{\bar{\mathbf{L}}}(X) + col_{\bar{\mathbf{R}}}(X)$, we have $c'(h^i) = col_{\bar{\mathbf{L}}}(h^i)$ and $c'(h^{i+N_{\mathsf{int}}}) = col_{\bar{\mathbf{R}}}(h^i)$ for $i \in [N_{\mathsf{int}}]$. Moreover, $c'(X)$ is a permutation of $i(X)$ which, by definition, represents a sparse commitment of the matrix whose rows are the elementary vectors $(\mathbf{e}_{j+1})_{j \in [N_{\mathsf{tot}}-1]}$.

Proof. We can prove hiding by relying on the hiding of the matrix commitment scheme and the zero-knowledge of the underlying CP-SNARKs[21].

[20] Notice that, using higher degrees for the randomizers of the commitments, hiding would still hold even if the proofs leaked evaluation points (see the notion of leaky-zero-knowledge from [7]) from the commitments in π

[21] Notice that, using higher degrees for the randomizers of the commitments, hiding would still hold even if the proofs leaked evaluation points (see the notion of leaky-zero-knowledge from [7]) from the commitments in π

By definition of extractable commitment we can interpret the commitment function as a first sub-procedure that generates a (binding) commitment and second procedure that generates a proof. By inspection of the algorithm, we can divide the commitment function above in this way. We prove knowledge soundness in the AGM for the derived CP-SNARK.

The extractor $\mathcal{E}_{\mathsf{DT}}$, using the algebraic representations, extract polynomials from the commitments $\mathsf{c}_{\leftarrow}, \mathsf{c}_{\rightarrow}, \mathsf{c}_v$, the commitments $\mathsf{c}'_{\leftarrow}, \mathsf{c}'_{\rightarrow}$, the sparse commitments $\mathsf{c}_L, \mathsf{c}_R$ and the commitment c_E. From such polynomials, the extractor can derive matrices $(\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}, \mathbf{L}, \mathbf{R}, \mathbf{E}, \mathbf{v})$. In particular, the matrix $\mathbf{N}^{\leftarrow}$ is defined as the sum of the matrices $\bar{\mathbf{F}}_{\leftarrow}$ extracted from $\mathsf{c}_{\leftarrow}$ and $\bar{\mathbf{P}}_{\leftarrow}$ extracted from $\mathsf{c}'_{\leftarrow}$ (similarly for $\mathbf{N}^{\rightarrow}$). We show that the constraints of Lemma 4 hold, thus the extractor can compute a valid quasi-complete decision tree T from the extracted extended encoding. We proceed with a sequence of hybrids where $\mathbf{H}_0$ is the extractability game for CS_{DT} that returns 1 if the verifier accepts the proof and the extractor fails to produce a valid witness. In the next hybrids we sometimes reduce to the (N_1, N_2)-PDL assumption (see Definition 12).

Hybrid $\mathbf{H}_1$. The Hybrid $\mathbf{H}_1$ is the same as $\mathbf{H}_0$ but it additionally runs the extractor w.r.t. instances $([N_{\mathsf{int}}], \mathsf{c}_{\leftarrow})$, $([N_{\mathsf{int}}], \mathsf{c}_{\rightarrow})$, $((N_{\mathsf{int}}, N_{\mathsf{tot}}], \mathsf{c}'_{\leftarrow})$, $((N_{\mathsf{int}}, N_{\mathsf{tot}}], \mathsf{c}'_{\rightarrow})$ let $\tilde{\mathbf{F}}^{\leftarrow}, \tilde{\mathbf{F}}^{\rightarrow}, \tilde{\mathbf{P}}^{\leftarrow}$ and $\tilde{\mathbf{P}}^{\rightarrow}$ be the extracted matrices the hybrid returns 0 if either $\tilde{\mathbf{F}}^{\leftarrow}$ or $\tilde{\mathbf{F}}^{\rightarrow}$ have the first N_{int} rows set to $\mathbf{0}$ or $\tilde{\mathbf{P}}^{\leftarrow}$ or $\tilde{\mathbf{P}}^{\rightarrow}$ have the last $N_{\mathsf{tot}} - N_{\mathsf{int}}$ rows set to $\mathbf{0}$ or the matrices are not the valid opening for the respective commitments $\mathsf{c}_{\leftarrow}, \mathsf{c}_{\rightarrow}, \mathsf{c}'_{\leftarrow}$ and $\mathsf{c}'_{\rightarrow}$. It is easy to see that $\Pr[\mathbf{H}_0 = 1] \leq \Pr[\mathbf{H}_1 = 1] = \mathsf{negl}(\lambda)$ where the negligible factor comes from the knowledge soundness of $\mathsf{CP}_{\mathsf{sm}}$.

Hybrid $\mathbf{H}_2$. The Hybrid $\mathbf{H}_2$ is the same as $\mathbf{H}_1$ but it additionally returns 0 if $\tilde{\mathbf{F}}^{\leftarrow} \neq \mathbf{F}^{\leftarrow}$ or $\tilde{\mathbf{F}}^{\rightarrow} \neq \mathbf{F}^{\rightarrow}$ or $\tilde{\mathbf{P}}^{\leftarrow} \neq \mathbf{P}^{\leftarrow}$ or $\tilde{\mathbf{P}}^{\rightarrow} \neq \mathbf{P}^{\rightarrow}$. Let E_1 be the event that one of previous dis-equations holds true. We have that $\Pr[\mathbf{H}_0 = 1] \leq \Pr[\mathbf{H}_1 = 1] + \Pr[E_1]$. Notice the event E_1 implies an attacker against the binding property of the matrix commitment scheme, thus $\Pr[E_1] = \mathsf{negl}(\lambda)$.

In the next hybrids we can iteratively use the same proof strategy of the previous hybrids by first relying on the knowledge soundness of one of the CP-SNARKs and then on the binding property of the commitment scheme. Thus from now on we implicitly assume that the matrices extracted by the extractors of the CP-SNARKs match the matrices extracted by the extractor $\mathcal{E}_{\mathsf{DT}}$.

Hybrid $\mathbf{H}_3$. The Hybrid $\mathbf{H}_3$ is the same as $\mathbf{H}_2$ but it additionally returns 0 if $\mathbf{N}^{\leftarrow}_1 \neq \mathbf{0}$ or $\mathbf{N}^{\rightarrow}_1 \neq (B+1, \ldots, B+1)$. By the knowledge soundness of $\mathsf{CP}_{\mathsf{sm}}$ we have that both $\bar{\mathbf{F}}_{\leftarrow 1}$ and $\bar{\mathbf{P}}_{\leftarrow 1}$ equal the row vector $\mathbf{0}$, thus $\mathbf{N}^{\leftarrow}_1 = \mathbf{0}$. Similarly for $\mathbf{N}^{\rightarrow}_1$ however here we notice that we prove that $(\mathbf{N}^{\rightarrow} + \mathbf{B})_1$ equals to $\mathbf{0}$ and $\bar{\mathbf{F}}_{\rightarrow 1}$ equals to $\mathbf{0}$ and thus $\mathbf{N}^{\rightarrow}_1$ is the row vector $(B+1, \ldots, B+1)$.

Notice that when $\mathbf{H}_3 = 1$ the constraint Eq. (17) of Lemma 4 holds.

Hybrid $\mathbf{H}_4$. The Hybrid $\mathbf{H}_4$ is the same as $\mathbf{H}_3$, but it additionally (runs the extractor w.r.t. instance-proof tuple $(\mathsf{c}_L, \mathsf{c}_{N,\leftarrow}, \mathsf{c}'_{\leftarrow}), \pi_1$, such extractor exists because of the knowledge soundness of $\mathsf{CP}_{\mathsf{lin}}$) and returns 0 if $\mathbf{L} \cdot \mathbf{N}^{\leftarrow} \neq \mathbf{P}^{\leftarrow}$. The distinghuishing event between $\mathbf{H}_3$ and $\mathbf{H}_4$ is the event the extractor fails to extract

a valid witness, thus if $\mathsf{CP}_{\mathsf{lin}}$ is knowledge-sound (and the matrix commitment is binding) then $\Pr[\mathbf{H}_3] \leq \Pr[\mathbf{H}_4] + \mathsf{negl}(\lambda)$.

Hybrid $\mathbf{H}_5$. The hybrid $\mathbf{H}_5$ is the same as $\mathbf{H}_4$, but it additionally (runs the extractor w.r.t. instance-proof tuple $(\mathsf{c}_R, \mathsf{c}_{N,\rightarrow}, \mathsf{c}'_{\rightarrow}), \pi_3)$ returns 0 if $\mathbf{R} \cdot \mathbf{N}^{\rightarrow} \neq \mathbf{P}^{\rightarrow}$. Similarly to the previous hybrids, this hybrid is negligibly close to $\mathbf{H}_4$ because of the knowledge-soundness of $\mathsf{CP}_{\mathsf{lin}}$ (and the binding property of the commitment scheme).

Notice that when $\mathbf{H}_5 = 1$ the constraint Eq. (18) of Lemma 4 holds.

Hybrid $\mathbf{H}_6$. The hybrid $\mathbf{H}_6$ is the same as $\mathbf{H}_5$, but it additionally returns 0 if $\bar{\mathbf{L}} \cdot \mathbf{N}^{\rightarrow}$ is not a valid opening for c_{ln} or $\bar{\mathbf{R}} \cdot \mathbf{N}^{\leftarrow}$ is not a valid opening for c_{rn}. We can prove $\Pr[\mathbf{H}_5 = 1] \leq \Pr[\mathbf{H}_6 = 1] + \mathsf{negl}(\lambda)$ based on the knowledge soundness of $\mathsf{CP}_{\mathsf{lin}}$.

Hybrid $\mathbf{H}_7$. The hybrid $\mathbf{H}_7$ is the same as $\mathbf{H}_6$, but it additionally returns 0 if

$$\bar{\mathbf{E}} \cdot (\bar{\mathbf{L}} \cdot \mathbf{N}^{\rightarrow} - \bar{\mathbf{R}} \cdot \mathbf{N}^{\leftarrow}) \neq \mathbf{0} \vee \mathbf{1} - \bar{\mathbf{E}} \cdot (\mathbf{P}^{\leftarrow} - \bar{\mathbf{R}} \cdot \mathbf{N}^{\leftarrow}) \neq \mathbf{0} \vee \mathbf{1} - \bar{\mathbf{E}} \cdot (\mathbf{P}^{\rightarrow} - \bar{\mathbf{L}} \cdot \mathbf{N}^{\rightarrow}) \neq \mathbf{0}.$$

Leveraging the conditions from $\mathbf{H}_6$ on c_{ln} and c_{rn}, using knowledge soundness of $\mathsf{CP}_{\mathsf{had}}$ and the binding of the matrix commitment scheme we have $\Pr[\mathbf{H}_6 = 1] \leq \Pr[\mathbf{H}_7 = 1] + \mathsf{negl}(\lambda)$.

Notice that when $\mathbf{H}_7 = 1$ the constraints of Eqs. (19) and (20) of Lemma 4 hold.

Hybrid $\mathbf{H}_8$. The Hybrid $\mathbf{H}_8$ is the same as $\mathbf{H}_7$, but it additionally returns 0 if $\mathbf{N}^{\rightarrow} - \mathbf{N}^{\leftarrow} - \mathbf{1} \notin [B]^{N_{\mathsf{tot}} \times d}$. By the binding property of the commitment scheme and by the knowledge soundness of $\mathsf{CP}_{\mathsf{range}}$ we have $\Pr[\mathbf{H}_7] \leq \Pr[\mathbf{H}_8] + \mathsf{negl}(\lambda)$. Notice that when $\mathbf{H}_8 = 1$ we have that $\mathbf{N}^{\leftarrow}_{i,j} < \mathbf{N}^{\rightarrow}_{i,j}$ for any i, j (namely, the constraint in Item b) of Lemma 4 holds).

Hybrid $\mathbf{H}_9$. The Hybrid $\mathbf{H}_9$ is the same as $\mathbf{H}_8$, but it additionally extracts the (sparse) matrices $\bar{\mathbf{L}}, \underline{\mathbf{R}}$ and returns 0 if $\bar{\mathbf{L}} + \underline{\mathbf{R}}$ is not a permutation of the matrix $(\mathbf{0} \| \mathbf{I}_{N_{\mathsf{tot}}-1})$. We reduce to knowledge soundness of $\mathsf{CP}_{\mathsf{perm}}$ noticing that the polynomial $id(X)$ in the indexer of the instance $((N_{\mathsf{tot}} - 1, id), \mathsf{c}_L + \mathsf{c}'_R)$ is a valid representation (according to the sparse matrix commitment scheme) of the sparse-matrix $(\mathbf{0} \| \mathbf{I}_{N_{\mathsf{tot}}-1})$.

Hybrid $\mathbf{H}_{10}$. The Hybrid $\mathbf{H}_{10}$ is the same as $\mathbf{H}_9$, but it returns 0 if $\underline{\mathbf{R}}$ is not a shift of $\bar{\mathbf{R}}$. By the knowledge soundness of $\mathsf{CP}_{\mathsf{shift}}$ we have that the probability of the two hybrids is negligibly close.

Notice that if $\mathbf{H}_{10} = 1$ then all constraints in Lemma 4 are satisfied thus there must exist a valid quasi-complete decision tree T with N_{tot} nodes associated to the extracted matrices returned by the extractor, which is in contradiction with the winning condition of the adversary, thus the probability of $\mathbf{H}_{10} = 1$ is equal to 0.

E.3 CP-SNARK for Statistics on Decision Trees

Theorem 4. *If the building blocks have the properties stated in Items 1 to 3 then* $\mathsf{CP}_{DT} = (\mathsf{Derive}, \mathsf{Prove}, \mathsf{Verify})$ *define an Universal CP-SNARK for the indexed CP-relation* $\hat{\mathcal{R}}_{\mathsf{DTstat}}$.

Proof. Zero-knowledge follows easily from the hiding of the commitments $\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3$ and the zero-knowledge of the three CP-SNARKs. In particular, the zero-knowledge simulator could sample the commitments by commiting to dummy values and run the zero-knowledge simulators of the three CP-SNARKs.

We recall that $\mathsf{c}_{\leftarrow}$ (resp. $\mathsf{c}_{\rightarrow}$) commits to the matrix $\bar{\mathbf{F}}_{\leftarrow}$ (resp. $\bar{\mathbf{F}}_{\rightarrow}$) whose first N_{int} rows are filled with 0 and the remaining submatrix is $\mathbf{F}^{\leftarrow}$ (resp. $\mathbf{F}^{\rightarrow}$).

The completeness follows by Lemma 3, the homomorphic properties of the matrix commitment scheme and the completeness of the CP-SNARKs. In particular, the lemma implies that c_3 commits to the vector $(\mathsf{T}(\mathbf{x}_j))_{j\in[m]}$, moreover by definition of k_{T}, the matrix $(\mathbf{X} - \bar{\mathbf{F}}_{\leftarrow|K})$ contains non negative numbers smaller than B and the matrix $(\bar{\mathbf{F}}_{\rightarrow|K} - \mathbf{X} - \mathbf{1})$ contains non negative numbers smaller than B.

For knowledge soundness, we define the extractor of the CP-SNARK to be the same as the extractor $\mathcal{E}_{\mathsf{Com}}$ of the extractable commitment scheme. We proceed with a sequence of hybrids where $\mathbf{H}_0$ is the knowledge soundness game for the CP_{DT} with extractor $\mathcal{E}_{\mathsf{Com}}$.

Hybrid $\mathbf{H}_1$. Let $\tilde{\mathsf{T}}$ (and opening material $(\rho_v, \rho_{\leftarrow}, \rho_{\rightarrow})$) be the extracted quasi-complete decision tree, the hybrid $\mathbf{H}_1$ additionally computes $(\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}, \mathbf{v}, \mathbf{L}, \mathbf{R}, \mathbf{E}) \leftarrow \mathsf{Encode}(\tilde{\mathsf{T}})$ and sets $\bar{\mathbf{F}}_{\leftarrow}, \bar{\mathbf{F}}_{\rightarrow}$ be the sub-matrices (relative to the leaf) of $\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}$ and returns 0 if $\mathbf{v}, \bar{\mathbf{F}}_{\leftarrow}, \bar{\mathbf{F}}_{\rightarrow}$ (and their opening materials) do not commit to $\mathsf{c}_v, \mathsf{c}_{\leftarrow}, \mathsf{c}_{\rightarrow}$. It is easy to see that $\Pr[\mathbf{H}_0] \leq \Pr[\mathbf{H}_1] + \mathsf{negl}(\lambda)$, where the latter negligible value depends on the error of the extractable decision-tree commitment scheme.

Hybrid $\mathbf{H}_2$. The hybrid $\mathbf{H}_2$ is the same as $\mathbf{H}_1$ but it additionally runs the extractor of $\mathsf{CP}_{\mathsf{lookup}*}$ extracting matrices $\tilde{\mathbf{M}}_1, \tilde{\mathbf{M}}_2, \tilde{\mathbf{m}}_3, \tilde{\mathbf{F}}_{\leftarrow}, \tilde{\mathbf{F}}_{\rightarrow}, \tilde{\mathbf{v}}$ and it outputs 0 if $(\tilde{\mathbf{M}}_1 \| \tilde{\mathbf{M}}_2 \| \tilde{\mathbf{m}}_3) \not\prec (\tilde{\mathbf{F}}_{\leftarrow} \| \tilde{\mathbf{F}}_{\rightarrow} \| \tilde{\mathbf{v}})$. It is easy to see that $\Pr[\mathbf{H}_1] \leq \Pr[\mathbf{H}_2] + \mathsf{negl}(\lambda)$, where the latter negligible value depends on the knowledge soundness error of $\mathsf{CP}_{\mathsf{lookup}*}$.

Hybrid $\mathbf{H}_3$. The hybrid $\mathbf{H}_3$ is the same as $\mathbf{H}_2$ but it additionally returns 0 if $(\tilde{\mathbf{F}}_{\leftarrow}, \tilde{\mathbf{F}}_{\rightarrow}, \tilde{\mathbf{v}}) \neq (\bar{\mathbf{F}}_{\leftarrow}, \bar{\mathbf{F}}_{\rightarrow}, \mathbf{v})$, where the former matrices are extracted from the extractor of $\mathsf{CP}_{\mathsf{lookup}*}$ and the latter from the extractor of the extractable commitment. It is easy to see that $\Pr[\mathbf{H}_2] \leq \Pr[\mathbf{H}_3] + \mathsf{negl}(\lambda)$, because the distinguishing event allows to break the binding property of the matrix commitment scheme.

Hybrid $\mathbf{H}_4$. The hybrid $\mathbf{H}_4$ is the same as $\mathbf{H}_3$ but that additionally returns 0 if $(\mathbf{X} - \tilde{\mathbf{M}}_1) \notin [B]^{m\times d}$. To show $\Pr[\mathbf{H}_3] \leq \Pr[\mathbf{H}_4] + \mathsf{negl}(\lambda)$ we can follow the same two-fold strategy of the previous two hybrids, namely we can (1) define a sub-hybrid experiment where we run the extractor of $\mathsf{CP}_{\mathsf{range}}$ and return 0 if

the extracted matrix is not in the range thus reducing to the extractability of $\mathsf{CP}_{\mathsf{range}}$ and (2) we can show that the extracted matrix must be equal to $\mathbf{X}-\tilde{\mathbf{M}}_1$ because of the binding and homomorphic property of the matrix commitment scheme.

Hybrid $\mathbf{H}_5$. Similarly to the previous item, the hybrid $\mathbf{H}_5$ additionally returns 0 if $(\tilde{\mathbf{M}}_2 - \mathbf{X} - \mathbf{1}) \notin [B]^{m\times d}$. We can show $\Pr[\mathbf{H}_4] \leq \Pr[\mathbf{H}_5] + \mathsf{negl}(\lambda)$ in a very similar manner to the previous step.

Hybrid $\mathbf{H}_6$. Let K be the set of indexes such that $(\tilde{\mathbf{M}}_1 \| \tilde{\mathbf{M}}_2 \| \tilde{\mathbf{m}}_3) = (\bar{\mathbf{F}}_{\leftarrow} \| \bar{\mathbf{F}}_{\rightarrow} \| \mathbf{v})_{|K}$. The hybrid $\mathbf{H}_6$ additionally returns 0 if $K \neq \{k_{\tilde{\mathsf{T}}}(\mathbf{x}_1), \ldots, k_{\tilde{\mathsf{T}}}(\mathbf{x}_m)\}$. By the change introduced in $\mathbf{H}_1$ and by Lemma 3 for any $i \neq j$ the boxes $(\mathbf{F}^{\leftarrow}{}_i, \mathbf{F}^{\rightarrow}{}_i)$ and $(\mathbf{F}^{\leftarrow}{}_j, \mathbf{F}^{\rightarrow}{}_j)$ do not overlap. Thus for any $i \in [m]$ there must exists only one index k_i such that $(\tilde{\mathbf{M}}_1)_i = \bar{\mathbf{F}}_{\leftarrow k_i}$ (resp. $(\tilde{\mathbf{M}}_2)_i = \bar{\mathbf{F}}_{\leftarrow k_i}$), moreover by the changes introduced in $\mathbf{H}_4$ and $\mathbf{H}_5$ we have that $\bar{\mathbf{F}}_{\leftarrow k_i} \leq \mathbf{x}_i < \bar{\mathbf{F}}_{\rightarrow k_i}$, thus such a unique index k_i must be equal to $k_{\tilde{\mathsf{T}}}(\mathbf{x}_i)$. We have $\Pr[\mathbf{H}_6] = \Pr[\mathbf{H}_5]$.

Hybrid $\mathbf{H}_7$. The hybrid $\mathbf{H}_7$ additionally returns 0 if $y \neq S((\tilde{\mathbf{m}}_3)_1, \ldots, (\tilde{\mathbf{m}}_3)_m)$. Similarly previous hybrids we can reduce to the biding of the vector commitment to prove that the vector $\tilde{\mathbf{m}}_3$ is the same that the knowledge extractor of $\mathsf{CP}_{\mathsf{stat}}$ would compute and then reduce to the knowledge soundness of $\mathsf{CP}_{\mathsf{stat}}$. Thus, we have $\Pr[\mathbf{H}_6] \leq \Pr[\mathbf{H}_7] + \mathsf{negl}(\lambda)$.

We show that the probability for $\mathbf{H}_7$ is 0, we can now conclude the proof by chaining the equations proved in the previous steps. Because of the changes in $\mathbf{H}_2$ and $\mathbf{H}_3$, in $\mathbf{H}_7$ we have that $\tilde{\mathbf{m}}_3 = \mathbf{v}_{|K}$, moreover, by the check introduced in $\mathbf{H}_6$ we have that $K = \{k_{\tilde{\mathsf{T}}}(\mathbf{x}_1), \ldots, k_{\tilde{\mathsf{T}}}(\mathbf{x}_m)\}$. Notice this already implies that $\forall j : \mathsf{T}(\mathbf{x}_j) \neq \perp$. Moreover, because of the check introduced in $\mathbf{H}_7$ we have $y = S(v_{k_{\tilde{\mathsf{T}}}(\mathbf{x}_1)}, \ldots, v_{k_{\tilde{\mathsf{T}}}(\mathbf{x}_m)})$. These last two implications negate the winning condition of the knowledge soundness of CP_{DT}, thus $\Pr[\mathbf{H}_7] = 0$.

E.4 CP-SNARKs for Linear Relations with Sparse Matrix Commitment

Let $\mathbf{M}$ be a *basic matrix*, namely a matrix whose rows are elementary vectors. Let $\mathbb{H}$ be the subgroup of $\mathbb{F}$ generated by h and defined in the commitment key for the vector commitment[22]. For any basic matrix $\mathbf{M} \in \{0,1\}^{k\times n}$ and $k, n \in \mathbb{N}$, let $col_{\mathbf{M}}(X)$ be the (low-degree) polynomial such that $col_{\mathbf{M}}(h^i) = h^j$ where the i-th row of $\mathbf{M}$ is the vector $\mathbf{e}_j^\top$ (notice that $col_{\mathbf{M}}$ can also be interpreted as a vector whose i-th element is the value h^j). We define the sparse (hiding) commitment of a matrix $\mathbf{M}$ as a (hiding) polynomial commitment of $col_{\mathbf{M}}$. Namely, we define:

$$\mathsf{sparseCom}(\mathsf{ck}, \mathbf{M}, \rho) := \mathsf{Com}(\mathsf{ck}, col_{\mathbf{M}}, \rho).$$

We write $\boxed{\mathbf{M}}$ to underline that the matrix $\mathbf{M}$ is committed with a sparse matrix commitment. Our goal is to realize a CP-SNARK for the relation:

$$\hat{\mathcal{R}}_{\mathsf{lin}} = \left\{ \mathsf{pp}; \varepsilon; (\boxed{M}, \mathbf{N}, \mathbf{R}) : \mathbf{M}\cdot\mathbf{N} = \mathbf{R}, \quad \mathbf{N} \in \mathbb{F}^{n\times d} \right\}.$$

[22] We assume $|\mathbb{H}| \geq k, n$.

Our building blocks are CP-SNARKs CP' and CP'' for the relations:

$$\hat{\mathcal{R}}'_{\mathsf{lin}} = \left\{\mathsf{pp}; \varepsilon; (\boxed{M}, \mathbf{n}, \mathbf{r}) : \mathbf{M} \cdot \mathbf{n} = \mathbf{r}, \quad \mathbf{n} \in \mathbb{F}^{n \cdot d}\right\},$$
$$\hat{\mathcal{R}}''_{\mathsf{lin}} = \left\{\mathsf{pp}; \mathbf{M}; \varepsilon; (\mathbf{n}, \mathbf{r}) : \mathbf{M} \cdot \mathbf{n} = \mathbf{r}, \quad \mathbf{n} \in \mathbb{F}^{n \cdot d}\right\}.$$

Notice that above $\mathbf{n}$ is a vector, while n is an integer. The difference between the two relations is that in the first $\mathbf{M}$ is part of the witness while in the second $\mathbf{M}$ is part of the index. Notice that an instantiation of CP' can be found in Baloo [37] while instantiations of CP'' can be derived easily from zkSNARKs for R1CS based on holographic polynomial IOP such as [7,21,31]. The prover time complexity for the latter scheme depends quasi-linearly on the sparsity of the matrix $\mathbf{M}$.

Gadget Matrices, Operators and Vectorizations. Consider the matrix $\bar{\mathbf{I}}_{n,d}$ which stacks d identity matrices $\mathbf{I}_n$ of size n, namely $\bar{\mathbf{I}}_{n,d} = \mathbf{I}_n \otimes \mathbf{1}$ where $\mathbf{1}$ is of length d. Consider the linear operator r that maps a matrix $\mathbf{A}$ to the vectorization row-by-row of $\mathbf{A}$, similarly, consider the linear operator c that maps a matrix $\mathbf{A}$ to the vectorization column-by-column of $\mathbf{A}$. Finally, we consider the permutation matrix $\mathbf{P}$ such that for any $\mathbf{A}$ we have:

$$\mathbf{P} \cdot \mathsf{r}(\mathbf{A}) = \mathsf{c}(\mathbf{A}) \tag{30}$$

We also recall that to compute a matrix commitment of $\mathbf{A}$ we are implicitly computing a vector commit to $\mathsf{r}(\mathbf{A})$.

Let $\mathbf{M} \otimes \mathbf{I}$ be the tensor-product of $\mathbf{M}$ and $\mathbf{I}$, namely the following matrix:

$$\mathbf{M} \otimes \mathbf{I} = \begin{bmatrix} \mathbf{M} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{M} \end{bmatrix}$$

It is not hard to prove that the following holds:

$$\mathbf{M} \cdot \mathbf{N} = \mathbf{R} \iff (\mathbf{M} \otimes \mathbf{I}) \cdot \mathsf{c}(\mathbf{N}) = \mathsf{c}(\mathbf{R}). \tag{31}$$

Moreover if $\mathbf{M} \in \{0,1\}^{k \times n}$ is a basic matrix then :

$$col_{\mathbf{M} \otimes \mathbf{I}}(h^{k \cdot i + j}) = col_{\mathbf{M}}(h^j) + n \cdot i.$$

Namely, the $(k \cdot i + j)$-row of $\mathbf{M} \otimes \mathbf{I}$ contains the j-th row vector of $\mathbf{M}$ *shifted* of $n \cdot i$ columns. The equation above can be translated in the vector domain. Namely, if we let $\mathbf{c}$ the evaluation over $\mathbb{H}$ of $col_{\mathbf{M}}(X)$ and $\mathbf{c}'$ the evaluation over $\mathbb{H}$ of $col_{\mathbf{M} \otimes \mathbf{I}}(X)$ we have that:

$$\mathbf{I}_{n,d} \cdot \mathbf{c} = \mathbf{c}' - \mathbf{s}$$

where the *shifting vector* $\mathbf{s}$ is such that $s_{k \cdot i + j} = n \cdot i$ for all $i, j \in \mathbb{N}$ and $j < k$. Thus for any basic matrix $\mathbf{M} \in \{0,1\}^{k \times n}$ and for any d we have:

$$\mathbf{M}' = \mathbf{M} \otimes \mathbf{I}_d \iff \bar{\mathbf{I}}_{n,d} \cdot col_{\mathbf{M}} = col_{\mathbf{M}'} - \mathbf{s}. \tag{32}$$

Our Scheme. Our CP-SNARK scheme is shown in Fig. 11.

$\underline{\mathsf{KGen}(\mathsf{ck})}$:

Run the keygen algorithms of CP' and CP''. Moreover, derive proving and verification keys for the matrix $\bar{\mathbf{I}}_{n,d}$ and for the matrix $\mathbf{P}$.

$\underline{\mathsf{Prove}(\mathsf{srs}, (\mathsf{c}_M, \mathsf{c}_N, \mathsf{c}_R), (M, \mathbf{N}, \mathbf{R}), (\rho_M, \rho_N, \rho_R))}$:

Commit $\mathsf{c}_{R,c} \leftarrow \mathsf{Com}(\mathsf{ck}, \mathsf{c}(\mathbf{R}))$, $\mathsf{c}_{N,c} \leftarrow \mathsf{Com}(\mathsf{ck}, \mathsf{c}(\mathbf{N}))$.
Prove that $(\mathbf{P}; \varepsilon; \mathsf{r}(\mathbf{R}), \mathsf{c}(\mathbf{R})) \in \hat{\mathcal{R}}''_{\mathsf{lin}}$.
Prove that $(\mathbf{P}; \varepsilon; \mathsf{r}(\mathbf{N}), \mathsf{c}(\mathbf{N})) \in \hat{\mathcal{R}}''_{\mathsf{lin}}$.
Compute $\mathbf{M}' = \mathbf{M} \otimes \mathbf{I}$ and commit $\mathsf{c}_{M'} \leftarrow \mathsf{sparseCom}(\mathsf{ck}, \mathbf{M}')$.
Prove $(\bar{\mathbf{I}}_{n,d}; \varepsilon; col_{\mathbf{M}}, col_{\mathbf{M}'} - \mathbf{s}) \in \hat{\mathcal{R}}''_{\mathsf{lin}}$.
Prove $(\varepsilon; M', \mathsf{c}(\mathbf{N}), \mathsf{c}(\mathbf{R})) \in \mathcal{R}'_{\mathsf{lin}}$.

$\underline{\mathsf{Verify}(\mathsf{srs}, (\mathsf{c}_M, \mathsf{c}_N, \mathsf{c}_R), \pi)}$:

Parse $\pi = (\mathsf{c}_{M'}, \mathsf{c}_{R,c}, \pi_1, \pi_2, \pi_3)$.
Verify the proofs:

1. π_1 with instance $(\mathsf{c}_R, \mathsf{c}_{R,c})$ and verification key for $\mathbf{P}$ (for CP')
2. π_2 with instance $(\mathsf{c}_N, \mathsf{c}_{N,c})$ and verification key for $\mathbf{P}$ (for CP')
3. π_3 with instance $(\mathsf{c}_{M'} - \mathsf{Com}(\mathsf{ck}, \mathbf{s}), \mathsf{c}_M)$ and verification key for $\bar{\mathbf{I}}_{n,d}$ (for CP').
4. π_4 with instance $(\mathsf{c}_{M'}, \mathsf{c}_{N,c}, \mathsf{c}_{R,c})$ (for CP'').

Fig. 11. Our CP-SNARK scheme for linear relations with sparse matrix commitments.

Theorem 8. *The CP-SNARK defined in Fig. 11 is zero-knowledge and knowledge sound.*

Proof (Proof Sketch.). Zero-knowledge is trivially implied by the zero-knowledge of the CP-SNARK CP' and CP'' and by the hiding property. As for knowledge soundness, we let the extractor be the same of CP'' on instance $(\mathsf{c}_{M'}, \mathsf{c}_{N,c}, \mathsf{c}_{R,c})$. Because of the knowledge soundness of π_3 and by Eq. (32) we have that the extracted matrix $\mathbf{M}'$ is of the form $\mathbf{M}' = \mathbf{M} \otimes \mathbf{I}$. Because of the knowledge soundness of π_1 and π_2 and Eq. (30) we have that the commitments $\mathsf{c}_{N,c}$ and $\mathsf{c}_{R,c}$ commits to $\mathsf{c}(\mathbf{N})$ and $\mathsf{c}(\mathbf{R})$. Finally, because of the knowledge soundness of π_4 we have that $\mathbf{M}' \cdot \mathsf{c}(\mathbf{N}) = \mathsf{c}(\mathbf{R})$ and thus because of Eq. (31) we have that the extracted witness (once parsed adequately) is valid for the relation.

F Efficiency Breakdown for our Matrix Lookup Arguments

- **Proof size.** Our protocol requires one zkcq^+proof $(9\mathfrak{g}_1 + 1\mathfrak{f})$, plus: three KZG commitments, four group elements (π_R and $\pi_{R'}$), and one field element. The total proof size is $16\mathfrak{g}_1 + 2\mathfrak{f}$.
- **Proving time.** In addition to the zkcq^+prover (requiring $O(nd)$ group operations and $O(nd\log(nd))$ field operations) the prover performs:
  - field operations required to compute polynomial evaluations for w, σ and $\nu_{\bar{\mathbb{K}}}(X)$. We have $\deg(w) = n$, $\deg(\sigma) = n \cdot d$ and $\deg(\nu_{\bar{\mathbb{K}}}(X)) = n \cdot d$.
  - three multiexponentiations of size n in order to compute $[\sigma(s)]_1$, $[\sigma(\kappa \cdot s)]_1$, $[w(s)]_1$; notice that we use the sparsity of σ (see definition of σ) (we ignore the masking factors for simplicity).

  - four multiexponentiations of size $n \cdot d$ in order to compute group elements π_R and $\pi_{R'}$ (two batched KZG proofs in zero-knowledge).
- **Verification time.** In addition to the steps for zkcq^+'s verifier (requiring 7 pairings), verification requires: a constant number of group operations and six pairings (we use the fact that we can batch some of the pairing equations). The total number of pairings performed by the verifier is 13.