Protostar: Generic Efficient Accumulation/Folding for Special Sound Protocols

Benedikt Bünz
Espresso Systems

Binyi Chen
Espresso Systems

May 4, 2023

Abstract

We provide a generic, efficient accumulation (or folding) scheme for any $(2k-1)$-move special sound protocol with a verifier that checks ℓ degree-d equations. The accumulation verifier only performs $k+d-1$ elliptic curve multiplications and $k+1$ field operations. Alternatively the accumulation verifier performs just k EC multiplications at the cost of 1 additional hash and $O(k+d+\log \ell)$ field operations. Using the compiler from BCLMS21 (Crypto 21), this enables building efficient IVC schemes where the recursive circuit only depends on the number of rounds and the verifier degree of the underlying special-sound protocol but not the proof size or the verifier time. We use our generic accumulation compiler to build Protostar. Protostaris a non-uniform IVC scheme for Plonk that supports high-degree gates and (vector) lookups. The recursive circuit is dominated by 3 group scalar multiplications and hashes and $O(d^* + \log n)$ field operations, where d^* is the degree of the highest gate and n is the number of gates in a supported circuit. The scheme does not require a trusted setup or pairings, and the prover does not need to compute any FFTs. The prover in each accumulation/IVC step is also only logarithmic in the number of supported circuits and independent of the table size in the lookup.

Contents

1 Introduction

Incrementally Verifiable Computation[Val08] is a powerful primitive that enables a possibly infinite computation to be run, such that the correctness of the state of the computation can be verified at any point. IVC, and it's generalization to DAGs, PCD[CT10], have many applications, including distributed computation[BCCT13; CTV15], blockchains[BMRS20; KB20], verifiable delay functions [BBBF18], verifiable photo editing [NT16], and SNARKs for machine-computations[BCTV14]. An IVC-based VDF construction is the current candidate VDF for Ethereum[KMT22]. One of the most exciting applications of IVC and PCD is the ZK-EVM. This is an effort to build a proof system that can prove that Ethereum blocks, as they exist today, are valid[But22].

Accumulation and folding. Historically, IVC was built from recursive SNARKs, proving that the previous computation step had a valid SNARK that proves correctness up to that point. Recently, an exciting new approach was initiated by Halo[BGH19] and has led to a series of significant advances[BCMS20; BCLMS21; KST22]. The idea is related to batch verification. Instead of verifying a SNARK at every step of the computation, we can instead *accumulate* the SNARK verification check with previous checks. We define an *accumulator*[1] such that we can combine a new SNARK and an old accumulator into a new accumulator. Checking or *deciding* the new accumulator implies that all previously accumulated SNARKs were valid. Now the recursive statement just needs to ensure the accumulation was performed correctly. Amazingly, this accumulation step can be significantly cheaper than SNARK verification[BGH19; BCMS20]. Even more surprising, this process does not even require a SNARK but instead can be instantiated with a non-succinct NARK[BCLMS21], as long as there exists an efficient accumulation scheme for that NARK. The most efficient accumulation (aka folding) scheme constructions yield IVC constructions, where the recursive circuit is dominated by as few as 2 elliptic curve scalar multiplications[BCLMS21; KST22]. These constructions only require the discrete logarithm assumption in the random oracle model and, unlike many efficient SNARK-based IVCs, do not require a trusted setup, pairings, or FFTs. These constructions build an accumulation scheme for one fixed (but universal) R1CS language by taking a random linear combination between the accumulator and a new proof. R1CS is a minimal expression of NP, defined by three matrices A, B, C, that close resembles arithmetic circuits with addition and multiplication gates. However, it has limited flexibility, especially as the current constructions require fixing R1CS matrices that are used for all computation steps. These limitations are especially problematic for ZK-EVMs. In a ZK-EVM, each VM instruction (OP-CODE) is encoded in a different circuit. Each circuit uses high-degree gates, instead of just multiplication, and so-called lookup gates [GWC19]. These lookup gates enable looking up that a circuit value is in some table, simplifying range proofs and bit-operations.

[1] Unrelated to set accumulators.

These R1CS-based accumulation schemes contrast non-IVC SNARK developments, with an increased focus on high-degree gate[GWC19; CBBZ22] and lookup support[GW20]. For lookups, a recent line of work has shown that if the table can be pre-computed, we can perform n lookups in a table of size T in time $O(n \log n)$, independent of T[ZBKMNS22; PK22; ZGKMR22; EFG22].

More expressive accumulation. There have been efforts to build accumulation schemes that overcome the limitations of fixed R1CS. SuperNova[KS22] enables selecting the appropriate R1CS instance at runtime without a recursive circuit that is linear in all R1CS instances. The approach, however, still has limitations. The recursive circuit is linear in the number of R1CS circuits, and additionally, the accumulator, and thus the final proof, is still linear in the total size of all instances. Sangria[Moh23] describes an accumulation scheme for a Plonk-like[GWC19] constraint system with degree-2 gates. It also proposes a solution for higher-degree gates in the future work section but without security proof. These accumulation schemes are built from simple underlying protocols performing a linear combination between an accumulator and a proof. However, the constructions seem ad hoc and need individual security proof. This leads us to our main research questions:

Recipe for accumulation Is there a general recipe for building accumulation schemes? Can we formalize this recipe, simplifying the task of constructing secure and efficient accumulation schemes?

Efficient accumulation for ZK-EVM Can we build an accumulation/folding scheme for a language that combines the benefits of the most advanced proof systems today? Can we support multiple circuits, high-degree, and lookup gates?

We answer both questions positively. Firstly we show a general compiler that takes any $(2k-1)$-move special sound interactive argument for an NP-complete relation $\mathcal{R}_{\mathsf{NP}}$ with an algebraic degree d verifier and construct an efficient IVC-scheme from it. This is done in 4 simple steps.

1. We compress the prover message by committing to them in a homomorphic commitment scheme.
2. Then we apply the Fiat-Shamir transform to yield a secure NARK. [AFK22; Wik21]
3. We build a simple and efficient accumulation scheme that samples a random challenge α and takes a linear combination between the current accumulator and the new NARK.
4. We apply the compiler by [BCLMS21] to yield a secure IVC scheme.

The recursive circuit of this transformation is dominated by only $d + k - 1$ scalar multiplications in the additive group of the commitment scheme[2] for a protocol with k prover messages and a degree d verifier. For R1CS, where $k = 1$ and $d = 2$, this yields the same protocol and efficiency as Nova[KST22]. We also provide an optimization that reduces the size of the recursive circuit to only k group scalar multiplication, at the cost of extra field multiplications, in the message space field.

Efficient simple protocols for $\mathcal{R}_{\text{mplkup}}$. Equipped with this compiler, we design ProtoStar, a simple and efficient IVC scheme for a highly expressive language $\mathcal{R}_{\text{mplkup}}$ that supports multiple non-uniform circuits and enables high degree and lookup gates. The schemes can be instantiated from any linearly homomorphic vector commitment, e.g., the discrete logarithm-based Pedersen commitment[Ped92], and do not require a trusted setup or the computation of large FFTs. The protocol has several advantages over prior schemes:

Non-uniform IVC without overhead. Each iteration has a program counter **pc** that selects one out of I circuits. Part of the circuit constraints **pc**; e.g., **pc** could depend on the iteration or indicate which instruction within a VM is executed. The IVC-prover, including the recursive statement, only requires one exponentiation per non-zero bit in the witness. The prover's computation is independent of I.

Flexible high degree gates. Our protocol supports Plonk-like constraint systems with degree d gates instead of just addition and multiplication. The recursive statement consists of $d + 1$ group scalar multiplications. Using an optimization, we can reduce this to 3 group scalar multiplications at the cost of 1 additional hash and $O(d)$ field operations. Unlike in traditional Plonk, there is no additional cost for additional gate types (of degree less than d) and additional selectors. This enables a high level of non-uniformity, even within a circuit.

Lookups, linear and independent of table size. ProtoStar supports lookup gates that ensure a value is in some precomputed table T. In each computation step, the prover commits to 2 vectors of length ℓ_{lk}, where ℓ_{lk} is the number of lookups. The prover, in each step, is independent of the table size (assuming free indexing in T). We also support tables that store tuples of size v using $\log_2(v)$ additional challenge computations within the recursive circuit.

Our protocols are built of multiple small building blocks. In the protocol for high-degree gates, the prover simply sends the witness, and the degree d verifier checks the circuit with degree d gates. For lookup, we leverage an insight by Haböck [Hab22] on logarithmic derivates. This yields a protocol where a prover performing ℓ_{lk} in a table of size T only needs to commit to two vectors of length ℓ_{lk}, independent of T. This is the most efficient

[2]When instantiated with elliptic curve Pedersen commitments, this translates to $d + k - 1$ elliptic curve multiplications. This is usually the largest component of the recursive statement.

	Protostar	HyperNova	SuperNova
Language	Degree d gates	Degree d gates	R1CS (degree 2)
Non-uniform	yes	no	yes
P native	$\lvert\mathbf{w}\rvert\,\mathbb{G}$ $O(\lvert\mathbf{w}\rvert d\log^2 d)\,\mathbb{F}$	$\lvert\mathbf{w}\rvert\,\mathbb{G}$ $O(\lvert\mathbf{w}\rvert d\log^2 d)\,\mathbb{F}$	$\lvert\mathbf{w}\rvert\,\mathbb{G}$
extra P native w/ lookup	$O(\lvert\ell_{\mathsf{lk}}\rvert)\,\mathbb{G}$	$O(T)\,\mathbb{F}$	N/A
P recursive	$1\mathbb{G}+2\mathsf{H}$ $O(d+\log n)\mathbb{F}$	$1\mathbb{G}+\log n\mathsf{H}$ $O(d\log n)\mathbb{F}$	$2\mathbb{G}$ $O_\lambda(1)\mathsf{H}+1\mathsf{H}_{\mathbb{G}}$
extra P recursive w/ lookup	$2\mathbb{G}+1\mathsf{H}$	$O(\log T)\,\mathsf{H}$ $O(\ell_{\mathsf{lk}}\log T)\,\mathbb{F}$	N/A

Table 1: The comparison between IVCs.

lookup protocol today. While the verification is linear time, it is low degree (2) and thus compatible with our generic compiler. Combining all these yields Protostar, a new IVC-scheme for $\mathcal{R}_{\text{mplkup}}$. We compare Protostar, with Super-Nova, in Table 1 (for more detail see Corollary 1): In the table, $|\mathbf{w}|$ is the size of the witness for circuit i, and ℓ_{lk} is the number of lookups in a table of size T. $\mathbb{G}$ is the cost of a group scalar multiplication. $\mathbb{F}$ is the cost of a field multiplication. H is the cost of a hash. $\mathsf{H}_{\mathbb{G}}$ is the cost of a hash-to-group function. Note that the $O_\lambda(1)\,\mathsf{H}$ in SuperNova's recursive circuit involves constant number of hashes to the input of two accumulator instances and one circuit verification key, by using multiset-based offline memory checking in a circuit [SAGL18].

Remark 1. *A more fine-grained number of field operations of* P *native is* $O(|\mathbf{w}|(d+\log n)\log^2(d+\log n))$ *if we want to achieve* $1\mathbb{G}+2\mathsf{H}+O(d+\log n)\mathbb{F}$ *in* P *recursive circuit. Note* $O(|\mathbf{w}|(d+\log n)\log^2(d+\log n)) = O(|\mathbf{w}|d\log^2 d)$ *when* $d\approx\log n$*. It's an interesting open question on whether we can achieve optimal complexity for both the* P *native and* P *recursive complexities when* d *is much less than* $\log n$*.*

Concurrent work. In a paper concurrent with this work, Kothapalli and Setty [KS23] introduce an IVC for high degree relations. They use a generalization of R1CS called customizable constraint systems (CCS) [STW23] that covers the Plonkish relations. It also enables gates with a high additive fan-in. Protostar also has no restriction to the fan-in an individual gate has, so it is an interesting, open question whether it can directly be applied to CCS. HyperNova is based on so-called multi-folding schemes. They also provide a lookup argument suitable for recursive arguments. However, they do not explicitly explain how to integrate lookup to Plonk in their IVC scheme or provide any explicit constructions for non-uniform computations. Their scheme is built using sumchecks [LFKN92] and the resulting IVC recursive circuit is dominated by 1 group scalar multiplication, $2\log n$ hashes and $O(d\log n+\ell_{\mathsf{in}})$ field multiplications where d is the custom gate degree, n is the number

of gates and ℓ_{in} is the public input length. In comparison, our IVC recursive circuit (without lookup or non-uniformity support) is dominated by d group scalar multiplications, 1 hash and $1 + \ell_{\mathsf{in}}$ field operations for low-degree gates; or only 1 group scalar multiplications, 2 hashes and $O(\log n + \ell_{\mathsf{in}} + d)$ field multiplications for high-degree gates after an optimization. A detailed comparison is given in Table 1.

For a lookup relation with table size T and ℓ_{lk} lookup gates, their accumulation/folding scheme leads to an accumulation prover whose work is dominated by $O(T)$ field operations and an accumulation verifier whose work is dominated by $O(\ell_{\mathsf{lk}} \log T)$ field operations and $O(\log T)$ hashes. This is undesirable when the table size $T \gg \ell_{\mathsf{lk}}$. In comparison, our scheme has prover complexity $O(\ell_{\mathsf{lk}})$ and the verifier is only dominated by 3 group scalar multiplications, 2 hashes and 2 field multiplications. Lastly, their lookup scheme does not support vector-valued lookups, which is essential for applications like ZK-EVM and encoding bit-wise operations in circuits.

1.1 Technical overview

Given an NP-complete relation $\mathcal{R}$, we introduce a generic framework for constructing efficient incremental verifiable computation (IVC) schemes with predicates expressed in $\mathcal{R}$. For $\mathcal{R}$ being the non-uniform Plonkup circuit satisfiability relation, we obtain an efficient (non-uniform) IVC scheme for proving correct program executions on stateful machines (e.g., EVM). The framework starts by designing a simple special-sound protocol Π_{sps} for relation $\mathcal{R}$, which is easy to analyze. Next, we use a generic compiler to transform Π_{sps} into a Non-interactive Argument of Knowledge Scheme (NARK) whose verification predicate is easy to accumulate/fold. Finally, we build an efficient accumulation/folding scheme for the NARK verifier, and apply the generic compiler from [BCLMS21] to obtain the IVC/PCD scheme for relation $\mathcal{R}$.

The paper begins by describing the compiler from special-sound protocols to NARKs in Section 3, and presents an efficient accumulation scheme for the compiled NARK verifier in Section 3.4. Next, we describe simple and efficient special-sound protocols for Plonkup circuit-satisfiability relations in Section 5, and extend it to support non-uniform computation in Section 6. We give an overview of our approach below.

Efficient IVCs from special sound protocols. Let Π_{sps} be any *multi-round* special sound protocol for some relation $\mathcal{R}$, in which the verifier is *algebraic*, that is, the verifier algorithm only checks algebraic equations over the input and the prover messages. E.g., the following naive protocol for the Hadamard product relation over vectors $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{F}^n$ is special-sound and has a degree-2 algebraic verifier: The prover simply sends the vectors $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ to the verifier, and the verifier checks that $\mathbf{a}_i \cdot \mathbf{b}_i = \mathbf{c}_i$ for all $i \in [n]$. However, as shown in the example, the prover message can be large in Π_{sps} and the folding scheme can be expensive if we directly accumulate the verifier predicate. Inspired by the splitting accumulation scheme [BCLMS21], to enable efficient accumulation/folding, we split each

prover message into a short instance and a large opening, where the short instance is built from the homomorphic commitment to the prover message. Next, we use the Fiat-Shamir transform to compile the protocol into a NARK where the verifier challenges are generated from a random oracle.

Now we can view the NARK transcript as an accumulator (or a relaxed NP instance-witness pair in the language of folding schemes), where the accumulator instance consists of the prover message commitments and the verifier challenges; while the accumulator witness consists of the prover messages (i.e., the opening to the commitments). Note we also need to introduce an error vector/commitment into the accumulator witness/instance to absorb the "noise" that arises after each accumulation/folding step.

In the accumulation scheme, given two accumulators (or NARK proofs), the prover folds the witnesses and the instances of both accumulators via a random linear combination and generates a list of d "error-correcting terms" as accumulation proof (d is the degree of the NARK verifier); the verifier only needs to check that the folded accumulator instance is consistent with the accumulation proof and the original instances being folded, both of which are small. After finishing all the accumulation steps, a decider applies a final check to the accumulator, scrutinizing that (i) the accumulator witness is consistent with the commitments in the accumulator instance, and (ii) the "relaxed" NARK verifier check still passes. Here by "relaxed" we mean that the algebraic equation also involves the error vector in the accumulator. If the decider accepts, this implies that all accumulated NARKs were valid and thus that all accumulated statements are in $\mathcal{R}$ (and the prover knows witnesses for these statements).

Finally, given the accumulation scheme, if the relation $\mathcal{R}$ is NP-complete, we can apply the compiler in [BCLMS21] to obtain an efficient IVC scheme with predicates expressed in $\mathcal{R}$.

In Theorem 4, we show that for any $(2k-1)$-move[3] special-sound protocols with degree-d verifiers, the resulting IVC recursive circuit only involves k hashes, $k+1$ non-native field operations and $k+d-1$ commitment group scalar multiplications. For verifiers with high degree $d \gg k$, we introduce a generic approach in Section 3.5 to optimize the IVC complexity, which might be of independent interest. The number of group scalar multiplications in the recursive circuit is only k, with the tradeoff of 1 additional hash and $O(d+\log \ell)$ additional non-native field operations, where ℓ denotes the number of algebraic equations checked in the verifier algorithm (e.g., $\ell = n$ in the Hadamard product example).

Special sound protocols for (non-uniform) Plonkup relations. Given the generic compiler above, our ultimate goal of constructing a (non-uniform) IVC scheme for zkEVM becomes much easier . It is now sufficient to design a multi-round special sound protocol for the (non-uniform) Plonkup relation. Recall that a Plonkup circuit-satisfiability relation consists of three modular relations, namely, (i) a high-degree gate relation checking that

[3] k prover messages, $k-1$ challenges

each custom gate is satisfied; (ii) a permutation (wiring-identity) relation checking that different gate values are consistent if the same wire connects them, and (iii) a lookup relation checking that a subset of gate values belongs to a preprocessed table. The special-sound protocols for the permutation and high-degree gate relations are trivial, where the prover directly sends the witness to the verifier, and the verifier checks that the permutation/high-degree gate relation holds. The degree of the permutation check is only 1, and the degree of the gate-check is the highest degree in the custom gate formula.

The special-sound protocol for the lookup relation $\mathcal{R}_{\text{LK}}$ is more interesting as the statement of the lookup relation is not algebraic. Inspired by the log-derivative lookup scheme [Hab22], in Section 4.3, we design a simple 3-move special sound protocol Π_{LK} for $\mathcal{R}_{\text{LK}}$, in which the verifier degree is only 2. A great feature of Π_{LK} is that the number of non-zero elements in the prover messages is only proportional to the number of lookups, but independent of the table size. Thus the IVC prover complexity for computing the prover message commitments is independent of the table size, which is advantageous when the table size is much larger than the witness size. Moreover, we extend Π_{LK} in Section 4.4 to further support vector-valued lookup, where each table entry is a vector of elements. This feature is useful in applications like zkEVM and for simulating bit operations in circuits.

Given the special sound protocols for permutation/high-degree gate/lookup relations, the special sound protocol Π_{plonkup} for Plonkup is just a parallel composition of the three protocols. Furthermore, in Section 6, we apply a simple trick to support *non-uniform* IVC. More precisely, let $\{\mathcal{C}_i\}_{i=1}^{I}$ be I different branch circuits (e.g., the set of supported instructions in EVM), let $\mathsf{pi} := (pc, \mathsf{pi}')$ be the public input where $pc \in [I]$ is a program counter indicating which instruction/branch circuit is going to be executed in the next IVC step. Our goal is to prove that $(\mathsf{pi}, \mathbf{w})$ is in the relation $\mathcal{R}_{\text{mplkup}}$ in the sense that $\mathcal{C}_{pc}(\mathsf{pi}, \mathbf{w}) = 0$ for witness $\mathbf{w}$. The relation statement can also add additional constraints on pc depending on the applications. The special sound protocol for $\mathcal{R}_{\text{mplkup}}$ is almost identical to Π_{plonkup} for the Plonkup relation, except that the prover further sends a bool vector $\mathbf{b} \in \mathbb{F}^I$, and the verifier uses a degree-$\log(I)$ check to guarantee that $\mathbf{b}_{pc} = 1$ and $\mathbf{b}_i = 0 \forall i \neq pc$. Additionally, each algebraic equation $\mathcal{G}$ checked in Π_{plonkup} is replaced with $\sum_{i=1}^{I} \mathcal{G}_i \cdot \mathbf{b}_i$ where $\mathcal{G}_i$ $(1 \leq i \leq I)$ is the corresponding equation in the protocol for the i-th branch circuit. The resulting special sound protocol has 3 moves, and the verifier degree is $\max(\log(I), d+1)$, where d is the highest degree of the custom gates. This means that the IVC scheme for the non-uniform Plonkup relation adds only negligible overhead to that for the Plonkup relation, as long as the custom gate degree is no less than the logarithm of the number of supported instructions.

2 Preliminaries

2.1 Special Sound Protocols and Fiat-Shamir Transform

We define special-soundness and non-interactive arguments according to the definitions by [AFK22].

Definition 1 (Public-coin interactive proof)**.** *An interactive proof* $\Pi = (\mathsf{P}, \mathsf{V})$ *for relation* $\mathcal{R}$ *is an interactive protocol between two probabilistic machines, a prover* P*, and a polynomial time verifier* V*. Both* P *and* V *take as public input a statement* pi *and, additionally,* P *takes as private input a witness* $\mathbf{w} \in \mathcal{R}(\mathsf{pi})$ *. The verifier* V *outputs* 0 *if it accepts and a non-zero value otherwise. It's output is denoted by* $(\mathsf{P}(\mathbf{w}), \mathsf{V})(\mathsf{pi})$*. Accordingly, we say the corresponding transcript (i.e., the set of all messages exchanged in the protocol execution) is accepting or rejecting. The protocol is* public coin *if the verifier randomness is public. The verifier messages are referred to as challenges.* Π *is a* $(2k-1)$*-move protocol if there are* k *prover messages and* $k-1$ *verifier messages.*

Definition 2 (Tree of transcript)**.** *Let* $\mu \in \mathbb{N}$ *and* $(a_1, \ldots, a_\mu) \in \mathbb{N}^\mu$*. An* $(a_1, \ldots, a_\mu)$*-tree of transcript for a* $(2\mu+1)$*-move public-coin interactive proof* Π *is a set of* $a_1 \cdot a_2 \cdots a_\mu$ *accepting transcripts arranged in a tree of depth* μ *and arity* $a_1, \ldots, a_\mu$ *respectively. The nodes in the tree correspond to the prover messages and the edges to the verifier's challenges. Every internal node at depth* $i-1$ $(1 \leq i \leq \mu)$ *has* a_i *children with distinct challenges. Every transcript corresponds to one path from the root to a leaf node. We simply write the transcripts as an* (a^μ)*-tree of transcript when* $a = a_1 = a_2 = \cdots = a_\mu$*.*

Definition 3 (Special Sound Interactive Protocol)**.** *Let* $\mu, N \in \mathbb{N}$ *and* $(a_1, \ldots, a_\mu) \in \mathbb{N}^\mu$*. A* $(2\mu+1)$*-move public-coin interactive proof* Π *for relation* $\mathcal{R}$ *where the verifier samples its challenges from a set of size* N *is* $(a_1, \ldots, a_\mu)$*-out-of-*N *special-sound if there exists a polynomial time algorithm that, on input* pi *and any* $(a_1, \ldots, a_\mu)$*-tree of transcript for* Π *outputs* $\mathbf{w} \in \mathcal{R}(\mathsf{pi})$*. We simply denote the protocol as an* a^μ*-out-of-*N *(or* a^μ*) special sound protocol if* $a = a_1 = a_2 = \cdots = a_\mu$*.*

Definition 4 (Random-Oracle Non-Interactive Argument of Knowledge (RO-NARK))**.** *A non-interactive random oracle proof for relation* $\mathcal{R}$ *is a pair* (P, V) *of probabilistic random-oracle algorithms, such that: Given* $(\mathsf{pi}, \mathbf{w}) \in \mathcal{R}$ *and access to a random oracle* ρ_{NARK}*, the prover* $\mathsf{P}^{\rho_{\mathsf{NARK}}}(\mathsf{pi}, \mathbf{w})$ *outputs a proof* π*. Given* pi*, a proof* π*, and access to the same random oracle* ρ_{NARK}*, the verifier* $\mathsf{V}^{\rho_{\mathsf{NARK}}}(\mathsf{pi}, \pi)$ *outputs* 0 *to accept or any other value to reject.*

Perfect Completeness: *The* NARK *has perfect completeness if for all* $(\mathsf{pi}, \mathbf{w}) \in \mathcal{R}$

$$P[\mathsf{V}^{\rho_{\mathsf{NARK}}}(\mathsf{pi}, \mathsf{P}^{\rho_{\mathsf{NARK}}}(\mathsf{pi}, \mathbf{w})) = 0] = 1$$

Knowledge Soundness: *The* NARK *has adaptive knowledge-soundness with knowledge error* $\kappa : \mathbb{N} \times \mathbb{N} \to [0,1]$ *if there exists a knowledge extractor* Ext*, with the following properties: The extractor, given input* n*, and oracle-access to any polynomial-time* Q*-query random oracle prover* P^* *that outputs statement of size* n*, runs in an expected polynomial time in* $|\mathsf{pi}| + Q$*, and outputs* $\{(\mathsf{pi}, \pi, \mathsf{aux}, v; \mathbf{w})\}$ *such that a)* $(\mathsf{pi}, \pi, \mathsf{aux}, v)$ *is identically distributed to* $\{(\mathsf{pi}, \pi, \mathsf{aux}, v)\} : (\mathsf{pi}, \pi, \mathsf{aux}) \leftarrow \mathsf{P}^{*,\rho_{\mathsf{NARK}}}, v \leftarrow \mathsf{V}^{\rho_{\mathsf{NARK}}}(\mathsf{pi}, \pi)$ *and b)*

$$\Pr\left[\begin{array}{c}(\mathsf{pi};\mathbf{w}) \in \mathcal{R} \\ \mathsf{V}^{\rho_{\mathsf{NARK}}}(\mathsf{pi},\pi) = 0\end{array} : \{(\mathsf{pi}, \pi, \mathsf{aux}, v; \mathbf{w})\} \leftarrow \mathsf{Ext}^{\mathsf{P}^*}\right] \geq \frac{\epsilon(\mathsf{P}^*) - \kappa(n,Q)}{\mathsf{poly}(n)},$$

where $\epsilon(\mathsf{P}^*)$ *is* P*'s success probability, i.e.* $\epsilon(\mathsf{P}^*) = P[\mathsf{V}^{\rho_{\mathsf{NARK}}}(\mathsf{pi}, \pi) = 0 : (\mathsf{pi}, \pi) \leftarrow \mathsf{P}^{*,\rho_{\mathsf{NARK}}}]$*. Here,* Ext *implements* ρ_{NARK} *for* P^**; in particular, it can arbitrarily program the random oracle.*

Definition 5 (Fiat-Shamir Transform (adaptive))**.** *The Fiat-Shamir transform* $\mathsf{FS}[\Pi] = (\mathsf{P}_{\mathsf{fs}}, \mathsf{V}_{\mathsf{fs}})$ *is a RO-NARK, where* $\mathsf{P}^{\rho_{\mathsf{NARK}}}(\mathsf{pi};\mathbf{w})$ *runs* $\mathsf{P}(\mathsf{pi};\mathbf{w})$ *but instead of receiving challenge* c_i*, on message* m_i*, from the verifier, it computes it as follows:*

$$c_i = \rho_{\mathsf{NARK}}(c_{i-1}, m_i) \tag{1}$$

and $c_0 = \rho_{\mathsf{NARK}}(\mathsf{pi})$*.* $\mathsf{P}_{\mathsf{fs}}^{\rho_{\mathsf{NARK}}}$ *outputs* $\pi = (m_1, \ldots, m_\mu)$*. The verifier* $\mathsf{V}_{\mathsf{fs}}^{\rho_{\mathsf{NARK}}}$ *accepts, if* V *accepts the transcript* $(m_1, c_1, \ldots, m_\mu, c_\mu, m_{\mu+1})$ *for input* pi *and the challenges are computed as per equation* (1)*.*

2.2 Adaptive Fiat-Shamir transform

Lemma 1 (Fiat-Shamir transform of Special Sound Protocols [AFK22])**.** *The Fiat-Shamir transform of a* $(\alpha_1, \ldots, \alpha_\mu)$*-out-of-*$N$ *special-sound interactive proof* Π *is knowledge sound with knowledge error*

$$\kappa_{\mathsf{fs}}(Q) = (Q+1)\kappa$$

where $\kappa = 1 - \prod(1 - \frac{\alpha_i}{N})$ *is the knowledge error of the interactive proof* Π*.*

2.3 Incremental Verifiable Computation (IVC)

We adapt and simplify the definition from [BCLMS21; KST22].

Definition 6 (IVC)**.** *An incremental verifiable computation (IVC) scheme for function predicates expressed in a circuit-satisfiability relation* $\mathcal{R}_{\mathsf{NP}}$ *is a tuple of algorithms* $\mathsf{IVC} = (\mathsf{P}_{\mathsf{IVC}}, \mathsf{V}_{\mathsf{IVC}})$ *with the following syntax and properties:*

- $\mathsf{P}_{\mathsf{IVC}}(m, z_m, \mathbf{w}_{\mathsf{loc}}, z_0, z_{m-1}, \pi_{m-1}]) \to \pi_m$*. The IVC prover* $\mathsf{P}_{\mathsf{IVC}}$ *takes as input an output* z_m *at step* m*, local data* $\mathbf{w}_{\mathsf{loc}}$*, initial input* z_0*, previous output* z_{m-1} *and proof* π_{m-1} *and outputs a new IVC proof* π_m*.*

- $\mathsf{V}_{\mathsf{IVC}}(m, z_0, z_m, \pi_m) \to b$. *The IVC verifier* $\mathsf{V}_{\mathsf{IVC}}$ *takes the initial input* z_0*, the output* z_m *at step* m*, and an IVC proof* π_m*, 'accepts' by outputting* $b = 0$ *and 'rejects' otherwise.*

The scheme IVC *has perfect completeness if for any function predicate* ϕ *expressible in* $\mathcal{R}_{\mathsf{NP}}$*, and any* $(m, z_m, \mathbf{w}_{\mathsf{loc}}, z_0, [z_i, \pi_i]_{i=1}^{m-1})$ *such that*

$$\phi(z_m, \mathbf{w}_{\mathsf{loc}}, z_0, z_{m-1}) \wedge (\mathsf{V}_{\mathsf{IVC}}(i, z_0, z_i, \pi_i) = 0 \forall i \in [m-1])$$

it holds that $\mathsf{V}_{\mathsf{IVC}}(m, z_0, z_m, \pi_m)$ *accepts for proof* $\pi_m \leftarrow \mathsf{P}_{\mathsf{IVC}}(m, z_m, \mathbf{w}_{\mathsf{loc}}, z_0, z_{m-1}, \pi_{m-1})$*.*

The scheme IVC *has knowledge soundness if for every expected polynomial-time adversary* P^**, there exists an expected polynomial-time extractor* $\mathsf{Ext}_{\mathsf{P}^*}$ *such that*

$$\Pr\left[\begin{array}{c} \mathsf{V}_{\mathsf{IVC}}(m, z_0, z, \pi_m) = 0 \wedge \\ ([\exists i \in [m], \neg\phi(z_i, \mathbf{w}_i, z_0, z_{i-1})] \\ \vee z \neq z_m) \end{array} \middle| \begin{array}{c} [\phi, (m, z_0, z, \pi_m)] \leftarrow \mathsf{P}^* \\ [z_i, \mathbf{w}_i]_{i=1}^m \leftarrow \mathsf{Ext}_{\mathsf{P}^*} \end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

Here m *is a constant.*

2.4 Simple Accumulation

We take definitions and proofs from [BCLMS21].

Definition 7 (Accumulation Scheme)**.** *An accumulation scheme for a NARK* $(\mathsf{P}_{NARK}, \mathsf{V}_{NARK})$ *is a triple of algorithms* $\mathsf{acc} = (\mathsf{P}_{\mathsf{acc}}, \mathsf{V}_{\mathsf{acc}}, D)$*, all of which have access to the same random oracle* ρ_{acc} *as well as* ρ_{NARK}*, the oracle for the NARK. The algorithms have the following syntax and properties:*

- $\mathsf{P}_{\mathsf{acc}}(\mathsf{pi}, \pi = (\pi.x, \pi.\mathbf{w}), \mathsf{acc} = (\mathsf{acc}.x, \mathsf{acc}.\mathbf{w})) \to \{\mathsf{acc}' = (\mathsf{acc}'.x, \mathsf{acc}'.\mathbf{w}), \mathsf{pf}\}$*. The accumulation prover* $\mathsf{P}_{\mathsf{acc}}$ *takes as input a statement* pi*, NARK proof* π*, and an accumulator* acc *and outputs a new accumulator* acc′ *and correction terms* pf*.*
- $\mathsf{V}_{\mathsf{acc}}(\mathsf{pi}, \pi.x, \mathsf{acc}.x, \mathsf{acc}'.x, \mathsf{pf}) \to v$*. The accumulation verifier takes as input the statement* pi*, the instances of the NARK proof, the old and new accumulator, and 'accepts' by outputting* 0 *and 'rejects' otherwise.*
- $D(\mathsf{acc}) \to v$*. The decider on input* acc *'accepts' by outputting* 0 *and 'rejects' otherwise.*

An accumulation scheme has knowledge-soundness with knowledge error κ *if the RO-NARK* $(\mathsf{P}', \mathsf{V}')$ *has knowledge error* κ *for the relation*

$$\mathcal{R}_{\mathsf{acc}}((\mathsf{pi}, \pi.x, \mathsf{acc}.x); (\pi.\mathbf{w}, \mathsf{acc}.\mathbf{w})) : (\mathsf{V}_{NARK}(\mathsf{pi}, \pi) = 0 \wedge D(\mathsf{acc}) = 0)\ ,$$

where P' *outputs* $\mathsf{acc}', \mathsf{pf}$ *and* V' *on input* $((\mathsf{pi}, \pi.x, \mathsf{acc}.x), (\mathsf{acc}', \mathsf{pf}))$ *accepts if* $D(\mathsf{acc}')$ *and* $\mathsf{V}_{\mathsf{acc}}(\mathsf{pi}, \pi.x, \mathsf{acc}.x, \mathsf{acc}'.x, \mathsf{pf}) = 0$*.*

The scheme has perfect completeness if the RO-NARK $(\mathsf{P}', \mathsf{V}')$ *has perfect completeness for* $\mathcal{R}_{\mathsf{acc}}$*.*

Theorem 1 (IVC from accumulation[BCLMS21])**.** *Given a standard-model NARK for circuit-satisfiability and a standard-model accumulation scheme (Definition 7) for that NARK, there exists an efficient transformation that outputs an IVC scheme (see Section 3.2 of [BCLMS21]) for constant-depth compliance predicates, assuming that the circuit complexity of the accumulation verifier* $\mathsf{V}_{\mathsf{acc}}$ *is sub-linear in its input.*

Random Oracle. Note that both the NARK and accumulation scheme we construct are in the random oracle model. However, Theorem 1 requires a NARK and an accumulation scheme in the standard model. It remains an open problem to construct such schemes. However, we can heuristically instantiate the random oracle with a cryptographic hash function and assume that the resulting schemes still have knowledge soundness.

Complexity. The IVC transformation from [BCLMS21] recursively proves that the accumulation was performed correctly. To do that, it implements $\mathsf{V}_{\mathsf{acc}}$ as a circuit and proves that the previous accumulation step was done correctly. Note that this recursive circuit is independent of the size of $\pi.\mathbf{w}, \mathsf{acc}.\mathbf{w}$ and the runtime of D. The IVC prover is linear in the size of the recursive circuit plus the size of the IVC computation step expressed as a circuit. The final IVC verifier and the IVC proof size are linear in these components. This can be reduced using an additional SNARK as in [KST22].

PCD. IVC can be generalized to arbitrary DAGs instead of just path graphs in a primitive called proof-carrying data[BCCT13]. Accumulation schemes can be compiled into full PCD if they support accumulating an arbitrary number of accumulators and proofs[BCMS20; BCLMS21]. For simplicity, we only build accumulation for one proof and one accumulator, as well as for two accumulators. This enables PCD for DAGs of degree two. By transforming higher degree graphs into degree two graphs (by converting each degree d node into a $\log_2(d)$ depth tree), we can achieve PCD for these graphs.

Outsourcing the decider In the accumulation to IVC transformation, the IVC proof is linear in the accumulator, and the IVC verifier runs the decider. The accumulation schemes we construct are linear in the witness of a single computation step. However, we can outsource the decider by providing a SNARK that, given $\mathsf{acc}.x$, proves knowledge of $\mathsf{acc}.\mathbf{w}$, such that $D(\mathsf{acc}) = 0$. Nova[KST22] constructs a custom, concretely efficient SNARK for their accumulation/folding scheme.

2.5 Commitment Scheme

Definition 8 (Commitment Scheme)**.** $\mathsf{cm} = (\mathsf{Setup}, \mathsf{Commit})$ *is a binding commitment scheme, consisting of two algorithms:*
$\mathsf{Setup}(1^\lambda) \to \mathsf{ck}$ *takes as input the security parameter and outputs a commitment key* ck*.*
$\mathsf{Commit}(\mathsf{ck}, \mathbf{m} \in \mathcal{M}) \to C \in \mathcal{C}$*, takes as input the commitment key* ck *and a message* $\mathbf{m}$ *in*

$\mathcal{M}$ *and outputs a commitment* $C \in \mathcal{C}$.
The scheme is binding if for all polynomial-time randomized algorithms P^**:*

$$\Pr\left[\begin{array}{c|c} \mathsf{Commit}(\mathsf{ck},\mathbf{m}) = \mathsf{Commit}(\mathsf{ck},\mathbf{m}') & \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda) \\ \wedge & \mathbf{m},\mathbf{m}' \leftarrow \mathsf{P}^*(\mathsf{ck}) \\ \mathbf{m} \neq \mathbf{m}' & \end{array}\right] = \mathsf{negl}(\lambda)$$

Homomorphic commitment. *We say the commitment is homomorphic if* $(\mathcal{C},+)$ *is an additive group of prime order* p.

3 Protocols

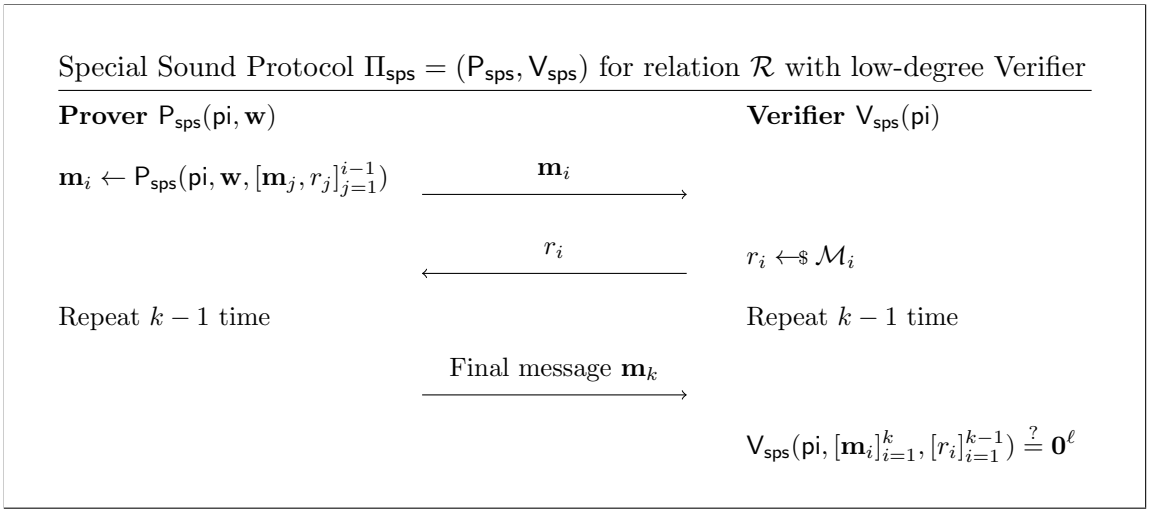
3.1 Special Sound Protocols

In this section, we describe a class of special sound protocols whose verifier is algebraic. The protocol Π_{sps} has 3 essential parameters $k, d, \ell \in \mathbb{N}$, meaning that Π_{sps} is a $(2k-1)$-move protocol with verifier degree d, and the output length of the verifier is ℓ (i.e., the verifier checks ℓ algebraic equations). In each round i ($1 \le i \le k$), the prover $\mathsf{P}_{\mathsf{sps}}(\mathsf{pi}, \mathbf{w}, [\mathbf{m}_j, r_j]_{j=1}^{i-1})$ generates the next message $\mathbf{m}_i$ on input the public input pi, the witness $\mathbf{w}$, and the current transcript $[\mathbf{m}_j, r_j]_{j=1}^{i-1}$, and sends $\mathbf{m}_i$ to the verifier; the verifier replies with a random challenge $r_i \in \mathcal{M}_i$ (the challenge can also be a vector). After the final message $\mathbf{m}_k$, the verifier computes the algebraic map $\mathsf{V}_{\mathsf{sps}}$ and checks that the output is a zero vector of length ℓ. More precisely, $\deg(\mathsf{V}_{\mathsf{sps}}) = d$, s.t.

$$\mathsf{V}_{\mathsf{sps}}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^{k}, [r_i]_{i=1}^{k-1}) := \sum_{j=0}^{d} f_j^{\mathcal{R}}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^{k}, [r_i]_{i=1}^{k-1}),$$

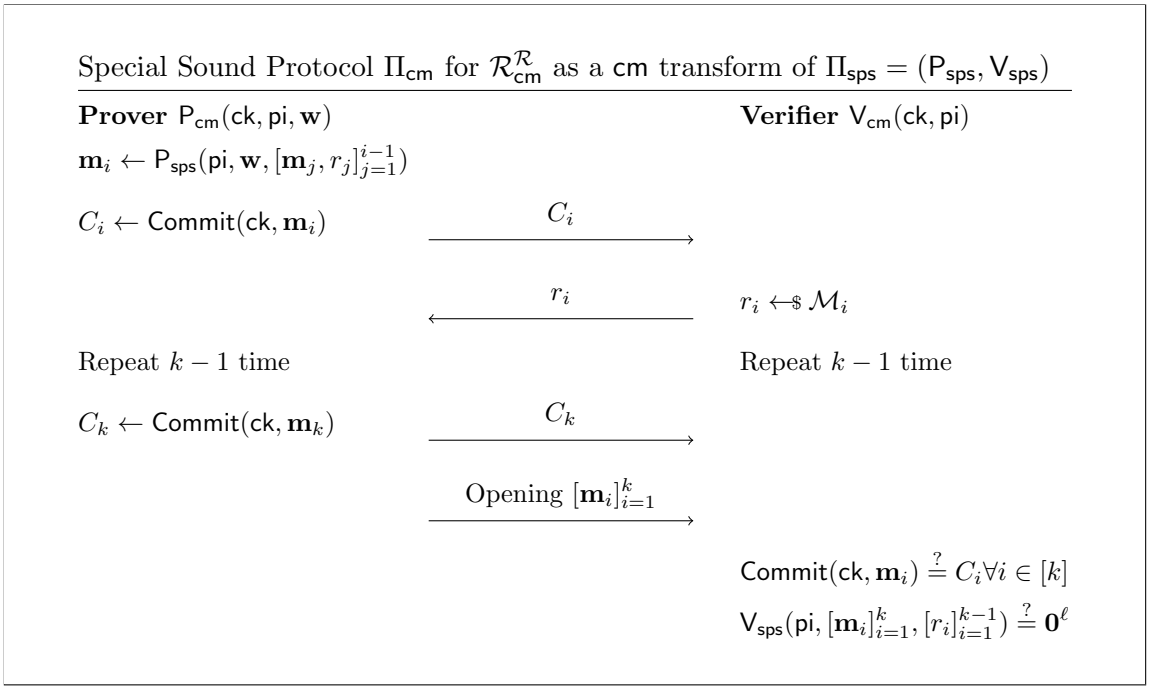
where $f_j^{\mathcal{R}}$ is a homogeneous degree-j algebraic map that outputs a vector of ℓ field elements. The definition of $f_j^{\mathcal{R}}$ depends on the structure of proved relation $\mathcal{R}$.

We describe the special sound protocol Π_{sps} below.



3.2 Commit and Open

For a commitment scheme $\mathsf{cm} = (\mathsf{Setup}, \mathsf{Commit})$, consider the following relation $\mathcal{R}_{\mathsf{cm}}^{\mathcal{R}} = (x; \mathbf{w}, \mathbf{m} \in \mathcal{M}, \mathbf{m}' \in \mathcal{M}) : \{(x, \mathbf{w}) \in \mathcal{R} \vee (\mathsf{Commit}(\mathbf{m}) = \mathsf{Commit}(\mathbf{m}') \wedge \mathbf{m} \neq \mathbf{m}')\}$. The relation's witness is either a valid witness for $\mathcal{R}$ or a break of the commitment scheme cm. We now design a special sound protocol $\Pi_{\mathsf{cm}} = (\mathsf{P}_{\mathsf{cm}}, \mathsf{V}_{\mathsf{cm}})$ for $\mathcal{R}_{\mathsf{cm}}^{\mathcal{R}}$ given $\Pi_{\mathsf{sps}} = (\mathsf{P}_{\mathsf{sps}}, \mathsf{V}_{\mathsf{sps}})$, a special sound protocol for $\mathcal{R}$. P_{cm} runs $\mathsf{P}_{\mathsf{sps}}$ to generate the ith message and then commits to the message. Along with the final message, P_{cm} sends the opening to the commitment. The verifier V_{cm} checks the correctness of the commitments and runs $\mathsf{V}_{\mathsf{sps}}$ on the commitment openings.



Lemma 2 (Π_{cm} is a^{μ}-special-sound)**.** *Let Π_{sps} be an a^{μ} special-sound protocol for relation $\mathcal{R}$, where the prover messages are all in a set $\mathcal{M}$. Let $(\mathsf{Setup}, \mathsf{Commit})$ be a binding commitment scheme for messages in $\mathcal{M}$. For $\mathsf{ck} \leftarrow \mathsf{Setup}_{\mathsf{cm}}(1^{\lambda})$ let $\mathcal{R}_{\mathsf{cm}} = (\mathsf{pi}; \mathbf{w}, m \in \mathcal{M}, m' \in \mathcal{M}) : (\mathsf{pi}; \mathbf{w}) \in \mathcal{R} \vee (\mathsf{Commit}(\mathsf{ck}, m) = \mathsf{Commit}(\mathsf{ck}, m') \wedge m \neq m')$. Then $\Pi_{\mathsf{cm}} = \mathsf{cm}[\Pi_{\mathsf{sps}}]$ is an a^{μ} special sound protocol for $\mathcal{R}^{\mathcal{R}}_{\mathsf{cm}}$.*

Proof. Let $\mathsf{Ext}_{\mathsf{sps}}$ be the extractor for Π_{sps}. We will construct $\mathsf{Ext}_{\mathsf{cm}}$ for Π_{cm} that computes a witness for $\mathcal{R}_{\mathsf{cm}}$, i.e., a witness for $\mathcal{R}$ or a collision for cm given an a^{μ}-transcript tree for Π_{cm}. The extractor $\mathsf{Ext}_{\mathsf{cm}}$ first checks whether there exist two transcripts that have inconsistent final messages. That is, the final message opening is different for the nodes in the intersection of the root-to-leaf paths of these two transcripts. This means we have $\mathbf{m}_i$ and $\mathbf{m}'_i$, such that $\mathsf{Commit}(\mathbf{m}_i) = \mathsf{Commit}(\mathbf{m}'_i)$ and $\mathbf{m}_i \neq \mathbf{m}'_i$. This is a break for cm, i.e., a valid witness for $\mathcal{R}_{\mathsf{cm}}$. Otherwise $\mathsf{Ext}_{\mathsf{cm}}$ builds a transcript tree for Π_{sps} by replacing all commitments with the openings and use $\mathsf{Ext}_{\mathsf{sps}}$ to compute $\mathbf{w} \in \mathcal{R}(\mathsf{pi})$, such that $(\mathbf{w}, \perp, \perp) \in \mathcal{R}_{\mathsf{cm}}(\mathsf{pi})$. □

3.3 Fiat-Shamir transform

Let ρ_{NARK} be a random oracle. Let Π_{cm} be the commit-and-open protocol for the special sound protocol $\Pi_{\mathsf{sps}} = (\mathsf{P}_{\mathsf{sps}}, \mathsf{V}_{\mathsf{sps}})$. The Fiat-Shamir Transform of the protocol Π_{cm} is the following. By Lemma 1, $\mathsf{FS}[\Pi_{\mathsf{cm}}]$ is knowledge sound if Π_{sps} is special-sound.

Fiat-Shamir Transform FS of Special Sound Protocol Π for relation $\mathcal{R}_{\mathsf{cm}}^{\mathcal{R}}$: $\mathsf{FS}[\Pi_{\mathsf{cm}}]$

Prover $\mathsf{P}_{\mathbf{NARK}}^{\rho_{\mathbf{NARK}}}(\mathsf{ck}, \mathsf{pi}, \mathbf{w})$		**Verifier** $\mathsf{V}_{\mathbf{NARK}}^{\rho_{\mathbf{NARK}}}(\mathsf{ck}, \mathsf{pi})$
$r_0 \leftarrow \rho_{\mathsf{NARK}}(\mathsf{pi})$		
For $i \in [k-1]$:		
$\mathbf{m}_i \leftarrow \mathsf{P}_{\mathsf{sps}}(\mathsf{pi}, \mathbf{w}, [\mathbf{m}_j, r_j]_{j=1}^{i-1})$		
$C_i \leftarrow \mathsf{Commit}(\mathsf{ck}, \mathbf{m}_i)$		
$r_i \leftarrow \rho_{\mathsf{NARK}}(r_{i-1}, C_i)$		
$\mathbf{m}_k \leftarrow \mathsf{P}_{\mathsf{sps}}(\mathsf{pi}, \mathbf{w}, [\mathbf{m}_j, r_j]_{j=1}^{k-1})$		
$C_k \leftarrow \mathsf{Commit}(\mathsf{ck}, \mathbf{m}_i)$	$\pi.x = [C_i]_{i=1}^k \longrightarrow$	
	$\pi.\mathbf{w} = [\mathbf{m}_i]_{i=1}^k \longrightarrow$	
		$r_0 \leftarrow \rho_{\mathsf{NARK}}(x)$
		$r_i \leftarrow \rho_{\mathsf{NARK}}(r_{i-1}, C_i) \forall i \in [k-1]$
		$\mathsf{Commit}(\mathsf{ck}, \mathbf{m}_i) \stackrel{?}{=} C_i \forall i \in [k]$
		$\mathsf{V}_{\mathsf{sps}}(\mathsf{pi}, \pi.x, \pi.\mathbf{w}, [r_i]_{i=1}^{k-1}) \stackrel{?}{=} \mathbf{0}^{\ell}$

Remark 2. *We slightly abuse the notation of $r_i \leftarrow \rho_{NARK}(r_{i-1}, C_i)$ as r_i are in different message spaces $\mathcal{M}_i$. Fortunately, we can assume that the cardinality of each $\mathcal{M}_i$ is the same (say M). Let $k \in [M]$ be the output of $\rho_{NARK}(r_{i-1}, C_i)$. r_i is denoted as the k-th element in $\mathcal{M}_i$. In practice, r_i is usually a deterministic function g of a random element r_i' in $\mathbb{F}$, so we can set $r_i \leftarrow g(r_i' := \rho_{NARK}(r_{i-1}', C_i))$ given a hash function ρ_{NARK} that outputs a field element.*

3.4 Accumulation Scheme for $\mathsf{V}_{\mathbf{NARK}}$

Let ρ_{acc} and ρ_{NARK} be two random oracles, and let $\mathsf{V}_{\mathsf{NARK}}$ be the verifier in Section 3.3, whose underlying special sound protocol is $\Pi_{\mathsf{sps}} = (\mathsf{P}_{\mathsf{sps}}, \mathsf{V}_{\mathsf{sps}})$ for a relation $\mathcal{R}$. We describe the accumulation scheme for $\mathsf{V}_{\mathsf{NARK}}$.

The accumulated predicate. The predicate to be accumulated is the "relaxed" verifier check of the NARK scheme $\mathsf{FS}[\Pi]$ for relation $\mathcal{R}$. Namely, given public input $\mathsf{pi} \in \mathcal{M}^{\ell_{\mathsf{in}}}$, random challenges $[r_i]_{i=1}^{k-1} \in \mathcal{M}_1 \times \cdots \times \mathcal{M}_{k-1}$, a NARK proof

$$\pi.x = [C_i]_{i=1}^k,\ \pi.\mathbf{w} = [\mathbf{m}_i]_{i=1}^k$$

where $[C_i]_{i=1}^k \in \mathcal{C}^k$ are commitments and $[\mathbf{m}_i]_{i=1}^k$ are prover messages in the special sound protocol Π_{sps}, the predicate checks that (i) $r_i = \rho_{\mathsf{NARK}}(r_{i-1}, C_i)$ for all $i \in [k-1]$ (where

$r_0 := \rho_{\mathsf{NARK}}(\mathsf{pi})$), (ii) $\mathsf{Commit}(\mathsf{ck}, \mathbf{m}_i) = C_i$ for all $i \in [k]$, and (iii)

$$\mathsf{V}_{\mathsf{sps}}(\mathsf{pi}, \pi.x, \pi.\mathbf{w}, [r_i]_{i=1}^{k-1}) := \sum_{j=0}^{d} \mu^{d-j} \cdot f_j^{\mathcal{R}}(\mathsf{pi}, \pi.\mathbf{w}, [r_i]_{i=1}^{k-1}) = \mathbf{e}$$

where $\mathbf{e} = \mathbf{0}^{\ell}$ and $\mu = 1$ for the NARK verifier $\mathsf{V}_{\mathsf{NARK}}$. Here $f_j^{\mathcal{R}}$ is a degree-j homogeneous algebraic map that outputs ℓ field elements. Degree-j homogeneity says that each monomial term of $f_j^{\mathcal{R}}$ has degree exactly j.

Remark 3. *Without loss of generality, we assume that the public input* pi *is of constant size, as otherwise, we can set it as the hash of the original public input.*

Accumulator. The accumulator has the following format:

- *Accumulator instance* $\mathsf{acc}.x := \{\mathsf{pi}, [C_i]_{i=1}^{k}, [r_i]_{i=1}^{k-1}, E, \mu\}$, where $\mathsf{pi} \in \mathcal{M}^{\ell_{\mathsf{in}}}$ is the accumulated public input, $[C_i]_{i=1}^{k} \in \mathcal{C}^k$ are the accumulated commitments, $[r_i]_{i=1}^{k-1} \in \mathcal{M}_1 \times \cdots \times \mathcal{M}_{k-1}$ are the accumulated challenges, $E \in \mathcal{C}$ is the accumulated commitment to the error terms, and $\mu \in \mathbb{F}$ is a slack variable.
- *Accumulator witness* $\mathsf{acc}.\mathbf{w} := \{[\mathbf{m}_i]_{i=1}^{k}\}$, where $[\mathbf{m}_i]_{i=1}^{k}$ are the accumulated prover messages.

Accumulation prover. On input commitment key ck (which can be hardwired in the prover's algorithm), accumulator acc, an instance-proof pair (pi, π) where

$$\begin{aligned} \mathsf{acc} &:= (\mathsf{acc}.x = \{\mathsf{pi}', [C_i']_{i=1}^{k}, [r_i']_{i=1}^{k-1}, E, \mu\}, \mathsf{acc}.\mathbf{w} = \{[\mathbf{m}_i']_{i=1}^{k}\}), \\ \pi &:= (\pi.x = [C_i]_{i=1}^{k}, \pi.\mathbf{w} = [\mathbf{m}_i]_{i=1}^{k}), \end{aligned}$$

the accumulation prover $\mathsf{P}_{\mathsf{acc}}$ works as in Figure 1.

Accumulation verifier. On input public input pi, NARK proof instance $\pi.x$, accumulator instance $\mathsf{acc}.x$, accumulation proof pf, and the updated accumulator instance $\mathsf{acc}'.x := \{\mathsf{pi}'', [C_i'']_{i=1}^{k}, [r_i'']_{i=1}^{k}, E', \mu'\}$, the accumulation verifier $\mathsf{V}_{\mathsf{acc}}$ works as in Figure 2.

Decider. On input the commitment key ck (which can be hardwired) and an accumulator

$$\mathsf{acc} = (\mathsf{acc}.x = \{\mathsf{pi}, [C_i]_{i=1}^{k}, [r_i]_{i=1}^{k-1}, E, \mu\}, \mathsf{acc}.\mathbf{w} = \{[\mathbf{m}_i]_{i=1}^{k}\}),$$

the decider does the checks described in Figure 3.

$\mathsf{P}_{\mathsf{acc}}^{\rho_{\mathsf{acc}},\rho_{\mathsf{NARK}}}(\mathsf{ck},\mathsf{acc},(\mathsf{pi},\pi))$

1. $r_i \leftarrow \rho_{\mathsf{NARK}}(r_{i-1}, C_i) \forall i \in [k-1]$ where $r_0 := \rho_{\mathsf{NARK}}(\mathsf{pi})$.
2. Compute $[\mathbf{e}_j]_{j=1}^{d-1} \in (\mathbb{F}^\ell)^{d-1}$, such that

$$\begin{aligned}
&\sum_{j=0}^{d}(X+\mu)^{d-j}\cdot f_j^{\mathcal{R}}(X\cdot \mathsf{pi}+\mathsf{pi}', [X\cdot \mathbf{m}_i+\mathbf{m}_i']_{i=1}^{k}, [X\cdot r_i + r_i']_{i=1}^{k-1}) \\
=&\sum_{j=0}^{d}\mu^{d-j} f_j^{\mathcal{R}}(\mathsf{pi}', [\mathbf{m}_i']_{i=1}^{k}, [r_i']_{i=1}^{k-1}) + X^d \cdot \mathsf{V}_{\mathsf{NARK}}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^{k}, [r_i]_{i=1}^{k-1}) + \sum_{j=1}^{d-1}\mathbf{e}_j X^j \\
=&\mathbf{e} + \sum_{j=1}^{d-1}\mathbf{e}_j X^j
\end{aligned}$$

3. $E_j \leftarrow \mathsf{Commit}(\mathsf{ck}, \mathbf{e}_j) \forall j \in [d-1]$
4. $\alpha \leftarrow \rho_{\mathsf{acc}}(\mathsf{acc}.x, \mathsf{pi}, \pi.x, [E_j]_{j=1}^{d-1}) \in \mathbb{F}$
5. Set vectors

$$\mathbf{v} := \left(1, \mathsf{pi}, [r_i]_{i=1}^{k-1}, [C_i]_{i=1}^{k}, [\mathbf{m}_i]_{i=1}^{k}\right),\ \mathbf{v}' := \left(\mu, \mathsf{pi}', [r_i']_{i=1}^{k-1}, [C_i']_{i=1}^{k}, [\mathbf{m}_i']_{i=1}^{k}\right).$$

6. $\mathbf{v}'' := \left(\mu', \mathsf{pi}'', [r_i'']_{i=1}^{k-1}, [C_i'']_{i=1}^{k}, [\mathbf{m}_i'']_{i=1}^{k}\right) \leftarrow \alpha \cdot \mathbf{v} + \mathbf{v}'$.
7. $E' \leftarrow E + \sum_{j=1}^{d-1} \alpha^j \cdot E_j$.
8. Set $\mathsf{acc}'.x := \{\mathsf{pi}'', [C_i'']_{i=1}^{k}, [r_i'']_{i=1}^{k}, E', \mu'\}$, $\mathsf{acc}'.\mathbf{w} := \{[\mathbf{m}_i'']_{i=1}^{k}\}$.
9. Set accumulation proof $\mathsf{pf} := [E_j]_{j=1}^{d-1}$

Figure 1: Accumulation Prover for low-degree Fiat-Shamired NARKs

$\mathsf{V}_{\mathsf{acc}}^{\rho_{\mathsf{acc}},\rho_{\mathsf{NARK}}}(\mathsf{pi}, \pi.x = [C_i]_{i=1}^{k}, \mathsf{acc}.x = (\mathsf{pi}', [C_i']_{i=1}^{k}, [r_i']_{i=1}^{k-1}, E, \mu), \mathsf{pf} = [E_j]_{j=1}^{d-1}, \mathsf{acc}'.x)$

1. $r_i \leftarrow \rho_{\mathsf{NARK}}(r_{i-1}, C_i) \forall i \in [k-1]$ where $r_0 := \rho_{\mathsf{NARK}}(\mathsf{pi})$.
2. $\alpha \leftarrow \rho_{\mathsf{acc}}(\mathsf{acc}.x, \mathsf{pi}, \pi.x, \mathsf{pf})$
3. Set vectors

$$\mathbf{v} := \left(1, \mathsf{pi}, [r_i]_{i=1}^{k-1}, [C_i]_{i=1}^{k}\right),\ \mathbf{v}' := \mathsf{acc}.x.\left(\mu, \mathsf{pi}', [r_i']_{i=1}^{k-1}, [C_i']_{i=1}^{k}\right).$$

4. Check $\mathsf{acc}'.x.\left(\mu', \mathsf{pi}'', [r_i'']_{i=1}^{k-1}, [C_i'']_{i=1}^{k}\right) \stackrel{?}{=} \alpha \cdot \mathbf{v} + \mathbf{v}'$.
5. Check $\mathsf{acc}'.x.E' \stackrel{?}{=} \mathsf{acc}.x.E + \sum_{j=1}^{d-1} \alpha^j \cdot E_j$.

Figure 2: Accumulation Verifier for low-degree Fiat-Shamired NARKs

$D_{\mathsf{acc}}(\mathsf{acc} = (\mathsf{acc}.x = \{\mathsf{pi}, [C_i]_{i=1}^k, [r_i]_{i=1}^{k-1}, E, \mu\}, \mathsf{acc}.\mathbf{w} = \{[\mathbf{m}_i]_{i=1}^k\}))$

1. $C_i \stackrel{?}{=} \mathsf{Commit}(\mathsf{ck}, \mathbf{m}_i)$ for all $i \in [k]$.
2. $\mathbf{e} \leftarrow \sum_{j=0}^{d} \mu^{d-j} f_j^{\mathcal{R}}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^k, [r_i]_{i=1}^{k-1})$ where $f_j^{\mathcal{R}}$ is the degree-j homogeneous algebraic map described in the accumulated predicate.
3. $E \stackrel{?}{=} \mathsf{Commit}(\mathsf{ck}, \mathbf{e})$.

Figure 3: Accumulation Decider for low-degree Fiat-Shamired NARKs

Remark 4. *The accumulation scheme for $\mathsf{V}_{\mathsf{NARK}}$ is also naturally a folding scheme as defined in Nova [KST22], where we can view an accumulator as a* relaxed *NP instance with error terms. A NARK proof π is an accumulator with $\mu = 1$ and $E = 0 \in \mathbb{G}$. We can use the same accumulation scheme to fold two accumulators $(\mathsf{acc}, \mathsf{acc}')$ into a new accumulator acc''. The scheme is identical to the one presented above but with non-trivial $\mu, \mathbf{e}, E$ terms for acc. The verifier performs one additional group scalar multiplication. In the language of folding schemes, we can fold two NARK instances into an accumulator; or fold a NARK instance and an accumulator into an updated accumulator; or fold two accumulators into an updated accumulator.*

Complexity. Let Π_{sps} be a $(2k-1)$-move special sound protocol with degree-d verifier whose output length is ℓ. Denote by $|M|$ the number of elements in prover messages and $|M^*|$ the number of non-zero elements in the prover messages. Assume that pi is a hash with length 1 (this saves the call $r_0 := \rho_{\mathsf{NARK}}(\mathsf{pi})$), and let $|R|$ be the number of elements in verifier's challenges. We analyze the computational complexity of the accumulation scheme:

- The *accumulation prover*
  - asks $k-1$ queries to ρ_{NARK} and 1 query to ρ_{acc};
  - computes $E_j = \mathsf{Commit}(\mathsf{ck}, \mathbf{e}_j)$ for all $j \in [d-1]$, where $\mathbf{e}_j \in \mathbb{F}^\ell$;
  - performs $|R| + |M^*| + 2$ $\mathbb{F}$-ops to combine $(\mu, \mathsf{pi}, [r_i]_{i=1}^{k-1}, [\mathbf{m}_i]_{i=1}^k)$;
  - performs k $\mathbb{G}$-ops to combine $[C_i]_{i=1}^k$;
  - computes the coefficient forms of ℓ degree-d polynomials for $[\mathbf{e}_j]_{j=1}^{d-1}$.
- The *accumulation verifier* performs
  - asks $k-1$ queries to ρ_{NARK} and 1 query to ρ_{acc};
  - $|R| + 2$ $\mathbb{F}$-ops to combine $(\mu, \mathsf{pi}, [r_i]_{i=1}^{k-1})$;
  - k $\mathbb{G}$-ops to combine $[C_i]_{i=1}^k$;
  - $d-1$ $\mathbb{G}$-ops to add $[E_j]_{j=1}^{d-1}$ onto E.
- The *decider*

– computes $C_i = \mathsf{Commit}(\mathsf{ck}, \mathbf{m}_i)$ for $i \in [k]$ and $E = \mathsf{Commit}(\mathsf{ck}, \mathbf{e})$, with total complexity around $|M| + \ell$ $\mathbb{G}$-ops.
– evaluate ℓ degree-d multivariate polynomials to compute vector $\mathbf{e}$.

Theorem 2. *Let* $(\mathsf{P}_{NARK}, \mathsf{V}_{NARK})$ *be the RO-NARK defined in Section 3.3. The accumulation scheme* $(\mathsf{P}_{\mathsf{acc}}, \mathsf{V}_{\mathsf{acc}}, D_{\mathsf{acc}})$ *for* V_{NARK} *satisfies perfect completeness as defined in Definition 7.*

Proof. Consider any tuple $((\mathsf{pi}, \pi), \mathsf{acc}) \in \mathcal{R}_{\mathsf{acc}}$, that is, $\mathsf{V}_{\mathsf{NARK}}(\mathsf{pi}, \pi)$ and $D(\mathsf{acc})$ both accept. Let $(\mathsf{acc}', \mathsf{pf})$ denote the output of the accumulation prover $\mathsf{P}_{\mathsf{acc}}(\mathsf{ck}, \mathsf{acc}, (\mathsf{pi}, \pi))$. We argue that both the decider $D(\mathsf{acc}')$ and the accumulation verifier $\mathsf{V}_{\mathsf{acc}}(\mathsf{pi}, \pi.x, \mathsf{acc}.x, \mathsf{pf}, \mathsf{acc}'.x)$ will accept, which finishes the proof of perfect completeness by Definition 7.

$\mathsf{V}_{\mathsf{acc}}$ accepts as $\mathsf{P}_{\mathsf{acc}}$ and $\mathsf{V}_{\mathsf{acc}}$ go through the same process of computing challenges $[r_i]_{i=1}^{k-1}$ and α, thus the linear combinations of $\mathsf{acc}.x$ and $(\mathsf{pi}, \pi.x; \mathsf{pf}, [r_i]_{i=1}^{k-1})$ via α will be consistent.

We prove that $D(\mathsf{acc}')$ accepts by scrutinizing the following decider checks.

The check $\mathsf{acc}'.C_i \stackrel{?}{=} \mathsf{Commit}(\mathsf{ck}, \mathsf{acc}'.\mathbf{m}_i)$ succeeds for all $i \in [k]$. This is because

$$\mathsf{acc}'.\{C_i, \mathbf{m}_i\} = \mathsf{acc}.\{C_i, \mathbf{m}_i\} + \alpha \cdot \pi.\{C_i, \mathbf{m}_i\}$$

for all $i \in [k]$, where $\pi.C_i = \mathsf{Commit}(\mathsf{ck}, \pi.\mathbf{m}_i)$ because $\mathsf{V}_{\mathsf{NARK}}(\mathsf{pi}, \pi)$ accepts, and $\mathsf{acc}.C_i = \mathsf{Commit}(\mathsf{ck}, \mathsf{acc}.\mathbf{m}_i)$ because $D(\mathsf{acc})$ accepts. Thus the check succeeds by the homomorphism of the commitment scheme.

The decider computes $\mathbf{e}' \leftarrow \sum_{j=0}^{d} (\mathsf{acc}'.\mu)^{d-j} f_j^{\mathcal{R}}(\mathsf{acc}'.\{\mathsf{pi}, [\mathbf{m}_i]_{i=1}^{k}, [r_i]_{i=1}^{k-1}\})$ such that for $\mathbf{e} = \sum_{j=0}^{d} \mathsf{acc}.\mu^{(d-j)} \cdot f_j^{\mathcal{R}}(\mathsf{acc}.\{\mathsf{pi}, [\mathbf{m}_i]_{i=1}^{k}, [r_i]_{i=1}^{k-1}\})$, it holds that

$$\begin{aligned}
\mathbf{e}' &= \mathbf{e} + \sum_{j=1}^{d-1} \alpha^j \cdot \mathsf{pf}.\mathbf{e}_j \\
&= \sum_{j=0}^{d} (\alpha + \mathsf{acc}.\mu)^{d-j} \cdot f_j^{\mathcal{R}}(\alpha \cdot \{\mathsf{pi}, \pi.[\mathbf{m}_i]_{i=1}^{k}, [r_i]_{i=1}^{k-1}\} + \mathsf{acc}.\{\mathsf{pi}, [\mathbf{m}_i]_{i=1}^{k}, [r_i]_{i=1}^{k-1}\}) .
\end{aligned}$$

By the definition of $\mathsf{pf}.\mathbf{e}_j$ and the homomorphism of the commitment scheme, and because $D(\mathsf{acc})$ accepts and checks $E = \mathsf{Commit}(\mathsf{ck}, \mathbf{e})$, we have that $E' = \mathsf{Commit}(\mathsf{ck}, \mathbf{e}')$. □

Theorem 3. *Let* $(\mathsf{P}_{NARK}, \mathsf{V}_{NARK})$ *be the RO-NARK defined in Section 3.3. Let* $\mathsf{cm} = (\mathsf{Setup}, \mathsf{Commit})$ *be a binding, homomorphic commitment scheme. Let* ρ_{acc} *be another random oracle. The accumulation scheme for* V_{NARK} *has knowledge error* $(Q+1)\frac{d+1}{|\mathbb{F}|} + \mathsf{negl}(\lambda)$ *against any randomized polynomial-time* Q*-query adversary.*

Proof. We show that the scheme is secure by showing that there exists an underlying $(d+1)$-special-sound protocol and then applying the Fiat-Shamir transform to show that

the accumulation scheme is knowledge sound. Consider the public-coin interactive protocol $\Pi_I = (\mathsf{P}_I(\mathsf{pi}, \pi, \mathsf{acc}), \mathsf{V}_I(\mathsf{pi}, \pi.x, \mathsf{acc}.x))$ where P_I sends $\mathsf{pf} = [E_j]_{j=1}^{d-1} \in \mathbb{G}^{d-1}$ as computed by $\mathsf{P}_{\mathsf{acc}}$ to V_I. The verifier sends a random challenge $\alpha \in \mathbb{F}$, and the prover P_I responds with acc' as computed by $\mathsf{P}_{\mathsf{acc}}$. V_I accepts if $D_{\mathsf{acc}}(\mathsf{acc}') = 0$ and $\mathsf{V}_{\mathsf{acc}}(\mathsf{pi}, \pi.x, \mathsf{acc}.x, \mathsf{pf}, \mathsf{acc}'.x) = 0$ using the random challenge α, instead of a Fiat-shamir challenge.

Claim 1: Π_I is $(d+1)$-special-sound Consider the relation $\mathcal{R}_{\mathsf{acc}}$ where $\mathcal{R}_{\mathsf{acc}}$ is defined in Definition 7. Consider $d+1$ accepting transcripts for Π_I :

$$\{\mathcal{T}_i := (\mathsf{pi}, \pi.x, \mathsf{acc}.x; \mathsf{acc}'_i, \mathsf{pf}_i)\}_{i=1}^{d+1} .$$

We construct an extractor $\mathsf{Ext}_{\mathsf{acc}}$ that extracts a witness for $\mathcal{R}_{\mathsf{acc}}(\mathsf{pi}.\pi.x, \mathsf{acc}.x)$ given $\mathcal{T}$.

For all $i \in [d+1]$,

$$(\mathsf{acc}'_i) = (\mu'_i, \mathsf{pi}'_i, [C'_{i,j}]_{j=1}^{k}, [r_{i,j}]_{j=1}^{k-1}, E'_i, [\mathbf{m}'_{i,j}]_{j=1}^{k})$$

and $\mathsf{pf}_i = \mathsf{pf} = [E_j]_{j=1}^{d-1}$.

Given that the transcripts are accepting, i.e. both $\mathsf{V}_{\mathsf{acc}}$ and D_{acc} accept, we have that $\mathsf{Commit}(\mathsf{ck}, \mathbf{e}'_i) = E'_i = \mathsf{acc}.E + \sum_{j=1}^{d-1} \alpha_i^j E_j$ for all $i \in [d+1]$, whereas

$$\mathbf{e}'_i := \sum_{j=0}^{d} {\mu'_i}^{d-j} f_j^{\mathcal{R}}(\pi'_i, [\mathbf{m}'_{i,j}]_{j=1}^{k}, [r_{i,j}]_{j=1}^{k-1}) .$$

Using a Vandermonde matrix of the challenges $\alpha_1, \ldots, \alpha_d$ we can compute $\mathbf{e}, [\mathbf{e}_j]_{j=1}^{d-1}$ such that $E_j = \mathsf{Commit}(\mathsf{ck}, \mathbf{e}_j)$ and $\mathsf{acc}.E = \mathsf{Commit}(\mathsf{ck}, \mathbf{e})$ from the equations above. Therefore we have that $\mathbf{e}'_i = \mathbf{e} + \sum_{j=1}^{d-1} \alpha_i^j \mathbf{e}_j$ for all $i \in [d+1]$.

Additionally using two challenges (α_1, α_2), $\mathsf{Ext}_{\mathsf{acc}}$ can compute $\pi.\mathbf{w} = [\mathbf{m}_j]_{j=1}^{k} = [\frac{\mathsf{acc}'.\mathbf{m}_{1,j} - \mathsf{acc}'.\mathbf{m}_{2,j}}{\alpha_1 - \alpha_2}]_{j=1}^{k}$. It holds that $\mathsf{acc}.\mathbf{m}_j = \mathsf{acc}'.\mathbf{m}_{1,j} - \alpha_1 \cdot \pi.\mathbf{m}_j \forall j \in [k]$, such that $\pi.C_j = \mathsf{Commit}(\mathsf{ck}, \pi.\mathbf{m}_j)$ and $\mathsf{acc}.C_j = \mathsf{Commit}(\mathsf{ck}, \mathsf{acc}.\mathbf{m}_j)$. If for any other challenge and any j, $\mathsf{acc}'.\mathbf{m}_j \neq \alpha\pi.\mathbf{m}_j + \mathsf{acc}.\mathbf{m}_j$, then this can be used to compute a break of the commitment scheme cm. This happens with negligible probability by assumption.

Otherwise, we have that $\sum_{j=0} \mu_i^{d-j} f_j^{\mathcal{R}}(\pi_j, [\mathbf{m}_{i,j}]_{i=1}^{k}, [r_{i,j}]_{i=1}^{k-1}) - \mathbf{e}_i = 0$ for all $i \in [d+1]$. Together this implies that the degree d polynomial

$$\begin{aligned} p(X) = &\sum_{j=0}^{d} (X + \mathsf{acc}.\mu)^{d-j} \cdot f_j^{\mathcal{R}}(X \cdot \mathsf{pi} + \mathsf{acc}.\mathsf{pi}, [X \cdot \mathbf{m}_i + \mathsf{acc}.\mathbf{m}_i]_{i=1}^{k}, [X \cdot r_i + \mathsf{acc}.r_i]_{i=1}^{k-1}) \\ &- \mathbf{e} - \sum_{j=1}^{d-1} \mathbf{e}_j X^j , \end{aligned} \tag{2}$$

is zero on $d+1$ points $(\alpha_1, \ldots, \alpha_{d+1})$, i.e. is zero everywhere. The constant term of this polynomial is

$$\sum_{j=0}^{d} \mathsf{acc}.\mu^{d-j} \cdot f_j^{\mathcal{R}}(\mathsf{acc}.\mathsf{pi}, [\mathsf{acc}.\mathbf{m}_i]_{i=1}^{k}, [\mathsf{acc}.r_i]_{i=1}^{k-1}) - \mathbf{e}\,.$$

It being 0 implies that $D(\mathsf{acc}) = 0$. Additionally, the degree d term of the polynomial is

$$\sum_{j=0}^{d} f_j^{\mathcal{R}}(\mathsf{pi}, [\pi.\mathbf{m}_i]_{i=1}^{k}, [\pi.r_i]_{i=1}^{k-1})\,.$$

Together with $\mathsf{V}_{\mathsf{acc}}$ checking that the challenges r_i are computed correctly this implies that $\mathsf{V}_{\mathsf{NARK}}(\mathsf{pi}, \pi) = 0$. Ext thus outputs a valid witness $(\pi.\mathbf{w}, \mathsf{acc}.\mathbf{w}) \in \mathcal{R}_{\mathsf{acc}}(\mathsf{pi}, \pi.x, \mathsf{acc}.x)$ and thus Π_I is $(d+1)$-special sound. Using Lemma 1, we have that $\Pi_{AS} = \mathsf{FS}[\Pi_I]$ is a NARK for $\mathcal{R}_{\mathsf{acc}}$ with knowledge soundness $Q \cdot \frac{d+1}{|\mathbb{F}|} + \mathsf{negl}(\lambda)$. This implies that acc is an accumulation scheme with $(Q \cdot \frac{d+1}{|\mathbb{F}|} + \mathsf{negl}(\lambda))$-knowledge soundness. □

3.5 Efficiency optimizations for high-degree verifiers

Observe that the accumulation prover needs to perform $\Omega(d\ell)$ group operations to commit to the $d-1$ error vectors $\mathbf{e}_j \in \mathbb{F}^\ell$ $(1 \le j < d)$; and the accumulation verifier needs to check the combination of d error vector commitments. This can be a bottleneck when the verifier degree d is high. In this circumstance, we can optimize the accumulation complexity by transforming the underlying special sound protocol Π_{sps} into a new special sound protocol $\mathsf{OPT}_{d^*}(\Pi_{\mathsf{sps}})$ for the same relation $\mathcal{R}$. This optimization reduces the verifier's output length to ℓ/d^* with verifier degree $d + \log(d^*)$. We describe the generic transformation below.

Generic transform to a verifier with smaller outputs. For a tunable parameter $d^* \le \ell$, we can transform Π_{sps} into a special sound protocol $\mathsf{OPT}_{d^*}(\Pi_{\mathsf{sps}})$ where the output vector length in $\mathsf{V}_{\mathsf{sps}}$ reduces from ℓ to ℓ/d^*. Essentially, instead of checking the output of $\mathsf{V}_{\mathsf{sps}}$ to be ℓ zeroes, we split the output vector of $\mathsf{V}_{\mathsf{sps}}$ into d^* chunks, and check that a random linear combination of the d^* chunks equals to a zero vector of length ℓ/d^*. For example, if the map is $\mathsf{V}_{\mathsf{sps}}(x_1, x_2) := (f_1(x_1, x_2), f_2(x_1, x_2)) = (x_1 + x_2, x_1 x_2)$ we can set the new algebraic map as $\mathsf{V}'_{\mathsf{sps}}(x_1, x_2, r) := f_1(x_1, x_2) + r \cdot f_2(x_1, x_2) = (x_1 + x_2) + r x_1 x_2$ for a random r. The output length becomes 1 while the degree of the map increases by 1.

To minimize degree increase without introducing many new random challenges, we define function eq^* where for $\mathbf{a} \in \mathbb{F}^\kappa$ and $\mathbf{b} \in \{0,1\}^\kappa$,

$$eq^*(\mathbf{a}, \mathbf{b}) := \prod_{i=1}^{\kappa} \mathbf{a}_i^{\mathbf{b}_i}\,. \tag{3}$$

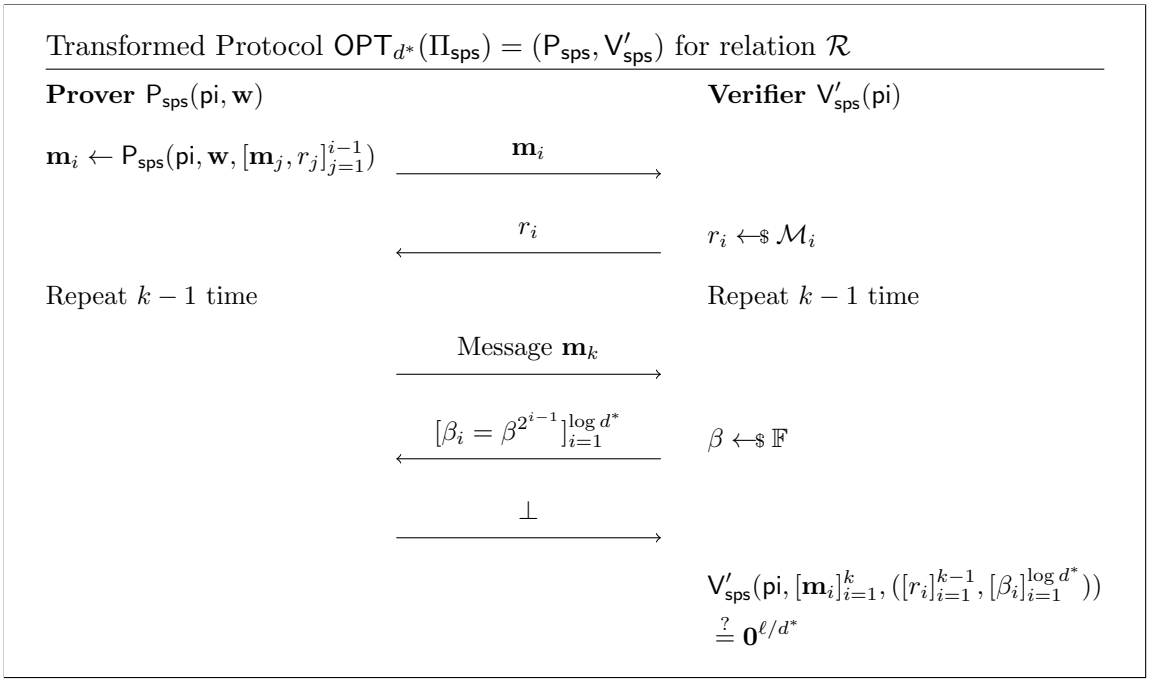


Figure 4: Generic Transformed Protocol to Π_{sps}.

For simplicity, we assume that d^* is a power of two that divides ℓ. Let

$$\mathsf{V}_{\mathsf{sps}}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^k, [r_i]_{i=1}^{k-1}) := \left[\mathsf{V}_{\mathsf{sps},0}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^k, [r_i]_{i=1}^{k-1}), \ldots, \mathsf{V}_{\mathsf{sps},d^*-1}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^k, [r_i]_{i=1}^{k-1})\right]$$

denote the original algebraic map where $\mathsf{V}_{\mathsf{sps},i-1}$ $(1 \le i \le d^*)$ is the i-th chunk of $\mathsf{V}_{\mathsf{sps}}$'s output with length ℓ/d^*. We describe the transformed protocol in Figure 4, where

$$\begin{aligned}\mathsf{V}'_{\mathsf{sps}}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^k, ([r_i]_{i=1}^{k-1}, [\beta_i]_{i=1}^{\log d^*})) &:= \sum_{j=0}^{d^*-1} L([\beta_i]_{i=1}^{\log d^*}, \langle j \rangle_{\log d^*}) \cdot \mathsf{V}_{\mathsf{sps},j}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^k, [r_i]_{i=1}^{k-1}) \\ &= \sum_{j=0}^{d^*-1} \beta^i \cdot \mathsf{V}_{\mathsf{sps},j}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^k, [r_i]_{i=1}^{k-1})\end{aligned}$$

and $\langle j \rangle_{\log d^*}$ denotes the $\log d^*$-bit binary representation of j.

The transformed protocol is a $(2k+1)$-move special sound protocol for the same relation $\mathcal{R}$. The verifier check now has output length ℓ/d^*, and the degree is $d + \log(d^*)$ because $\mathsf{V}_{\mathsf{sps}}$ has degree d and $eq^*([\beta_i]_{i=1}^{\log d^*}, \langle j \rangle_{\log d^*})$ has maximal degree $\log d^*$.

Lemma 3. *Let Π_{sps} be a $(2k-1)$-move protocol for relation $\mathcal{R}$ with $(a_1, \ldots, a_{k-1})$-special soundness, in which the verifier outputs ℓ elements. Let $d^* \le \ell$ and for simplicity we*

assume that d^* is a power of two and that divides ℓ. The transformed protocol $\mathsf{OPT}_{d^*}(\Pi_{\mathsf{sps}})$ of Π_{sps} is $(a_1, \ldots, , a_{k-1}, d^*)$-special sound.

Proof. Let $\mathsf{Ext}_{\mathsf{sps}}$ be the extractor for Π_{sps}. We construct a extractor $\mathsf{Ext}_{\mathsf{opt}}$ of $\mathsf{OPT}_{d^*}(\Pi_{\mathsf{sps}})$ for the same relation $\mathcal{R}$. Given an $(a_1, \ldots, , a_{k-1}, d^*)$-tree $\mathcal{T}$ of accepting transcripts, $\mathsf{Ext}_{\mathsf{opt}}$ invokes $\mathsf{Ext}_{\mathsf{sps}}$ on input the depth-$(k-1)$ transcript subtree of $\mathcal{T}$, and return what $\mathsf{Ext}_{\mathsf{sps}}$ outputs.

We prove that the extractor succeeds. For each internal node u at depth $k-1$, it has d^* children where each child maps to a distinct vector $(\beta, \beta^2, \beta^4, \ldots, \beta^{d^*/2}) \in \mathbb{F}^{\log d^*}$. Fix the messages $\mathsf{msg} = (\mathsf{pi}, [\mathbf{m}_i]_{i=1}^{k}, [r_i]_{i=1}^{k-1})$ at node u and let $\mathsf{V}_{\mathsf{sps}} := (\mathsf{V}_{\mathsf{sps},1}, \ldots, \mathsf{V}_{\mathsf{sps},d^*})$ be the verifier of Π_{sps}. Define the degree $d^* - 1$ univariate polynomial

$$p(X) := \sum_{j=0}^{d^*-1} X^j \cdot c_j$$

where $c_j := \mathsf{V}_{\mathsf{sps},j}(\mathsf{msg}) \in \mathbb{F}^{\ell/d^*}$ is $\mathsf{V}_{\mathsf{sps},j}$'s output on message msg. Since the transcripts are accepting, it holds that p evaluates to zero on the d^* different values of β that correspond to the d^* children of node u. Thus the univariate polynomial p is a zero polynomial, which implies that $\mathsf{V}_{\mathsf{sps}}$ outputs zero vector on message msg. Therefore for every node u at depth $k-1$, the sub-transcript from root to node u is an accepting transcript to Π_{sps}. Therefore the input to $\mathsf{Ext}_{\mathsf{sps}}$ is a valid $(a_1, \ldots, , a_{k-1})$-tree of accepting transcripts, and $\mathsf{Ext}_{\mathsf{sps}}$ will output the correct witness. □

Improve accumulation efficiency. By setting the parameter $d^* := \ell$, the error vectors $\mathbf{e}_j$ become single field elements, and we can use the trivial commitment $E_j := \mathsf{Commit}(\mathsf{ck}, \mathbf{e}_j) := \mathbf{e}_j$ without group operations. The prover needs to perform only k group operations (to combine $[C_i]_{i=1}^{k}$) and compute one more hash, but needs to evaluate a degree-$(d + \log(\ell))$ multivariate polynomial with $O(\ell)$ monomials. The accumulator instance needs to include $\log(\ell)$ additional values $(\beta, \beta^2, \beta^4, \ldots, \beta^{\ell/2})$. The accumulator verifier needs to do only k (rather than $k + d - 1$) group scalar multiplications, with the tradeoff of computing 1 more hash and doing $\log(\ell) + d - 1$ more field operations.

Theorem 4 (IVC for low-degree special sound protocols)**.** *Let $\mathbb{F}$ be a finite field and $\mathsf{cm} = (\mathsf{Setup}, \mathsf{Commit})$ be a binding homomorphic commitment scheme for vectors in $\mathbb{F}$. Given a $(2k-1)$-move $(a_1, \ldots, , a_{k-1})$-out-of-$|\mathbb{F}|$ special-sound protocol $\Pi_{\mathsf{sps}} = (\mathsf{P}_{\mathsf{sps}}, \mathsf{V}_{\mathsf{sps}})$ for an NP-complete relation $\mathcal{R}_{\mathsf{NP}}$ with inputs in $\mathbb{F}^{\ell_{\mathsf{in}}}$ and a degree-d verifier with output in $\mathbb{F}^{\ell}$, there exists a secure IVC schemes $\mathsf{IVC} = (\mathsf{P}_{\mathsf{IVC}}, \mathsf{V}_{\mathsf{IVC}})$ with predicates expressed in $\mathcal{R}_{\mathsf{NP}}$ with the following efficiencies:*

	$d^* = 1$	$d^* = \ell$
$\mathsf{P}_{\mathsf{IVC}}$ *native*	$\sum_{i=1}^{k} \lvert\mathbf{m}_i^*\rvert + (d-1)\ell\mathbb{G}$ $\mathsf{P}_{\mathsf{sps}} + L(\mathsf{V}_{\mathsf{sps}}, d, 1)$	$\sum_{i=1}^{k} \lvert\mathbf{m}_i^*\rvert\mathbb{G}$ $\mathsf{P}_{\mathsf{sps}} + L(\mathsf{V}_{\mathsf{sps}}, d, \ell)$
$\mathsf{P}_{\mathsf{IVC}}$ *recursive*	$d + k - 1\mathbb{G}$ $k + \ell_{\mathsf{in}}\mathbb{F}$ kH	$k\mathbb{G}$ $k - 1 + \ell_{\mathsf{in}} + d + 2\log\ell\mathbb{F}$ $k + 1H$
$\mathsf{V}_{\mathsf{IVC}}$*:*	$\ell + \sum_{i=1}^{k} \lvert\mathbf{m}_i\rvert\mathbb{G}$ $\mathsf{V}_{\mathsf{sps}}$	$\sum_{i=1}^{k} \lvert\mathbf{m}_i\rvert\mathbb{G}$ $\ell + d + \mathsf{V}_{\mathsf{sps}}$
$\lvert\pi_{\mathsf{IVC}}\rvert$ *:*	$k + \ell_{\mathsf{in}}\mathbb{F}$ $k + 1\mathbb{G}$ $\sum_{i=1}^{k} \lvert\mathbf{m}_i\rvert$	$k + 1 + \ell_{\mathsf{in}} + \log\ell\mathbb{F}$ $k\mathbb{G}$ $\sum_{i=1}^{k} \lvert\mathbf{m}_i\rvert$

d^* is a parameter; the first row displays the native operations of the IVC *prover. The second row describes the size of the recursive statement expressed as an instance of* $\mathcal{R}_{\mathsf{NP}}$ *for which* $\mathsf{P}_{\mathsf{IVC}}$ *creates a proof. The third row is the computation of* $\mathsf{V}_{\mathsf{IVC}}$*, and the last row is the size of the proof.*

In the table, $\lvert\mathbf{m}_i\rvert$ *denotes the prover message length;* $\lvert\mathbf{m}_i^*\rvert$ *is the number of non-zero elements in* $\mathbf{m}_i$*;* $\mathbb{G}$ *for rows 1-3 is the total length of the messages committed using* Commit*.* $\mathbb{F}$ *are field operations and* H *are calls to the random oracle.* $\mathsf{P}_{\mathsf{sps}}$ *(and* $\mathsf{V}_{\mathsf{sps}}$*) is the cost of running the prover (and the algebraic verifier) of the special sound protocol, respectively, and* $L(\mathsf{V}, d, d^*)$ *is the cost of computing the coefficients of the degree* $d + \log d^*$ *polynomial*

$$\mathbf{e}(X) := \sum_{i=0}^{d^*-1} eq^*\left([X \cdot \pi.\beta_k + \mathsf{acc}.\beta_k]_{k=1}^{\log d^*}, \langle i\rangle_{\log d^*}\right) \sum_{j=0}^{d} (\mu + X)^{d-j} \cdot f_{j,i}^{\mathcal{R}}(\mathsf{acc} + X \cdot \pi)\,, \quad (4)$$

where all inputs are linear functions in a formal variable X*.*[4] eq^* *is defined in Equation 3, and* $f_{j,i}^{\mathcal{R}}$ *is the* i*th* $(0 \le i \le d^* - 1)$ *chunk of* $f_j^{\mathcal{R}}$*'s output. For the proof size,* $\mathbb{G}$ *and* $\mathbb{F}$ *are the number of commitments and field elements, respectively.*

Proof. The construction first defines the NARK

$$\Pi_{\mathsf{NARK}} = (\mathsf{P}_{\mathsf{NARK}}, \mathsf{V}_{\mathsf{NARK}}) = \mathsf{FS}[\mathsf{cm}[\mathsf{OPT}_{d^*}(\Pi_{\mathsf{sps}})]]\,.$$

Then it uses the transformation from Theorem 1 to construct an IVC scheme $\Pi_{\mathsf{IVC}} = (\mathsf{P}_{\mathsf{IVC}}, \mathsf{V}_{\mathsf{IVC}})$.

Security: By Lemmas 1,2 and 3, we have that Π_{NARK} has $Q \cdot (1 - \frac{d}{|\mathbb{F}|}) \prod_{i=1}^{k-1}(1 - \frac{a_i}{|\mathbb{F}|})$ knowledge error for relation $\mathcal{R}_{\mathsf{cm}}^{\mathcal{R}_{\mathsf{NP}}}$ for a polynomial-time Q-query RO-adversary. Witnesses for $\mathcal{R}_{\mathsf{cm}}^{\mathcal{R}_{\mathsf{NP}}}$ are either a witness for $\mathcal{R}_{\mathsf{NP}}$ or a break of the binding property of cm. Assuming

[4] For example if $f_d = \prod_{i=1}^{d}(a_i + b_i \cdot X)$ then a naive algorithm takes $O(d^2)$ time but using FFTs it can be computed in time $O(d \log^2 d)$[CBBZ22].

that cm is a binding commitment scheme, the probability that a polynomial time adversary and a polynomial time extractor can compute such a break is $\mathsf{negl}(\lambda)$. Thus Π_{NARK} has knowledge error $Q \cdot (1 - \frac{2d}{|\mathbb{F}|}) \prod (1 - \frac{a_i}{|\mathbb{F}|}) + \mathsf{negl}(\lambda)$ for $\mathcal{R}_{\mathsf{NP}}$. Using Theorem 1, this yields that Π_{IVC} is a secure IVC scheme with predicates expressed in $\mathcal{R}_{\mathsf{NP}}$.

Efficiency: The IVC-prover runs $\mathsf{P}_{\mathsf{sps}}$ to compute all prover messages. It also commits to all the $\mathsf{P}_{\mathsf{sps}}$ messages using cm. Finally, it needs to compute and commit to all error terms $\mathbf{e}_1, \ldots, \mathbf{e}_{d+\log_2(d^*)-1}$ and commit to them. The error terms are computed by symbolically evaluating the polynomial $e(X)$ in Equation 4 with linear functions as inputs. If $d^* = \ell$, then the error terms are only one element in $\mathbb{F}$, so we can use the identity function as the trivial commitment scheme. Thus, there is no cost for committing to the error terms when $d^* = \ell$. The recursive circuit combines a new proof $\pi.x$ with an accumulator $\mathsf{acc}.x$. The size of the accumulator is ℓ_{in} field elements for the input, $k - 1 + \log_2(d^*)$ field elements for the interactive-proof challenges, 1 field element for the accumulator challenge, and k commitments for the $\mathsf{P}_{\mathsf{sps}}$ messages and 1 responsibility for the error term (if $d^* = \ell$ this is a trivial commitment, i.e., a field element). The IVC verifier checks the correctness of the commitments and runs $\mathsf{OPT}_{d^*}(\mathsf{V}_{\mathsf{sps}})$. □

Remark 5. *For simplicity, we assume that the public input, the prover messages, and the verifier challenges are all in the same field $\mathbb{F}$. This isn't strictly necessary; for example, the challenges could be drawn from a subset of $\mathbb{F}$. More generally, we can also allow prover messages to be group elements in $\mathbb{G}$ given a homomorphic commitment scheme to group elements(e.g. [AFGHO10]).*

4 Low-degree special-sound protocols for toolbox relations

In this section, we present special-sound protocols for permutation, high-degree gate, and lookup relations, which are the building blocks for the (non-uniform) Plonkish circuit-satisfiability relations. We can build accumulation schemes for (and thus IVCs from) these special-sound protocols via the framework presented in Section 3.

4.1 Permutation relation

Definition 9. *Let $\sigma : [n] \to [n]$ be a permutation, the relation $\mathcal{R}_\sigma$ is the set of tuples $\mathbf{w} \in \mathbb{F}^n$ such that $\mathbf{w}_i = \mathbf{w}_{\sigma(i)}$ for all $i \in [n]$.*

Special sound protocol Π_σ for permutation relation $\mathcal{R}_\sigma$

Prover $\mathsf{P}(\sigma, \mathbf{w} \in \mathbb{F}^n)$		**Verifier** $\mathsf{V}(\sigma)$
	$\xrightarrow{\mathbf{w}}$	
		Check $\mathbf{w}_i - \mathbf{w}_{\sigma(i)} = 0 \forall i \in [n]$

Complexity. Π_σ is a 1-move protocol (i.e. $k = 1$); the degree of the verifier is 1.

4.2 High-degree custom gate relation

Definition 10. *Given configuration* $\mathcal{C}_{GATE} := (n, c, d, [\mathbf{s}_i \in \mathbb{F}^n, G_i]_{i=1}^m)$ *where* n *is the number of gates,* c *is the arity per gate,* d *is the gate degree,* $[\mathbf{s}_i]_{i=1}^m$ *are the selector vectors, and* $[G_i]_{i=1}^m$ *are the gate formulas, the relation* $\mathcal{R}_{GATE}$ *is the set of tuples* $\mathbf{w} \in \mathbb{F}^{cn}$ *such that* $\sum_{j=1}^m \mathbf{s}_{j,i} \cdot G_j(\mathbf{w}_i, \mathbf{w}_{i+n}, \ldots, \mathbf{w}_{i+(c-1)\cdot n}) = 0$ *for all* $i \in [n]$.

Special sound protocol Π_{GATE} for relation $\mathcal{R}_{\text{GATE}}$

Prover $\mathsf{P}(\mathcal{C}_{\mathbf{GATE}}, \mathbf{w} \in \mathbb{F}^{cn})$		**Verifier** $\mathsf{V}(\mathcal{C}_{\mathbf{GATE}})$
	$\xrightarrow{\mathbf{w}}$	
		$\sum_{j=1}^m \mathbf{s}_{j,i} \cdot G_j(\mathbf{w}_i, \mathbf{w}_{i+n}, \ldots, \mathbf{w}_{i+(c-1)\cdot n})$ $\stackrel{?}{=} 0 \forall i \in [n]$

Complexity. Π_{GATE} is a 1-move protocol (i.e. $k = 1$); the degree of the verifier is d.

4.3 Lookup relation

Definition 11. *Given configuration* $\mathcal{C}_{LK} := (T, \ell, \mathbf{t})$ *where* ℓ *is the number of lookups and* $\mathbf{t} \in \mathbb{F}^T$ *is the lookup table, the relation* $\mathcal{R}_{LK}$ *is the set of tuples* $\mathbf{w} \in \mathbb{F}^\ell$ *such that* $\mathbf{w}_i \in \mathbf{t}$ *for all* $i \in [\ell]$.

We recall a useful lemma for lookup relation from [Hab22], and present a special sound protocol for the lookup relation.

Lemma 4 (Lemma 5 of [Hab22]). *Let $\mathbb{F}$ be a field of characteristic $p > \max(\ell, T)$. Given two sequences of field elements $[\mathbf{w}_i]_{i=1}^{\ell}$ and $[\mathbf{t}_i]_{i=1}^{T}$, we have $\{\mathbf{w}_i\} \subseteq \{\mathbf{t}_i\}$ as sets (with multiples of values removed) if and only if there exists a sequence $[\mathbf{m}_i]_{i=1}^{T}$ of field elements such that*

$$\sum_{i=1}^{\ell} \frac{1}{X + \mathbf{w}_i} = \sum_{i=1}^{T} \frac{\mathbf{m}_i}{X + \mathbf{t}_i} \,. \tag{5}$$

Special sound protocol Π_{LK} for $\mathcal{R}_{\mathrm{LK}}$

Prover $\mathsf{P}(\mathcal{C}_{\mathbf{LK}}, \mathbf{w} \in \mathbb{F}^{\ell})$		**Verifier** $\mathsf{V}(\mathcal{C}_{\mathbf{LK}})$
Compute $\mathbf{m} \in \mathbb{F}^T$ such that		
$\mathbf{m}_i := \sum_{j=1}^{\ell} \mathbb{1}(\mathbf{w}_j = \mathbf{t}_i) \forall i \in [T]$	$\xrightarrow{\mathbf{w}, \mathbf{m}}$	
	$\xleftarrow{r}$	$r \leftarrow\!\!\$\, \mathbb{F}$
Compute $\mathbf{h} \in \mathbb{F}^{\ell}$, $\mathbf{g} \in \mathbb{F}^T$		
$\mathbf{h}_i := \frac{1}{\mathbf{w}_i + r} \forall i \in [\ell]$		
$\mathbf{g}_i := \frac{\mathbf{m}_i}{\mathbf{t}_i + r} \forall i \in [T]$	$\xrightarrow{\mathbf{h}, \mathbf{g}}$	
		$\sum_{i=1}^{\ell} \mathbf{h}_i \stackrel{?}{=} \sum_{i=1}^{T} \mathbf{g}_i$
		$\mathbf{h}_i \cdot (\mathbf{w}_i + r) \stackrel{?}{=} 1 \forall i \in [\ell]$
		$\mathbf{g}_i \cdot (\mathbf{t}_i + r) \stackrel{?}{=} \mathbf{m}_i \forall i \in [T]$

Achieving perfect completeness. Note that the protocol does not have perfect completeness. If there exists an $\mathbf{w}_i$ or $\mathbf{t}_i$ such that $\mathbf{w}_i + r = 0$ $\mathbf{t}_i + r = 0$ then the prover message is undefined. We can achieve perfect completeness by having the verifier set $\mathbf{h}_i = 0$ or $\mathbf{g}_i = 0$ in this case and changing the verification equations to

$$(\mathbf{w}_i + r) \cdot (\mathbf{h}_i \cdot (\mathbf{w}_i + r) - 1) = 0$$

and

$$(\mathbf{t}_i + r) \cdot (\mathbf{g}_i \cdot (\mathbf{t}_i + r) - \mathbf{m}_i) = 0 \,.$$

These checks ensure that either $\mathbf{h}_i = \frac{1}{\mathbf{w}_i + r}$ or $\mathbf{w}_i + r = 0$. The checks increase the verifier degree to 3. Without these checks, the protocol has a negligible completeness error of

$\frac{\ell+T}{|\mathbb{F}|}$. This completeness error can likely be ignored in practice, and these checks do not need to be implemented. However, to achieve the full definition of PCD (which has perfect completeness) and use Theorem 1 by [BCLMS21], we require that all protocols have perfect completeness.

Complexity. Π_{LK} is a 3-move protocol (i.e. $k = 2$); the degree of the verifier is 2; the number of non-zero elements in the prover message is at most 4ℓ.

Accumulation with $O(\ell)$ prover complexity. The prover complexity of Π_{LK} is due to the sparseness of $\mathbf{g} \in \mathbb{F}^T$ and $\mathbf{m} \in \mathbb{F}^T$. However, there is no guarantee that when building an accumulation scheme for Π_{LK}, the accumulated $\mathsf{acc}.\mathbf{g}$ and $\mathsf{acc}.\mathbf{m}$ are sparse. This is an issue, as the prover needs to compute the error term $\mathbf{e}_1$. If we expand the accumulation procedures, we see that the three verification checks lead to three components of the error term $\mathbf{e}_1$:

$$\mathbf{e}_1^{(1)} = \left(\sum_{i=1}^{\ell} \mathsf{acc}.\mathbf{h}_i - \sum_{i=1}^{T} \mathsf{acc}.\mathbf{g}_i\right) + \mu \left(\sum_{i=1}^{\ell} \pi.\mathbf{h}_i - \sum_{i=1}^{T} \pi.\mathbf{g}_i\right) \in \mathbb{F}$$

$$\mathbf{e}_1^{(2)} = \mathsf{acc}.\mathbf{h} \circ (\pi.\mathbf{w} + \pi.r \cdot \mathbf{1}^{\ell}) + \pi.\mathbf{h} \circ (\mathsf{acc}.\mathbf{w} + \mathsf{acc}.r \cdot \mathbf{1}^{\ell}) - 2\mu \cdot \mathbf{1}^{\ell} \in \mathbb{F}^{\ell}$$

$$\mathbf{e}_1^{(3)} = \mathsf{acc}.\mathbf{g} \circ (\mathbf{t} + \pi.r \cdot \mathbf{1}^{T}) + \pi.\mathbf{g} \circ (\mu \cdot \mathbf{t} + \mathsf{acc}.r \cdot \mathbf{1}^{T}) - \mu \cdot \pi.\mathbf{m} - \mathsf{acc}.\mathbf{m} \in \mathbb{F}^{T}\,.$$

We examine all three components below.

For $\mathbf{e}_1^{(1)}$, we see that $(\sum_{i=1}^{\ell} \pi.\mathbf{h}_i - \sum_{i=1}^{T} \pi.\mathbf{g}_i) = 0$ by the assumption that π is valid, and $(\sum_{i=1}^{\ell} \mathsf{acc}.\mathbf{h}_i - \sum_{i=1}^{T} \mathsf{acc}.\mathbf{g}_i) = \mathsf{acc}.\mathbf{e}^{(1)}/\mathsf{acc}.\mu$ (where $\mathsf{acc}.\mathbf{e}^{(1)}$ is the first component of the error vector for acc). Thus $\mathbf{e}_1^{(1)} = \mathsf{acc}.\mathbf{e}^{(1)}/\mathsf{acc}.\mu$. We observe that since in IVC the accumulator $\mathsf{acc}.\mathbf{e}^{(1)}$ is initiated with 0, this implies that for all iterations $\mathbf{e}_1^{(1)} = 0$.

For $\mathbf{e}_1^{(2)}$, it is computed from terms of size ℓ, so can be computed in time $O(\ell)$.

For $\mathbf{e}_1^{(3)}$, note that $\mathsf{acc}.\mu$, $\mathsf{acc}.r$ and $\pi.r$ are all scalars. Also note that the accumulation prover only needs to compute the commitment $E_1 = \mathsf{Commit}(\mathsf{ck}, \mathbf{e}_1) = \mathsf{Commit}(\mathsf{ck}, \mathbf{e}_1^{(1)}) + \mathsf{Commit}(\mathsf{ck}, 0||\mathbf{e}_1^{(2)}) + \mathsf{Commit}(\mathsf{ck}, \mathbf{0}^{\ell+1}||\mathbf{e}_1^{(3)})$, not the actual vector $\mathbf{e}_1$. We will compute $E_1^{(3)} = \mathsf{Commit}(\mathsf{ck}, \mathbf{e}_1^{(3)})$ homomorphically from the commitments below (dropping the zero padding for readability):

1. $G = \mathsf{Commit}(\mathsf{ck}, \pi.\mathbf{g})$,
2. $G' = \mathsf{Commit}(\mathsf{ck}, \mathsf{acc}.\mathbf{g})$,
3. $M = \mathsf{Commit}(\mathsf{ck}, \pi.\mathbf{m})$,
4. $M' = \mathsf{Commit}(\mathsf{ck}, \mathsf{acc}.\mathbf{m})$,
5. $GT = \mathsf{Commit}(\mathsf{ck}, \pi.\mathbf{g} \circ \mathbf{t})$,
6. $GT' = \mathsf{Commit}(\mathsf{ck}, \mathsf{acc}.\mathbf{g} \circ \mathbf{t})$.

Given these commitments, we can compute

$$E_1^{(3)} = GT' + \pi.r \cdot G' + \mathsf{acc}.\mu \cdot GT + \mathsf{acc}.r \cdot G - \mathsf{acc}.\mu \cdot M - M' .$$

This reduces the problem to the problem of efficiently computing and updating the commitments. G, M and GT are all commitments to ℓ-sparse vectors, thus can be efficiently computed. The prover can cache the commitments G', M', and GT' and efficiently update them during accumulation. That is $G'' \leftarrow G' + \alpha G$, $M'' \leftarrow M' + \alpha M$ and $GT'' \leftarrow GT' + \alpha GT$. Additionally, we need to update the accumulation witnesses: $\mathsf{acc}'.\mathbf{m} \leftarrow \mathsf{acc}.\mathbf{m} + \alpha\pi.\mathbf{m}$ and $\mathsf{acc}'.\mathbf{g} \leftarrow \mathsf{acc}.\mathbf{g} + \alpha\pi.\mathbf{g}$. Again because $\pi.\mathbf{g}, \pi.\mathbf{m}$ are sparse this can be done in time $O(\ell_{\mathsf{lk}})$ independent of $T = |\mathbf{t}|$.

When Π_{LK} is used in composition with another special sound protocol with a higher degree d, the accumulation is made homogeneous using a $(X+\mu)^{d-2}$ factor when computing the error terms. The contribution to the error terms $\mathbf{e}_i$ $(1 \le i \le d-1)$ is still a linear function in $\mathsf{acc}.\mathbf{g}$, $\mathsf{acc}.\mathbf{m}$ and $\mathsf{acc}.\mathbf{g} \circ \mathbf{t}$, and thus can be computed homomorphically from commitments to these values.

Special-soundness. We prove special-soundness for the perfect complete version of Π_{LK}, the proof for Π_{LK} is almost identical (but even simpler).

Lemma 5. *The perfect complete version of Π_{LK} is $2(\ell + T)$-special-sound.*

Proof. We construct an extractor Ext that outputs $\mathbf{w}$. To show that the witness is valid, we look at the $2(\ell + T)$ transcripts that all have $\mathbf{w}, \mathbf{m}$ as the first message but different $(r^{(j)}, \mathbf{h}^{(j)} \in \mathbb{F}^{\ell}, \mathbf{g}^{(j)} \in \mathbb{F}^{T})$ as the second message. Note that by the pigeonhole principle, there must exist a subset of $S \subseteq [2(\ell+T)]$ transcripts such that $|S| = \ell+T$ and $\mathbf{w}_i + r^{(j)} \neq 0$ for all $i \in [\ell]$ and $j \in S$, and $\mathbf{t}_i + r^{(j)} \neq 0$ for all $i \in [T]$ and $j \in S$. For these transcripts, we have that $\mathbf{h}_i = \frac{1}{\mathbf{w}_i + r^{(j)}}$ and $\mathbf{g}_i = \frac{\mathbf{m}_i}{\mathbf{t}_i + r^{(j)}}$. Define the degree $\ell + T - 1$ polynomial

$$p(X) = \prod_{k=1}^{\ell}(X + \mathbf{w}_k) \cdot \prod_{j=1}^{T}(X + \mathbf{t}_j) \cdot \left(\sum_{i=1}^{\ell} \frac{1}{X + \mathbf{w}_i} - \sum_{i=1}^{T} \frac{\mathbf{m}_i}{X + \mathbf{t}_i} \right) .$$

If $p(X)$ is the zero polynomial then $\sum_{i=1}^{\ell} \frac{1}{X+\mathbf{w}_i} = \sum_{i=1}^{T} \frac{\mathbf{m}_i}{X+\mathbf{t}_i}$ and by Lemma 4 $(\mathcal{C}_{\mathrm{LK}}; \mathbf{w}) \in \mathcal{R}_{\mathrm{LK}}$. Since we have $\ell + T$ points $r^{(j)}$ at which $p(r_j) = 0$ we get that $p = 0$ and thus that the extracted witness $\mathbf{w}$ is valid. □

4.4 Vector-valued lookup

In some applications (e.g., simulating bit operations in circuits), we need to support lookup for a vector, i.e., each table value is a vector of field elements. In this section, we adapt the scheme in Section 4.3 to support vector lookups.

Definition 12. *Consider configuration* $\mathcal{C}_{VLK} := (T, \ell, v := 2^\nu \in \mathbb{N}, \mathbf{t})$ *where* ℓ *is the number of lookups, and* $\mathbf{t} \in (\mathbb{F}^v)^T$ *is a lookup table in which the ith* $(1 \leq i \leq T)$ *entry is*

$$\mathbf{t}_i := (\mathbf{t}_{i,1}, \ldots, \mathbf{t}_{i,v}) \in \mathbb{F}^v \,.$$

A sequence of vectors $\mathbf{w} \in (\mathbb{F}^v)^\ell$ *is in relation* $\mathcal{R}_{VLK}$ *if and only if for all* $i \in [\ell]$,

$$\mathbf{w}_i := (\mathbf{w}_{i,1}, \ldots, \mathbf{w}_{i,v}) \in \mathbf{t} \,.$$

As noted in Section 3.4 of [Hab22], we can extend Lemma 4 and replace Equation 5 with

$$\sum_{i=1}^{\ell} \frac{1}{X + w_i(Y_1, \ldots, Y_\nu)} = \sum_{i=1}^{T} \frac{\mathbf{m}_i}{X + t_i(Y_1, \ldots, Y_\nu)} \tag{6}$$

where let $\langle \mathbf{b} \rangle_\nu := \sum_{i=0}^{\nu-1} 2^i \cdot \mathbf{b}_i$, the polynomials are defined as

$$w_i(Y_1, \ldots, Y_\nu) := \sum_{\mathbf{b} \in \{0,1\}^\nu} \mathbf{w}_{i,1+\langle \mathbf{b} \rangle_\nu} \cdot Y_1^{\mathbf{b}_1} \cdots Y_\nu^{\mathbf{b}_\nu} \,,$$
$$t_i(Y_1, \ldots, Y_\nu) := \sum_{\mathbf{b} \in \{0,1\}^\nu} \mathbf{t}_{i,1+\langle \mathbf{b} \rangle_\nu} \cdot Y_1^{\mathbf{b}_1} \cdots Y_\nu^{\mathbf{b}_\nu} \,,$$

which represent the witness vector $\mathbf{w}_i \in \mathbb{F}^v$ and the table vector $\mathbf{t}_i \in \mathbb{F}^v$. We, therefore, can describe a special-sound protocol for the vector lookup relation as follows.

Special sound protocol Π^v_{VLK} for $\mathcal{R}_{\text{VLK}}$

Prover $\mathsf{P}(\mathcal{C}_{\mathbf{VLK}}, \mathbf{w} \in (\mathbb{F}^v)^\ell)$		**Verifier** $\mathsf{V}(\mathcal{C}_{\mathbf{VLK}})$
Compute $\mathbf{m} \in \mathbb{F}^T$ such that		
$\mathbf{m}_i := \sum_{j=1}^{\ell} \mathbb{1}(\mathbf{w}_j = \mathbf{t}_i) \forall i \in [T]$	$\xrightarrow{\mathbf{w}, \mathbf{m}}$	
	$\xleftarrow{\beta_1, \ldots, \beta_\nu}$	$(\beta_1, \ldots, \beta_\nu) \leftarrow\!\!\$\ \mathbb{F}^{\nu = \lceil \log(v) \rceil}$
	$\xrightarrow{\perp}$	
	$\xleftarrow{r}$	$r \leftarrow\!\!\$\ \mathbb{F}$
Compute $\mathbf{h} \in \mathbb{F}^\ell$, $\mathbf{g} \in \mathbb{F}^T$		
$\mathbf{h}_i := \frac{1}{w_i(\beta_1, \ldots, \beta_\nu) + r} \forall i \in [\ell]$		
$\mathbf{g}_i := \frac{\mathbf{m}_i}{t_i(\beta_1, \ldots, \beta_\nu) + r} \forall i \in [T]$	$\xrightarrow{\mathbf{h}, \mathbf{g}}$	
		$\sum_{i=1}^{\ell} \mathbf{h}_i \stackrel{?}{=} \sum_{i=1}^{T} \mathbf{g}_i$
		$\mathbf{h}_i \cdot (w_i(\beta_1, \ldots, \beta_\nu) + r) \stackrel{?}{=} 1 \forall i \in [\ell]$
		$\mathbf{g}_i \cdot (t_i(\beta_1, \ldots, \beta_\nu) + r) \stackrel{?}{=} \mathbf{m}_i$ $\forall i \in [T]$

Achieving perfect completeness. We can use the same trick in Section 4.3 to achieve perfect completeness for Π^v_{VLK}. Namely, the verifier sets $\mathbf{h}_i = 0$ or $\mathbf{g}_i = 0$ when $w_i(\beta_1, \ldots, \beta_\nu) + r = 0$ or $t_i(\beta_1, \ldots, \beta_\nu) + r = 0$ respectively. The verification equations become

$$(w_i(\beta_1, \ldots, \beta_\nu) + r) \cdot (\mathbf{h}_i \cdot (w_i(\beta_1, \ldots, \beta_\nu) + r) - 1) = 0$$

and

$$(t_i(\beta_1, \ldots, \beta_\nu) + r) \cdot (\mathbf{g}_i \cdot (t_i(\beta_1, \ldots, \beta_\nu) + r) - \mathbf{m}_i) = 0\,.$$

The degree of the verifier increases to $2 + 2\log(v)$. In practice, the negligible completeness error can likely be ignored without implementing these checks.

Prover complexity. Π_{VLK} is a 5-move protocol (i.e. $k = 3$) with the 2nd prover message being empty; the degree of the verifier is $2 + \log(v)$ where v is the vector length; the number

of non-zero elements in the prover message is at most $(v+3)\ell$.

Accumulation complexity To ensure that the accumulation procedure only requires $O(\ell_{\mathsf{lk}})$ operations independent of T, we can apply the same trick as in Section 4.3 and compute all error terms from homomorphic commitments to $\mathbf{g}$ and $\mathbf{m}$. This works because the error terms are a linear function of $\mathbf{g}$ and $\mathbf{m}$ and scalars. This means the contributions of the not necessarily sparse $\mathsf{acc}.\mathbf{g}, \mathsf{acc}.\mathbf{m}$ to $\mathbf{e}_1, \ldots, \mathbf{e}_{d-1}$ can be computed using the commitment homomorphism.

Special soundness. We prove that the perfect complete version of Π^v_{VLK} is special sound.

Lemma 6. *For any constant $v = 2^\nu \in \mathbb{N}$, the perfect complete version of Π^v_{VLK} is $[(\ell+T)^{\nu+1}, \ell+T+1]$-special-sound.*

Proof. We construct an extractor Ext that outputs $\mathbf{w}$. To show that the witness is valid, we look at the $[(\ell+T)^{\nu+1}, \ell+T+1]$-tree of accepting transcripts. Note that for each depth-$(\nu+1)$ internal node u that fixes the message $(\mathbf{w}, \mathbf{m}, \beta_1, \ldots, \beta_\nu)$, it has $\ell+T+1$ different choices of challenge $r^{(j)}$. By the pigeonhole principle, there exists at least one challenge r such that $t_i(\beta_1, \ldots, \beta_\nu) + r \neq 0$ for all $i \in [T]$ and $w_i(\beta_1, \ldots, \beta_\nu) + r \neq 0$ for all $i \in [\ell]$. Let $\mathbf{h}, \mathbf{g}$ be the last prover message in the corresponding leaf node. Since the transcript is accepting, we have that $\mathbf{h}_i = 1/(w_i(\beta_1, \ldots, \beta_\nu) + r)$ for all $i \in [\ell]$, $\mathbf{g}_i = \mathbf{m}_i/(t_i(\beta_1, \ldots, \beta_\nu) + r)$ for all $i \in [T]$, and $\sum_{i=1}^{\ell} \mathbf{h}_i = \sum_{i=1}^{T} \mathbf{g}_i$.

Define the $(\nu+1)$-variate polynomial where the *individual* degree of each variable is at most $\ell + T - 1$,

$$p(X, Y_1, \ldots, Y_\nu) = \prod_{k=1}^{\ell} (X + w_k(Y_1, \ldots, Y_\nu)) \cdot \prod_{j=1}^{T} (X + t_j(Y_1, \ldots, Y_\nu)) \cdot \left(\sum_{i=1}^{\ell} \frac{1}{X + w_i(Y_1, \ldots, Y_\nu)} - \sum_{i=1}^{T} \frac{\mathbf{m}_i}{X + t_i(Y_1, \ldots, Y_\nu)} \right).$$

For every depth-$(\nu+1)$ internal node u, we denote by $(r, \beta_1, \ldots, \beta_\nu)$ the partial transcript for one of the u's children whose challenge r satisfies $t_i(\beta_1, \ldots, \beta_\nu) + r \neq 0$ for all $i \in [T]$ and $w_i(\beta_1, \ldots, \beta_\nu) + r \neq 0$ for all $i \in [\ell]$. As argued in the previous paragraph, we observe that $\sum_{i=1}^{\ell} \frac{1}{r + w_i(\beta_1, \ldots, \beta_\nu)} - \sum_{i=1}^{T} \frac{\mathbf{m}_i}{r + t_i(\beta_1, \ldots, \beta_\nu)} = 0$, hence p evaluates to zero at point $(r, \beta_1, \ldots, \beta_\nu)$. Since there are $(\ell+T)^{\nu+1}$ depth-$(\nu+1)$ internal nodes, it holds that p evaluates to zero on $(\ell+T)^{\nu+1}$ different points, which implies that p is the zero polynomial. Therefore, $\sum_{i=1}^{\ell} \frac{1}{X + w_i(Y_1, \ldots, Y_\nu)} = \sum_{i=1}^{T} \frac{\mathbf{m}_i}{X + t_i(Y_1, \ldots, Y_\nu)}$. Then by the extension of Lemma 4 described in Equation 6, we have $(\mathcal{C}_{\mathrm{VLK}}, \mathbf{w}) \in \mathcal{R}_{\mathrm{VLK}}$ and thus the extracted witness is valid. □

5 Special sound protocols for Plonkup relations

Definition 13. *Consider configuration* $\mathcal{C}_{plonkup} := (n, T; \sigma; c, d, [\mathbf{s}_i, G_i]_{i=1}^m; L, \mathbf{t})$ *where* $\sigma : [cn] \to [cn]$ *is a permutation,* $(c, d, [\mathbf{s}_i, G_i]_{i=1}^m)$ *are the parameters for the high-degree custom gates,* $L \subseteq [cn]$ *is the subset of indices for variables that have a lookup gate,* $\mathbf{t} \in \mathbb{F}^T$ *is the lookup table. The relation* $\mathcal{R}_{plonkup}$ *is the set of tuples* $(\mathsf{pi} \in \mathbb{F}^{\ell_{\mathsf{in}}}, \mathbf{w} \in \mathbb{F}^{cn})$ *such that*

$$\mathbf{w} \in \mathcal{R}_\sigma \wedge \mathbf{w} \in \mathcal{R}_{GATE} \wedge \mathbf{w}_L \in \mathcal{R}_{LK} \wedge \mathbf{w}[1..\ell_{\mathsf{in}}] = \mathsf{pi}\,.$$

We present the special sound protocol for the Plonkup relation $\mathcal{R}_{\text{plonkup}}$ below.

Special sound protocol Π_{plonkup} for relation $\mathcal{R}_{\text{plonkup}}$

Prover $\mathsf{P}(\mathcal{C}_{\mathbf{plonkup}}, \mathsf{pi}, \mathbf{w})$		**Verifier** $\mathsf{V}(\mathcal{C}_{\mathbf{plonkup}}, \mathsf{pi})$
Compute $\mathbf{m} \in \mathbb{F}^T$ such that		
$\mathbf{m}_i := \sum_{j \in L} \mathbb{1}(\mathbf{w}_j = \mathbf{t}_i) \forall i \in [T]$	$\xrightarrow{\mathbf{w}, \mathbf{m}}$	$\mathbf{w}_i - \mathbf{w}_{\sigma(i)} \stackrel{?}{=} 0 \forall i \in [cn]$
		$\sum_{j=1}^m \mathbf{s}_{j,i} \cdot G_j(\mathbf{w}_i, \ldots, \mathbf{w}_{i+cn-n}) \stackrel{?}{=} 0 \forall i \in [n]$
		$\mathbf{w}_i \stackrel{?}{=} \mathsf{pi}_i \forall i \in [\ell_{\mathsf{in}}]$
	$\xleftarrow{r}$	$r \leftarrow\!\!\$\, \mathbb{F}$
Compute $\mathbf{h} \in \mathbb{F}^{\lvert L \rvert}$, $\mathbf{g} \in \mathbb{F}^T$		
$\mathbf{h}_i := \frac{1}{\mathbf{w}_{L_i} + r} \forall i \in [\lvert L \rvert]$		
$\mathbf{g}_i := \frac{\mathbf{m}_i}{\mathbf{t}_i + r} \forall i \in [T]$	$\xrightarrow{\mathbf{h}, \mathbf{g}}$	$\sum_{i=1}^{\lvert L \rvert} \mathbf{h}_i \stackrel{?}{=} \sum_{i=1}^T \mathbf{g}_i$
		$\mathbf{h}_i \cdot (\mathbf{w}_{L_i} + r) \stackrel{?}{=} 1 \forall i \in [\lvert L \rvert]$
		$\mathbf{g}_i \cdot (\mathbf{t}_i + r) \stackrel{?}{=} \mathbf{m}_i \forall i \in [T]$

Complexity. Π_{plonkup} is a 3-move protocol (i.e. $k = 2$); the degree of the verifier is d; the number of non-zero elements in the prover message is at most $cn + 3|L|$.

Completeness and security. We need to add the checks described in Section 4.3 to achieve perfect completeness. This changes the verification degree to $\max(d, 3)$. Without these checks, the protocol still has all but negligible completeness.

Lemma 7. *$\Pi_{plonkup}$ is $2(T + |L|)$-special sound.*

Proof. The protocol is a parallel composition of $\Pi_\sigma, \Pi_{\mathrm{GATE}}$ and Π_{LK} plus a public input check. In Π_σ and Π_{GATE}, the prover simply sends the witness, and the verifier checks it is in the relation. These protocols are thus trivially 1-special sound. The public input relation also trivially holds as the verifier checks $\mathbf{w}_i = \mathsf{pi}_i$ for all $i \in [\ell_{\mathsf{in}}]$. By Lemma 5 Π_{LK} is $2(T + |L|)$-special sound. Thus Π_{plonkup} is $2(T + |L|)$-special sound. □

6 Special sound protocols for non-uniform Plonkup relations

In this section, we describe a special sound protocol for capturing non-uniform Plonkup circuit computations. In particular, the relation is checking that one of the I circuits is satisfied, where the index of the target circuit is determined by a part of the public input called program counter *pc*. The non-uniform Plonkup circuit can add arbitrary constraints on input *pc*. For example, the list of I circuits can be the opcodes supported by EVM, the program counter *pc* can be computed from the pc' and the register state in the previous step, and the circuit will further check that $\mathsf{opcode}[pc]$ is executed correctly in the current step. For another application, we can consider the I circuits as the predicates of I smart contracts (or transaction types), a user can call one of the smart contracts/transaction types by specifying the index *pc*, and the cost of proving correct execution is only proportional to the size of an individual smart contract/transaction type rather than the sum of the sizes of the supported smart contracts/transaction types.

For ease of exposition, we assume that the I circuits have the same

- number of gates n;
- gate arity c;
- maximum gate degree d;
- number of gate types m;
- number of public inputs ℓ_{in};
- number of lookup gates ℓ_{lk}.

The scheme naturally extends when different branch circuits have different parameters.

Definition 14. *Consider configuration $\mathcal{C}_{mplkup} := (\mathsf{pp} = (n, T, c, d, m, \ell_{\mathsf{in}}, \ell_{\mathsf{lk}}); [\mathcal{C}_i]_{i=1}^{I}; \mathbf{t}))$ where the ith $(1 \le i \le I)$ branch circuit has configuration $\mathcal{C}_i := (\mathsf{pp}, \sigma_i, [\mathbf{s}_{i,j}, G_{i,j}]_{j=1}^{m}, L_i)$, and $\mathbf{t} \in \mathbb{F}^T$ is the global lookup table. For a public input $\mathsf{pi} := (pc, \mathsf{pi}') \in \mathbb{F}^{\ell_{\mathsf{in}}}$ where $pc \in [I]$ is a program counter, we say that a instance-witness pair $(\mathsf{pi}, \mathbf{w} \in \mathbb{F}^{cn})$ is in the relation $\mathcal{R}_{mplkup}$ if and only if $(\mathsf{pi}, \mathbf{w}) \in \mathcal{R}_{plonkup}$ w.r.t. circuit configuration $(\mathcal{C}_{pc}, \mathbf{t})$.*

Notation. For an integer $x \in [0, 2^\kappa)$, we denote by $\langle x \rangle_\kappa$ the κ-bit binary representation of x. For vector $\mathbf{a}, \mathbf{b} \in \mathbb{F}^\kappa$, we define

$$eq(\mathbf{a}, \mathbf{b}) := \prod_{i=1}^{\kappa} (\mathbf{a}_i \mathbf{b}_i + (1 - \mathbf{a}_i)(1 - \mathbf{b}_i)) .$$

For simplicity, we assume that the number of branch circuits I is a power of two. We present the special-sound protocol Π_{mplkup} for the multi-circuit Plonkup relation.

Protocol $\Pi_{\mathbf{mplkup}} = \langle \mathsf{P}(\mathcal{C}_{\mathbf{mplkup}}, \mathsf{pi}, \mathbf{w}), \mathsf{V}(\mathcal{C}_{\mathbf{mplkup}}, \mathsf{pi} = (pc \in [I], \mathsf{pi}')) \rangle$:

1. P computes and sends V vector $\mathbf{pc} = (i_1, \ldots, i_{\log I}) \in \{0,1\}^{\log I}$ where the program counter pc satisfies $pc = 1 + \sum_{j=1}^{\log I} 2^{j-1} \cdot i_j$.
2. V checks that $i_j \cdot (1 - i_j) \stackrel{?}{=} 0 \forall j \in [\log I]$ and $pc \stackrel{?}{=} 1 + \sum_{j=1}^{\log I} 2^{j-1} \cdot i_j$.
3. P sends V vector $\mathbf{b} = (0, \ldots, 0, b_{pc} = 1, 0, \ldots, 0) \in \mathbb{F}^I$.
4. V checks that $\mathbf{b}_i \stackrel{?}{=} eq(\langle i-1 \rangle_{\log I}, \mathbf{pc})$ for all $i \in [I]$.
5. P computes vector $\mathbf{m} \in \mathbb{F}^T$ such that $\mathbf{m}_i := \sum_{j \in L_{pc}} \mathbb{1}(\mathbf{w}_j = \mathbf{t}_i) \forall i \in [T]$.
6. P sends V vectors $\mathbf{w}, \mathbf{m}$.
7. V checks that

   Permutation check: $\sum_{j=1}^{I} \mathbf{b}_j (\mathbf{w}_i - \mathbf{w}_{\sigma_j(i)}) \stackrel{?}{=} 0$ for all $i \in [cn]$.

   Public input check: $\mathbf{w}[1..\ell_{\mathsf{in}}] \stackrel{?}{=} \mathsf{pi}$.

   Gate check: for all $i \in [n]$, it holds that

$$\sum_{j=1}^{I} \mathbf{b}_j \cdot \mathsf{GT}_j(\mathbf{s}_{j,1}[i], \ldots, \mathbf{s}_{j,m}[i], \mathbf{w}_i, \ldots, \mathbf{w}_{i+cn-n}) = 0$$

   where $\mathsf{GT}_j(s_1, \ldots, s_m, x_1, \ldots, x_c) := \sum_{i=1}^{m} s_i \cdot G_{j,i}(x_1, \ldots, x_c)$.
8. V samples and sends P random challenge $r \leftarrow\!\!\$\, \mathbb{F}$.
9. P computes vectors $\mathbf{h} \in \mathbb{F}^{\ell_{\mathsf{lk}}}$, $\mathbf{g} \in \mathbb{F}^T$ such that

$$\mathbf{h}_i := \frac{1}{\mathbf{w}_{L_{pc}[i]} + r} \forall i \in [\ell_{\mathsf{lk}}], \qquad \mathbf{g}_i := \frac{\mathbf{m}_i}{\mathbf{t}_i + r} \forall i \in [T].$$

10. V checks that $\sum_{i=1}^{\ell_{\mathsf{lk}}} \mathbf{h}_i \stackrel{?}{=} \sum_{i=1}^{T} \mathbf{g}_i$ and

$$\sum_{j=1}^{I} \mathbf{b}_j \cdot \Big[\mathbf{h}_i \cdot (\mathbf{w}_{L_j[i]} + r) \Big] \stackrel{?}{=} 1 \quad \forall i \in [\ell_{\mathsf{lk}}],$$

$$\mathbf{g}_i \cdot (\mathbf{t}_i + r) \stackrel{?}{=} \mathbf{m}_i \quad \forall i \in [T]$$

Remark 6. *By the public input check* $\mathbf{w}[1..\ell_{\mathsf{in}}] \stackrel{?}{=} \mathsf{pi}$*, we guarantee that* $\mathbf{w}[1] = pc$*, and the circuit relation can add arbitrary constraints on pc depending on the applications (like the ones described in the header of Section 6).*

Complexity. Π_{mplkup} is a 3-move protocol (i.e. $k = 2$); the degree of the verifier is $\max(d+1, \log I)$; the number of non-zero elements in the prover message is at most $cn + 3\ell_{\mathsf{lk}} + 1 + \log I$; the prover message length is $\log I + I + cn + 3T$. Hence in the resulting accumulation scheme, the accumulation prover complexity is only $O(n + \ell_{\mathsf{lk}} + \log I)$ that is independent of the table size, and the accumulator size is $O(n + T + I)$ that is independent of the sum of the sizes of the branch circuits. The decider still runs in time $O(I \cdot c \cdot n)$ as it needs to evaluate all circuits at the accumulator.

Special soundness. We prove the special soundness property of Π_{mplkup} below.

Lemma 8. Π_{mplkup} *is* $2(T + \ell_{\mathsf{lk}})$*-special sound.*

Proof. The extractor Ext outputs the witness $\mathbf{w}$ sent by the prover. Note that $\mathbf{pc} = (i_1, \ldots, i_{\log I}) \in \{0,1\}^{\log I}$ is the $\log I$-bit binary representation of $pc - 1$ if the verifier-check at step 2 passes. Conditioned on this, if the verifier check at step 4 also passes, since for any $\mathbf{a}, \mathbf{b} \in \{0,1\}^{\log I}$, $eq(\mathbf{a}, \mathbf{b}) = 1$ if and only if $\mathbf{a} = \mathbf{b}$ and equals 0 otherwise, it must be the case that $\mathbf{b}$ is a bool vector with a single non-zero element b_{pc}. Also, note that given $2(T + \ell_{\mathsf{lk}})$ accepting transcripts with distinct challenges r, the vector $\mathbf{b}$ won't change. Therefore the sub-transcript between step 5 to step 10 is essentially a transcript for a Plonkup special sound protocol Π_{plonkup} with configuration $\mathcal{C}_{\text{plonkup}} := (n, T, c, d, \mathcal{C}_{pc}, \mathbf{t})$. By Lemma 7, it holds that Π_{mplkup} is $2(T + \ell_{\mathsf{lk}})$-special sound. □

7 Protostar

We will now use Π_{mplkup} and our compiler described in Theorem 4 to design Protostar. Before that, we must address an efficiency issue when combining the high-degree gate and sparse lookup protocols with the generic transform OPT in Section 3.5.

Efficient accumulation of $\mathsf{OPT}_n(\Pi_{\mathbf{mplkup}})$. $\mathsf{OPT}_n(\Pi_{\text{GATE}})$ reduces the number of verification checks in Π_{GATE} from n to 1. In the resulting accumulation scheme, the error terms are, thus, only of length 1. This enables using the trivial identity commitment for the error terms and thus reduces the number of group operations by the accumulation verifier. Unfortunately, applying OPT to mplkup seems to have a major tradeoff. The number of verification checks is $n + \ell_{\mathsf{lk}} + T + c \cdot n$. This requires using a) $\mathsf{OPT}_{\ell_{\mathsf{lk}}+T+(c+1)\cdot n}$(mplkup) and b) is not composable with the sparseness optimizations for lookup described in Sections 4.3 and 4.4. These optimizations make the prover computation independent of T. A closer look at the verification of mplkup reveals that only n of these verification checks are of high

degree d, namely the checks in Π_{GATE}. The other checks are of degree 2 or lower. With a slight abuse of notation, we can define $\mathsf{OPT}_n(\Pi_{\text{mplkup}})$ as applying the generic transform OPT only to the Π_{GATE} part of Π_{mplkup}. This means that there are $\log_2(n) + d - 1$ cross error vectors (each of length 1) for the checks in $\mathsf{OPT}_n(\Pi_{\text{GATE}})$, and 1 cross error vector of length $T + \ell_{\mathsf{lk}} + cn$ for the rest checks. We can use the identity function to commit to the field elements and a vector commitment to commit to the long error term. We can again leverage homomorphism as described in Section 4.3 to make the prover independent of T.

Corollary 1 (PROTOSTAR protocol)**.** *Consider the configuration*

$$\mathcal{C}_{mplkup} := (n, T, c, d, m, \ell_{\mathsf{in}}, \ell_{\mathsf{lk}}; [\mathcal{C}_i]_{i=1}^{I}; \mathbf{t}).$$

Given a binding homomorphic commitment scheme $\mathsf{cm} = (\mathsf{Setup}, \mathsf{Commit})$*, there exists an IVC schemes* PROTOSTAR_{d^*} *with parameter* d^* *for* $\mathcal{R}_{mplkup}$ *relations with the following efficiencies for* $m = 1$ *(i.e. each circuit has a single degree-d gate type), public input length* $\ell_{\mathsf{in}} = 1$*, and* $d + 1 \geq \log_2(I)$*: (we omit cost terms that are negligible compared to the dominant parts)*

	$d^* = 1$	$d^* = n$
$\mathsf{P}_{\text{PROTOSTAR}}$ *native*	$O((c+d) \cdot n + \ell_{\mathsf{lk}})\mathbb{G}$ $L(\mathcal{C}_i, d, 1) + 2\ell_{\mathsf{lk}}\mathbb{F}$	$O(c \cdot n + \ell_{\mathsf{lk}})\mathbb{G}$ $L(\mathcal{C}_i, d, \log n) + 2\ell_{\mathsf{lk}}\mathbb{F}$
$\mathsf{P}_{\text{PROTOSTAR}}$ *recursive*	$d + 1\mathbb{G}$ $2\mathbb{F}$ $2H$	$3\mathbb{G}$ $d + 1 + 2\log n\mathbb{F}$ $3H$
$\mathsf{V}_{\text{PROTOSTAR}}$	$c \cdot n + T + \ell_{\mathsf{lk}}\mathbb{G}$ $\sum_{i=1}^{I} \mathcal{C}_i + T + \ell_{\mathsf{lk}}\mathbb{F}$	$c \cdot n + T + \ell_{\mathsf{lk}}\mathbb{G}$ $n + \sum_{i=1}^{I} \mathcal{C}_i + T + \ell_{\mathsf{lk}}\mathbb{F}$
$\lvert\pi_{\text{PROTOSTAR}}\rvert$	$O(c \cdot n + T + \ell_{\mathsf{lk}})$	$O(c \cdot n + T + \ell_{\mathsf{lk}})$

Here $\sum_{i=1}^{I} \mathcal{C}_i$ *is the cost of evaluating all circuits on some random input, and* $L(\mathcal{C}_i, d, d^*)$ *is the cost of computing the coefficients of (the vector of) polynomials* $\mathsf{e}(X)$ *(with degree* $d + \log d^*$*) defined in Equation 4.*[5]

Proof. Let $\mathsf{SPS} - \mathsf{IVC}_{d^*}$ be the transformation from a special sound protocol to an IVC-scheme described by Theorem 4. Then given a commitment scheme cm by that theorem $\text{PROTOSTAR}_{d^*} = \mathsf{SPS} - \mathsf{IVC}_{d^*}[\Pi_{\text{mplkup}}]$ is an IVC scheme for predicates expressed in $\mathcal{R}_{\text{mplkup}}$. We apply Theorem 4 to get the efficiencies in the table above. □

Acknowledgments. We want to thank Ariel Gabizon for pointing out an elegant optimization to the generic transformation protocol in Section 3.5, which improves the number of additional hashes executed by the accumulation verifier from $\log \ell$ to 1.

[5] As noted in Theorem 4, $L(\mathcal{C}_i, d, d^*)$ is bounded by $O(n(d + \log d^*)^2)$ field operations, and for certain structures of the gate formula we can do it in $O(n(d + \log d^*)\log^2(d + \log d^*))$.

References

[AFGHO10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. "Structure-Preserving Signatures and Commitments to Group Elements". In: *CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. LNCS. Springer, Heidelberg, Aug. 2010, pp. 209–236. DOI: 10.1007/978-3-642-14623-7_12.

[AFK22] Thomas Attema, Serge Fehr, and Michael Klooß. "Fiat-Shamir Transformation of Multi-round Interactive Proofs". In: *TCC 2022, Part I*. Ed. by Eike Kiltz and Vinod Vaikuntanathan. Vol. 13747. LNCS. Springer, Heidelberg, Nov. 2022, pp. 113–142. DOI: 10.1007/978-3-031-22318-1_5.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. "Verifiable Delay Functions". In: *CRYPTO 2018, Part I*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10991. LNCS. Springer, Heidelberg, Aug. 2018, pp. 757–788. DOI: 10.1007/978-3-319-96884-1_25.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. "Recursive composition and bootstrapping for SNARKS and proof-carrying data". In: *45th ACM STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 111–120. DOI: 10.1145/2488608.2488623.

[BCLMS21] Benedikt Bünz, Alessandro Chiesa, William Lin, Pratyush Mishra, and Nicholas Spooner. "Proof-Carrying Data Without Succinct Arguments". In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 681–710. DOI: 10.1007/978-3-030-84242-0_24.

[BCMS20] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. "Recursive Proof Composition from Accumulation Schemes". In: *TCC 2020, Part II*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12551. LNCS. Springer, Heidelberg, Nov. 2020, pp. 1–18. DOI: 10.1007/978-3-030-64378-2_1.

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. "Scalable Zero Knowledge via Cycles of Elliptic Curves". In: *CRYPTO 2014, Part II*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8617. LNCS. Springer, Heidelberg, Aug. 2014, pp. 276–294. DOI: 10.1007/978-3-662-44381-1_16.

[BGH19] Sean Bowe, Jack Grigg, and Daira Hopwood. *Halo: Recursive Proof Composition without a Trusted Setup*. Cryptology ePrint Archive, Report 2019/1021. https://eprint.iacr.org/2019/1021. 2019.

[BMRS20] Joseph Bonneau, Izaak Meckler, Vanishree Rao, and Evan Shapiro. *Coda: Decentralized Cryptocurrency at Scale*. Cryptology ePrint Archive, Report 2020/352. https://eprint.iacr.org/2020/352. 2020.

[But22] Vitalik Buterin. *The different types of ZK EVM*. https://vitalik.ca/general/2022/08/04/zkevm.html. Accessed: 2023-04-27. 2022.

[CBBZ22] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. *HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates*. Cryptology ePrint Archive, Report 2022/1355. https://eprint.iacr.org/2022/1355. 2022.

[CT10] Alessandro Chiesa and Eran Tromer. "Proof-Carrying Data and Hearsay Arguments from Signature Cards". In: *ICS 2010*. Ed. by Andrew Chi-Chih Yao. Tsinghua University Press, Jan. 2010, pp. 310–331.

[CTV15] Alessandro Chiesa, Eran Tromer, and Madars Virza. "Cluster Computing in Zero Knowledge". In: *EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, Heidelberg, Apr. 2015, pp. 371–403. DOI: 10.1007/978-3-662-46803-6_13.

[EFG22] Liam Eagen, Dario Fiore, and Ariel Gabizon. *cq: Cached quotients for fast lookups*. Cryptology ePrint Archive, Report 2022/1763. https://eprint.iacr.org/2022/1763. 2022.

[GW20] Ariel Gabizon and Zachary J. Williamson. *plookup: A simplified polynomial protocol for lookup tables*. Cryptology ePrint Archive, Report 2020/315. https://eprint.iacr.org/2020/315. 2020.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Report 2019/953. https://eprint.iacr.org/2019/953. 2019.

[Hab22] Ulrich Haböck. *Multivariate lookups based on logarithmic derivatives*. Cryptology ePrint Archive, Report 2022/1530. https://eprint.iacr.org/2022/1530. 2022.

[KB20] Assimakis Kattis and Joseph Bonneau. *Proof of Necessary Work: Succinct State Verification with Fairness Guarantees*. Cryptology ePrint Archive, Report 2020/190. https://eprint.iacr.org/2020/190. 2020.

[KMT22] Dmitry Khovratovich, Mary Maller, and Pratyush Ranjan Tiwari. *MinRoot: Candidate Sequential Function for Ethereum VDF*. Cryptology ePrint Archive, Report 2022/1626. https://eprint.iacr.org/2022/1626. 2022.

[KS22] Abhiram Kothapalli and Srinath Setty. *SuperNova: Proving universal machine executions without universal circuits*. Cryptology ePrint Archive, Report 2022/1758. https://eprint.iacr.org/2022/1758. 2022.

[KS23] Abhiram Kothapalli and Srinath Setty. "HyperNova: Recursive arguments for customizable constraint systems". In: *Cryptology ePrint Archive* (2023).

[KST22] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. "Nova: Recursive Zero-Knowledge Arguments from Folding Schemes". In: *CRYPTO 2022, Part IV*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13510. LNCS. Springer, Heidelberg, Aug. 2022, pp. 359–388. DOI: 10.1007/978-3-031-15985-5_13.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. "Algebraic methods for interactive proof systems". In: *Journal of the ACM (JACM)* 39.4 (1992), pp. 859–868.

[Moh23] Nicholas Mohnblatt. *Sangria: A Folding Scheme for PLONK*. https://github.com/geometryresearch/technical_notes/blob/main/sangria_folding_plonk.pdf. Accessed: 2023-04-27. 2023.

[NT16] Assa Naveh and Eran Tromer. "PhotoProof: Cryptographic Image Authentication for Any Set of Permissible Transformations". In: *2016 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2016, pp. 255–271. DOI: 10.1109/SP.2016.23.

[Ped92] Torben P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". In: *CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. LNCS. Springer, Heidelberg, Aug. 1992, pp. 129–140. DOI: 10.1007/3-540-46766-1_9.

[PK22] Jim Posen and Assimakis A. Kattis. *Caulk+: Table-independent lookup arguments*. Cryptology ePrint Archive, Report 2022/957. https://eprint.iacr.org/2022/957. 2022.

[SAGL18] Srinath Setty, Sebastian Angel, Trinabh Gupta, and Jonathan Lee. "Proving the correct execution of concurrent services in zero-knowledge". In: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 2018, pp. 339–356.

[STW23] Srinath Setty, Justin Thaler, and Riad Wahby. "Customizable constraint systems for succinct arguments". In: *Cryptology ePrint Archive* (2023).

[Val08] Paul Valiant. "Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency". In: *TCC 2008*. Ed. by Ran Canetti. Vol. 4948. LNCS. Springer, Heidelberg, Mar. 2008, pp. 1–18. DOI: 10.1007/978-3-540-78524-8_1.

[Wik21] Douglas Wikström. *Special Soundness in the Random Oracle Model*. Cryptology ePrint Archive, Report 2021/1265. https://eprint.iacr.org/2021/1265. 2021.

[ZBKMNS22] Arantxa Zapico, Vitalik Buterin, Dmitry Khovratovich, Mary Maller, Anca Nitulescu, and Mark Simkin. "Caulk: Lookup Arguments in Sublinear Time". In: *ACM CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM Press, Nov. 2022, pp. 3121–3134. DOI: 10.1145/3548606.3560646.

[ZGKMR22] Arantxa Zapico, Ariel Gabizon, Dmitry Khovratovich, Mary Maller, and Carla Ràfols. *Baloo: Nearly Optimal Lookup Arguments*. Cryptology ePrint Archive, Report 2022/1565. https://eprint.iacr.org/2022/1565. 2022.