Protostar: Generic Efficient Accumulation/Folding for Special-Sound Protocols

Benedikt Bünz[1,2] and Binyi Chen[1(✉)]

[1] Espresso Systems, Middletown, USA
binyi@espressosys.com
[2] New York University, New York, USA

Abstract. Accumulation is a simple yet powerful primitive that enables incrementally verifiable computation (IVC) without the need for recursive SNARKs. 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 $k+2$ elliptic curve multiplications and $k+d+O(1)$ field/hash 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. Protostar is 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 a hash of d^* field elements, where d^* is the degree of the highest gate. 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.

1 Introduction

Incrementally Verifiable Computation [30] 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 [12], have many applications, including distributed computation [3,13], blockchains [5,18], verifiable delay functions [4], verifiable photo editing [25], and SNARKs for machine-computations [2]. An IVC-based VDF construction is the current candidate VDF for Ethereum [19]. 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 [10].

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

J. Guo and R. Steinfeld (Eds.): ASIACRYPT 2023, LNCS 14439, pp. 77–110, 2023.
https://doi.org/10.1007/978-981-99-8724-5_3

by Halo [6] and has led to a series of significant advances [8,9,22]. 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 [6,9]. Even more surprising, this process does not even require a SNARK but instead can be instantiated with a non-succinct NARK [8], 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 [8,22]. 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 [16]. These lookup gates enable looking up that a circuit value is in some table, simplifying range proofs and bit-operations. These R1CS-based accumulation schemes contrast non-IVC SNARK developments, with an increased focus on high-degree gate [11,16] and lookup support [15]. 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 [14,27,33,34].

More Expressive Accumulation. There have been efforts to build accumulation schemes that overcome the limitations of fixed R1CS. SuperNova [21] 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 still requires many (though a constant number of) hashes and a hash-to-group gadget, and additionally, the accumulator, and thus the final proof, is still linear in the total size of all instances.

Sangria [24] describes an accumulation scheme for a Plonk-like [16] constraint system with degree-2 gates. It also proposes a solution for higher-degree gates in the future work section but without security proof. Moreover, as the gate degree d increases, the number of group operations in Sangria grows by a factor of d, which cancels out the advantages of using the more expressive high-degree

[1] Unrelated to set accumulators.

gates. Origami [35] recently introduced a folding scheme for lookups using a product check and degree 7 polynomials. 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. [1,31]
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 [8] 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 [22]. We can further reduce the size of the recursive circuit to only $k + 2$ group scalar multiplication, by compressing all verification equations using a random linear combination.

Efficient Simple Protocols for $\mathcal{R}_{mplkup}$. 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 [26], 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 constrains **pc**; e.g.,

[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.

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 3 group scalar multiplications and $d + O(1)$ hash and 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 1 additional challenge computations within the recursive circuit.

Table 1. The comparison between IVCs.

	Protostar	HyperNova	SuperNova
Language	Degree d Plonk/CCS	Degree d CCS	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	$3\mathbb{G}$ $(d + O(1))\mathsf{H} + \mathsf{H}_{\mathsf{in}}$ $(d + O(1))\,\mathbb{F}$	$1\mathbb{G}$ $d \log n\mathsf{H} + \mathsf{H}_{\mathsf{in}}$ $O(d \log n)\mathbb{F}$	$2\mathbb{G}$ $\mathsf{H}_{\mathsf{in}} + O(1)\mathsf{H} + 1\mathsf{H}_{\mathbb{G}}$
extra P recursive w/ lookup	$1\mathsf{H}$	$O(\log T)\,\mathsf{H}$ $O(\ell_{\mathsf{lk}} \log T)\,\mathbb{F}$	N/A

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 [17] 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 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 SuperNova [21] and HyperNova [20], in Table 1 (for more detail see Corollary 1): P native is the running time of

the accumulation prover and P recursive refers to the cost of implementing the accumulation verifier as a circuit. In the table, $|\mathbf{w}|$ is the number of non-zero entries 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. $d\,\mathsf{H}$ denotes the cost of hashing d λ-bit numbers. We assume that the cost scales linearly with the size of the input and output. In Protostar d field elements are hashed once and in HyperNova d field elements are hashed $\log(n)$ times. $\mathsf{H}_{\mathbb{G}}$ is the cost of a hash-to-group function. H_{in} is the cost of hashing the public input and the accumulator instance. Note that the $O(1)\,\mathsf{H}$ 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 [28].

Concurrent Work. In a paper concurrent with this work, Kothapalli and Setty [20] introduce an IVC for high degree relations. They use a generalization of R1CS called customizable constraint systems (CCS) [29] 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, but we subsequently showed that our compiler can also be directly applied to CCS (See full version [7]). 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/CCS in their IVC scheme or provide any explicit constructions for non-uniform computations. Their scheme is built using sumchecks [23] and the resulting IVC recursive circuit is dominated by 1 group scalar multiplication, $d\log n + \ell_{\mathsf{in}}$ hash operations 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, even with lookup and non-uniformity support, is only dominated by 3 group scalar multiplications and $O(\ell_{\mathsf{in}} + d)$ field/hash operations, entirely independent of n. The 2 additional group operations compared to HyperNova are likely offset by the additional lookup support [32] and the significantly fewer hashes and non-native field operations (d vs. $d\log(n)$). 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. Moreover, the lookup support adds almost no overhead to the IVC scheme for high-degree Plonk relations. In particular, it adds no group scalar 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 [8] to obtain the IVC/PCD scheme for relation $\mathcal{R}$. We describe the workflow in Fig. 1.

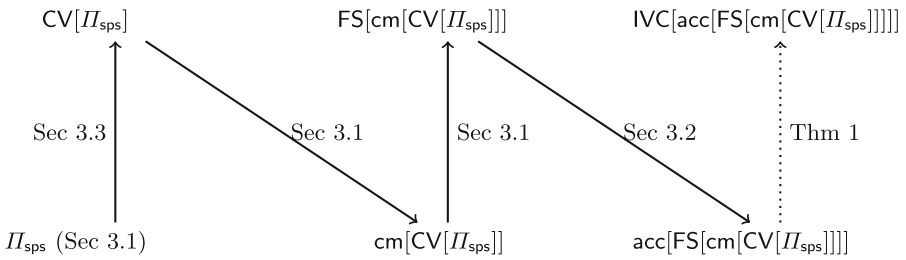

Fig. 1. The workflow for building an IVC from a special sound protocol. We start from a special-sound protocol Π_{sps} for an NP-complete relation $\mathcal{R}_{\mathsf{NP}}$, and transform it to $\mathsf{CV}[\Pi_{\mathsf{sps}}]$ with a compressed verifier check. $\mathsf{CV}[\Pi_{\mathsf{sps}}]$ is converted to a NARK $\mathsf{FS}[\mathsf{cm}[\mathsf{CV}[\Pi_{\mathsf{sps}}]]]$ via commit-and-open and the Fiat-Shamir transform. We then build a generic accumulation scheme for the NARK and apply Theorem 1 from [8] to obtain the IVC scheme. This last connection is dotted as it requires heuristically replacing random oracles with cryptographic hash functions.

The paper begins by describing the compiler from special-sound protocols to NARKs in Sect. 3, and presents an efficient accumulation scheme for the compiled NARK verifier in Sect. 3.2. Next, we describe simple and efficient special-sound protocols for Plonkup circuit-satisfiability relations and extend it to support non-uniform computation in Sect. 5. Similarly, we extend the CCS relation [29] to support non-uniform computation and lookup (see full version [7]). 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 $a_i \cdot b_i = c_i$ 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 [8], 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 [8] to obtain an efficient IVC scheme with predicates expressed in $\mathcal{R}$.

In Theorem 3, we show that for any $(2k-1)$-move[3] special-sound protocols with degree-d verifiers, the resulting IVC recursive circuit only involves $k + d + O(1)$ hashes, $k+1$ non-native field operations and $k+d-1$ commitment group scalar multiplications. We also introduce a generic approach for further reducing the number of group operations to $k+2$ in Sect. 3.3. This is favorable for $d \geq 3$. The idea is to compress all ℓ degree d verification checks into a single verification check using a random linear combination with powers of a challenge β. This means that error-correcting terms are field elements and, thus, can be sent directly without committing to them. The prover also sends a single commitment to powers of β and powers of $\beta^{\sqrt{\ell}}$. The verification equation uses one power of β and one power of $\beta^{\sqrt{\ell}}$, which increases the degree of the verification check to $d+2$. The verifier also checks the correctness of the powers of β using $2\sqrt{\ell}$ degree 2 checks.

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

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. We describe the components of the special-sound protocol in Fig. 2. Note we also extend CCS relation [29] to support lookup and non-uniform computation and build a special-sound protocol for it (See Fig. 2). Recall that a Plonkup circuit-satisfiability relation consists of three modular relations, namely, (i) a high-degree gate relation checking that 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.

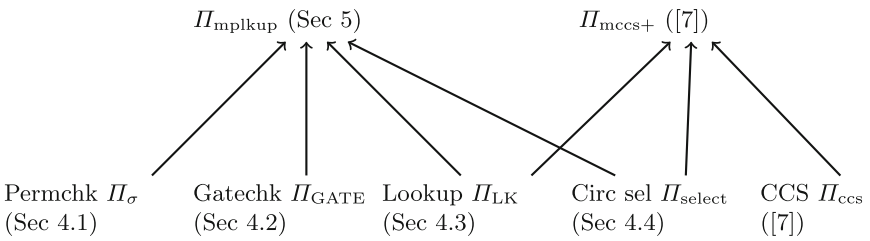

Fig. 2. The special-sound protocols for ProtoStar and ProtoStar$_{\text{ccs}}$. The special-sound protocol Π_{mplkup} for the multi-circuit Plonkup relation $\mathcal{R}_{\text{mplkup}}$ consists of the sub-protocols for permutation, high-degree custom gate, lookup, and circuit selection relations. The special-sound protocol $\Pi_{\text{mccs+}}$ for the extended CCS relation $\mathcal{R}_{\text{mccs+}}$ consists of the sub-protocols for lookup, circuit selection, as well as the CCS relation [29]. From Π_{mplkup} or $\Pi_{\text{mccs+}}$, we can apply the workflow described in Fig. 1 to obtain the IVC schemes ProtoStar or ProtoStar$_{\text{ccs}}$.

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 [17], in Sect. 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. However, the prover work for computing the error terms is not independent of the table size because the accumulator is not sparse. Fortunately, we observe that the prover can efficiently update the error term commitments without recomputing the error term vectors from scratch,

thus preserving the efficiency of the accumulation prover. Moreover, we extend Π_{LK} in Sect. 4.3 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 Sect. 5, 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 $\mathcal{R}_{\mathrm{mplkup}}$ 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 $\mathcal{R}_{\mathrm{mplkup}}$ 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 $2I$ degree 2 equations to check that $b_{pc} = 1$ and $b_i = 0 \forall i \neq pc$. Additionally, each algebraic equation $\mathcal{G}$ checked in Π_{plonkup} is replaced with $\sum_{i=1}^{I} \mathcal{G}_i \cdot b_i$ where $\mathcal{G}_i$ $(1 \leq i \leq I)$ is the corresponding gate in the i-th branch circuit. The resulting special-sound protocol has 3 moves, and the verifier degree is $d+1$, where d is the highest degree of the custom gates. This means that the IVC scheme for the non-uniform Plonkup relation adds negligible overhead to that for the Plonkup relation.

2 Preliminaries

The definitions of special-sound protocols and non-interactive arguments follow from [1]. We defer the definition of Fiat-Shamir transform and commitment schemes to the full version [7].

Lemma 1 (Fiat-Shamir transform of Special-sound Protocols [1]). *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.

2.1 Incremental Verifiable Computation (IVC)

We adapt and simplify the definition from [8,22].

Definition 1 (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:*

- $\mathsf{P}_{\mathsf{IVC}}(m, z_0, z_m, z_{m-1}, \mathbf{w}_{\mathsf{loc}}, \pi_{m-1}]) \to \pi_m$. *The IVC prover $\mathsf{P}_{\mathsf{IVC}}$ takes as input a program output z_m at step m, local data $\mathbf{w}_{\mathsf{loc}}$, initial input z_0, previous program 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 adversarial completeness if for any function predicate* ϕ *expressible in* $\mathcal{R}_{\mathsf{NP}}$*, and any, possibly adversarially created,* $(m, z_0, z_m, , z_{m-1}, \mathbf{w}_{\mathsf{loc}}, \pi_{m-1})$ *such that*

$$\phi(z_0, z_m, z_{m-1}, \mathbf{w}_{\mathsf{loc}}) \wedge (\mathsf{V}_{\mathsf{IVC}}(m-1, z_0, z_{m-1}, \pi_{m-1}) = 0)$$

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

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*

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

Here m *is a constant.*

Efficiency. The runtime of $\mathsf{P}_{\mathsf{IVC}}$ *and* $\mathsf{V}_{\mathsf{IVC}}$ *as well as the size of* π_{IVC} *only depend on* $|\phi|$ *and are independent on the number of iterations.*

Recently, [21] introduced the notion of non-uniform IVC, where the predicate ϕ is selected from a fixed set of predicates at every step of the computation. The selection depends on the current state of the computation. Non-uniform IVC fits into our model by simply setting the predicate to be the union of all predicates, including the selection circuit. The one key difference is an additional efficiency requirement that the IVC prover in step i only depends on the size of the predicate that is being executed in step i. Our Protostar construction achieves this requirement.

2.2 Simple Accumulation

We take definitions and proofs from [8].

Definition 2 (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.

- $\mathsf{V}_{\mathsf{acc}}(\mathsf{pi}, \pi.x, \mathsf{acc}.x, \mathsf{acc}'.x, \mathsf{pf}) \rightarrow v$. *The accumulation verifier takes as input the statement* pi*, the instances of the NARK proof, the old and new accumulator, the correction terms, and 'accepts' by outputting* 0 *and 'rejects' otherwise.*
- $D(\mathsf{acc}) \rightarrow v$. *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 [8]). *Given a standard-model NARK for circuit-satisfiability and a standard-model accumulation scheme (Definition 2) for that NARK, both with negligible knowledge error, there exists an efficient transformation that outputs an IVC scheme (see Sect. 3.2 of [8]) 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.

Definition 3 (Fiat-Shamir Heuristic). *The Fiat-Shamir Heuristic, relative to a secure cryptographic hash function* H*, states that a random oracle NARK with negligible knowledge error yields a NARK that has negligible knowledge error in the standard (CRS) model if the random oracle is replaced with* H.

Complexity. The IVC transformation from [8] 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 [22].

PCD. IVC can be generalized to arbitrary DAGs instead of just path graphs in a primitive called proof-carrying data [3]. Accumulation schemes can be compiled into full PCD if they support accumulating an arbitrary number of accumulators and proofs [8,9]. 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 [22] constructs a custom, concretely efficient SNARK for their accumulation/folding scheme.

3 Protocols

3.1 Special-Sound Protocols and Their Basic Transformations

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 output length ℓ (i.e. the verifier checks ℓ degree d algebraic equations). In each round i $(1 \leq i \leq k)$, 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 $r_i \in \mathbb{F}$. 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.

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

where $f_j^{\mathsf{V}_{\mathsf{sps}}}$ is a homogeneous degree-j algebraic map that outputs a vector of ℓ field elements.

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 $(a_1, \ldots, a_\mu)$-special-sound). *Let Π_{sps} be an $(a_1, \ldots, a_\mu)$-out-of-N 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 $(a_1, \ldots, a_\mu)$-out-of-N special-sound protocol for $\mathcal{R}^{\mathcal{R}}_{\mathsf{cm}}$.*

We defer the proof to the full version [7].

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 $\mathsf{FS}[\Pi_{\mathsf{cm}}]$ of the protocol Π_{cm} is the following. The prover generates the round challenges by computing ρ_{NARK} on input the challenge and the prover message commitment in the previous round. The prover then sends the proof as the list of prover messages and the corresponding commitments. The verifier checks the proof by recomputing the challenges and runs the verifier for Π_{cm}. By Lemma 1, $\mathsf{FS}[\Pi_{\mathsf{cm}}]$ is knowledge sound if Π_{sps} is special-sound.

3.2 Accumulation Scheme for $\mathsf{V}_{\mathsf{NARK}}$

Let ρ_{acc} and ρ_{NARK} be two random oracles, and let $\mathsf{V}_{\mathsf{NARK}}$ be the verifier of $\mathsf{FS}[\Pi_{\mathsf{cm}}]$ in Sect. 3.1, 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 $\mathsf{FS}[\Pi_{\mathsf{cm}}]$ for relation $\mathcal{R}$. Namely, given public input $\mathsf{pi} \in \mathcal{M}^{\ell_{\mathsf{in}}}$, random challenges $[r_i]_{i=1}^{k-1} \in \mathbb{F}^{k-1}$, 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}, and a slack variable μ, 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)

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

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

Remark 1. 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, $[r_i]_{i=1}^{k-1} \in \mathbb{F}^{k-1}$ 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 Fig. 3.

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 Fig. 3.

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 Fig. 4.

Theorem 2. *Let* $(\mathsf{P}_{NARK}, \mathsf{V}_{NARK})$ *be the RO-NARK defined in Sect. 3.1. Let* $\mathsf{cm} = (\mathsf{Setup}, \mathsf{Commit})$ *be a binding, homomorphic commitment scheme. Let* ρ_{acc} *be another random oracle. The accumulation scheme* $(\mathsf{P}_{\mathsf{acc}}, \mathsf{V}_{\mathsf{acc}}, D_{\mathsf{acc}})$ *for* V_{NARK} *satisfies perfect completeness and has knowledge error* $(Q+1)\frac{d+1}{|\mathbb{F}|} + \mathsf{negl}(\lambda)$ *as defined in Definition 2, against any randomized polynomial-time* Q*-query adversary.*

Proof. **Completeness:** 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 2.

$\mathsf{V}_{\mathsf{acc}}$ 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

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

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

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

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

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

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

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 $\mathsf{acc}'.x.E' \stackrel{?}{=} \mathsf{acc}.x.E + \sum_{j=1}^{d-1} \alpha^j \cdot E_j$.

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

Fig. 4. Accumulation Decider for low-degree Fiat-Shamired NARKs

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

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

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 $E' = \mathsf{Commit}(\mathsf{ck}, \mathbf{e}')$.

Knowledge-Soundness: We show that the scheme has knowledge-soundness 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 2. 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

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

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

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

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

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

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 $(Q+1) \cdot \frac{d+1}{|\mathbb{F}|} + \mathsf{negl}(\lambda)$. This implies that acc is an accumulation scheme with $((Q+1) \cdot \frac{d+1}{|\mathbb{F}|} + \mathsf{negl}(\lambda))$-knowledge soundness. □

3.3 Compressing Verification Checks 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 $\mathbf{e}_j \in \mathbb{F}^\ell$ $(1 \leq j < d)$; 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 $\mathsf{CV}[\Pi_{\mathsf{sps}}]$ for the same relation $\mathcal{R}$. This optimization compresses the ℓ degree-d equations checked by the verifier into a single degree-$(d+2)$ equation using a random linear combination, with the tradeoff of additionally checking $2\sqrt{\ell}$ degree-2 equations. We describe the generic transformation below.

Compressing Verification Checks. W.l.o.g. assume ℓ is a perfect square, then we can transform Π_{sps} into a special-sound protocol $\mathsf{CV}[\Pi_{\mathsf{sps}}]$ where the $\mathsf{V}_{\mathsf{sps}}$ reduces from ℓ degree-d checks to 1 degree-$(d+2)$ check and additionally $2\sqrt{\ell}$ degree-2 checks. Instead of checking the output of $\mathsf{V}_{\mathsf{sps}}$ to be ℓ zeroes, we take a random linear combination of the ℓ verification equations using powers of a challenge β. For example, if the map is $\mathsf{V}_{\mathsf{sps}}(x_1, x_2) := (\mathsf{V}_{\mathsf{sps},1}(x_1, x_2), \mathsf{V}_{\mathsf{sps},2}(x_1, x_2)) = (x_1 + x_2, x_1 x_2)$ we can set the new algebraic map as $\mathsf{V}'_{\mathsf{sps}}(x_1, x_2, \beta) := \mathsf{V}_{\mathsf{sps},1}(x_1, x_2) + \beta \cdot \mathsf{V}_{\mathsf{sps},2}(x_1, x_2) = (x_1 + x_2) + \beta x_1 x_2$ for a random β. Doing this naively reduces the output length to 1 but also requires the verifier to compute the appropriate powers of β. This would increase the degree by ℓ, an undesirable tradeoff. To mitigate this, we can have the prover precompute powers of β, i.e. $\beta, \beta^2, \ldots, \beta^\ell$ and send them to the verifier. The verifier then only needs to check consistency between the powers of β, which can be done using a degree 2 check, e.g. $\beta^{i+1} = \beta^i \cdot \beta$ and the degree d verification equation increases in degree by 1. This mitigates the degree increase but requires the prover to send another message of length ℓ. To achieve a more optimal tradeoff, we write each $i = j + k \cdot \sqrt{\ell}$ for $j, k \in [1, \sqrt{\ell}]$. The prover then sends $\sqrt{\ell}$ powers of β and $\sqrt{\ell} - 1$ powers of $\beta^{\sqrt{\ell}}$. From these, each power of β from 1 to ℓ can be recomputed using just one multiplication. This results in the prover sending an additional message of length $2\sqrt{\ell}$, the original ℓ verification checks being transformed into a single degree $d+2$ check and additionally $2\sqrt{\ell}$ degree 2 checks for the consistency of the powers of β.

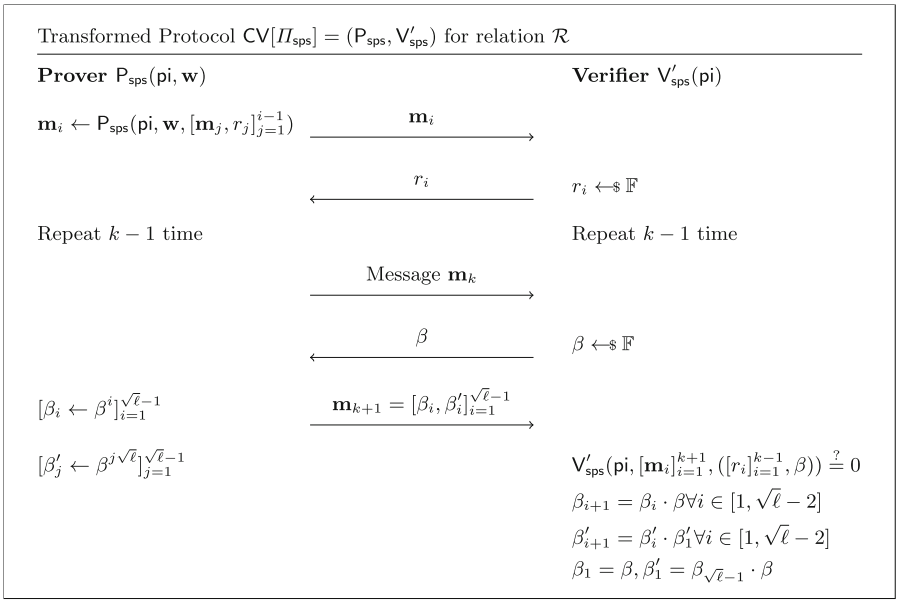

Fig. 5. Compressed verification of Π_{sps}.

We describe the transformed protocol in Fig. 5, where

$$\mathsf{V}'_{\mathsf{sps}}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^{k+1}, ([r_i]_{i=1}^{k-1}, \beta)) := \sum_{i=0}^{\sqrt{\ell}-1} \sum_{j=0}^{\sqrt{\ell}-1} \beta_i \cdot \beta'_j \cdot \mathsf{V}_{\mathsf{sps},i+j\sqrt{\ell}}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^{k}, [r_i]_{i=1}^{k-1})$$
$$= \sum_{j=0}^{\ell-1} \beta^j \cdot \mathsf{V}_{\mathsf{sps},j}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^{k}, [r_i]_{i=1}^{k-1})$$

and $\mathsf{V}_{\mathsf{sps},j}(\mathsf{pi}, [\mathbf{m}_i]_{i=1}^{k}, [r_i]_{i=1}^{k-1})$ is the $(j+1)$-th $(0 \leq j < \ell)$ equation checked by $\mathsf{V}_{\mathsf{sps}}$. The transformed protocol is a $(2k+1)$-move special-sound protocol for the same relation $\mathcal{R}$. The transformed verifier now checks 1 degree-$(d+2)$ equation and additionally $2\sqrt{\ell}$ degree-2 equations.

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. The transformed protocol $\mathsf{CV}[\Pi_{\mathsf{sps}}]$ of Π_{sps} is $(a_1, \ldots, , a_{k-1}, \ell)$-special-sound.*

We defer the proof to the full version [7].

High-Low Degree Accumulation. After the transformation, the error vectors $\mathbf{e}_j$ $(1 \leq j \leq d+1)$ become single field elements, and we can use the trivial commitment $E_j := \mathsf{Commit}(\mathsf{ck}, e_j) := e_j$ without group operations. Additionally, we can use a separate error vector $\mathbf{e}' \in \mathbb{F}^{2\sqrt{\ell}}$ to keep track of the error terms for the $2\sqrt{\ell}$ degree-2 checks, and set $E' := \mathsf{Commit}(\mathsf{ck}, \mathbf{e}') \in \mathbb{G}$ to be the corresponding error commitment. The accumulation prover only needs to perform $O(\sqrt{\ell})$ additional group operations to commit $\mathbf{m}_{k+1}$ and $\mathbf{e}'$, and compute the coefficients of a degree-$(d+2)$ univariate polynomial, which is described as the sum of $O(\ell)$ polynomials. The accumulator instance needs to include one more challenge β and two commitments (for $\mathbf{m}_{k+1}$ and $\mathbf{e}'$). The accumulator verifier needs to do only $k+2$ (rather than $k+d-1$) group scalar multiplications, with the tradeoff of 1 more hash and $O(d)$ more field operations. This high-low degree accumulation is described in detail in the full version [7].

Theorem 3 (IVC for high-degree special-sound protocols). *Let $\mathbb{F}$ be a finite field, such that $|\mathbb{F}| \geq 2^\lambda$ and $\mathsf{cm} = (\mathsf{Setup}, \mathsf{Commit})$ be a binding homomorphic commitment scheme for vectors in $\mathbb{F}$. Let $\Pi_{\mathsf{sps}} = (\mathsf{P}_{\mathsf{sps}}, \mathsf{V}_{\mathsf{sps}})$ be a special-sound protocol for an NP-complete relation $\mathcal{R}_{\mathsf{NP}}$ with the following properties:*

- *It's $(2k-1)$ move.*
- *It's $(a_1, \ldots, a_{k-1})$-out-of-$|\mathbb{F}|$ special-sound. Such that the knowledge error $\kappa = 1 - \prod_{i=1}^{k-1}(1 - \frac{a_i}{|\mathbb{F}|}) = \mathsf{negl}(\lambda)$*
- *The inputs are in $\mathbb{F}^{\ell_{\mathsf{in}}}$*
- *The verifier is degree $d = \mathsf{poly}(\lambda)$ with output in $\mathbb{F}^\ell$*

Then, under the Fiat-Shamir heuristic for a cryptographic hash function H (Definition 3), there exist two IVC schemes $\mathsf{IVC} = (\mathsf{P}_{\mathsf{IVC}}, \mathsf{V}_{\mathsf{IVC}})$ and $\mathsf{IVC}_{\mathsf{CV}} = (\mathsf{P}_{\mathsf{CV,IVC}}, \mathsf{V}_{\mathsf{CV,IVC}})$ with predicates expressed in $\mathcal{R}_{\mathsf{NP}}$ with the following efficiencies:

	No CV	CV
$\mathsf{P}_{\mathsf{IVC}}$ *native*	$\sum_{i=1}^{k} \lvert\mathbf{m}_i^*\rvert + (d-1)\ell\mathbb{G}$ $\mathsf{P}_{\mathsf{sps}} + L(\mathsf{V}_{\mathsf{sps}}, d)$	$\sum_{i=1}^{k} \lvert\mathbf{m}_i^*\rvert + O(\sqrt{\ell})\mathbb{G}$ $\mathsf{P}_{\mathsf{sps}} + L'(\mathsf{V}_{\mathsf{sps}}, d+2)$
$\mathsf{P}_{\mathsf{IVC}}$ *recursive*	$k + d - 1\mathbb{G}$ $k + \ell_{\mathsf{in}}\mathbb{F}$ $(k + d + O(1))\mathsf{H} + 1\mathsf{H}_{\mathsf{in}}$	$k + 2\mathbb{G}$ $k + \ell_{\mathsf{in}} + d + 1\mathbb{F}$ $(k + d + O(1))\mathsf{H} + 1\mathsf{H}_{\mathsf{in}}$
$\mathsf{V}_{\mathsf{IVC}}$:	$\ell + \sum_{i=1}^{k} \lvert\mathbf{m}_i\rvert\mathbb{G}$ $\mathsf{V}_{\mathsf{sps}}$	$O(\sqrt{\ell}) + \sum_{i=1}^{k} \lvert\mathbf{m}_i\rvert\mathbb{G}$ $O(\ell) + \mathsf{V}_{\mathsf{sps}}$
$\lvert\pi_{\mathsf{IVC}}\rvert$:	$k + \ell_{\mathsf{in}}\mathbb{F}$ $k + 1\mathbb{G}$ $\sum_{i=1}^{k} \lvert\mathbf{m}_i\rvert$	$k + \ell_{\mathsf{in}} + 1\mathbb{F}$ $k + 2\mathbb{G}$ $\sum_{i=1}^{k} \lvert\mathbf{m}_i\rvert + O(\sqrt{\ell})$

The first row displays the native operations of the IVC *prover (i.e., the complexity of running the accumulation prover). The second row describes the size of the recursive statement representing the accumulation verifier 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,* $|\mathbf{m}_i|$ *denotes the prover message length;* $|\mathbf{m}_i^*|$ *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.* H *denotes the total input length to a cryptographic hash, and* H_{in} *is the hash to the public input and accumulator instance.* $\mathsf{P}_{\mathsf{sps}}$ *(and* $\mathsf{V}_{\mathsf{sps}}$*) is the cost of running the prover (and the algebraic verifier) of the special-sound protocol, respectively.* $L(\mathsf{V}_{\mathsf{sps}}, d)$ *is the cost of computing the coefficients of the degree* d *polynomial*

$$\mathbf{e}(X) := \sum_{j=0}^{d} (\mu + X)^{d-j} \cdot f_j^{\mathsf{V}_{\mathsf{sps}}}(\mathsf{acc} + X \cdot \pi)\,, \tag{2}$$

and $L'(\mathsf{V}_{\mathsf{sps}}, d+2)$ *is the cost of computing the coefficients of the degree* $d+2$ *polynomial*

$$e(X) := \sum_{a=0}^{\sqrt{\ell}-1} \sum_{b=0}^{\sqrt{\ell}-1} (X \cdot \pi.\beta_a + \mathsf{acc}.\beta_a)(X \cdot \pi.\beta'_b + \mathsf{acc}.\beta'_b) \sum_{j=0}^{d} (\mu + X)^{d-j} \cdot f_{j,a+b\sqrt{\ell}}^{\mathsf{V}_{\mathsf{sps}}}(\mathsf{acc} + X \cdot \pi)\,, \tag{3}$$

where all inputs are linear functions in a formal variable X[4]*, and* $f_{j,i}^{\mathsf{V}_{\mathsf{sps}}}$ *is the* i*th* $(0 \le i \le \ell - 1)$ *component of* $f_j^{\mathsf{V}_{\mathsf{sps}}}$*'s output. For the proof size,* $\mathbb{G}$ *and* $\mathbb{F}$ *are the number of commitments and field elements, respectively.*

[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 $O(d \log^2 d)$ [11].

Proof. The construction first defines the two NARKs

$$\Pi_{\mathsf{NARK}} = (\mathsf{P}_{\mathsf{NARK}}, \mathsf{V}_{\mathsf{NARK}}) = \mathsf{FS}[\mathsf{cm}[\Pi_{\mathsf{sps}}]],$$

and

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

Then we construct the accumulation scheme $(\mathsf{P}_{\mathsf{acc}}, \mathsf{V}_{\mathsf{acc}}) = \mathsf{acc}[\Pi_{\mathsf{NARK}}]$ using the accumulation scheme from Sect. 3.2 and $(\mathsf{P}_{\mathsf{acc},\mathsf{HL}}\mathsf{V}_{\mathsf{acc},\mathsf{HL}}) = \mathsf{acc}_{\mathsf{HL}}[\Pi_{\mathsf{NARK},\mathsf{CV}}]$ using the accumulation scheme described in Sect. 3.3. Then we apply the transformation from Theorem 1 to construct the IVC schemes IVC and $\mathsf{IVC}_{\mathsf{CV}}$.

Security: By Lemmas 1, 2, we have that Π_{NARK} has $(Q+1)\cdot\left[1-\prod_{i=1}^{k-1}(1-\frac{a_i}{|\mathbb{F}|})\right]$ 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 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 $\kappa = (Q+1)\cdot\left[1-\prod_{i=1}^{k-1}(1-\frac{a_i}{|\mathbb{F}|})\right] + \mathsf{negl}(\lambda)$ for $\mathcal{R}_{\mathsf{NP}}$. Analogously and using Lemma 3, $\Pi_{\mathsf{NARK},\mathsf{CV}}$ has knowledge soundness with knowledge error $\kappa' = (Q+1)\cdot\left[1-(1-\frac{\ell}{|\mathbb{F}|})\prod_{i=1}^{k-1}(1-\frac{a_i}{|\mathbb{F}|})\right] + \mathsf{negl}(\lambda)$ for $\mathcal{R}_{\mathsf{NP}}$. By assumption, κ and κ' are negligible in λ. Using Theorem 2 and the high-low degree accumulation scheme described previously, we can construct accumulation schemes acc and $\mathsf{acc}_{\mathsf{CV}}$ for Π_{NARK} and $\Pi_{\mathsf{NARK},\mathsf{CV}}$, respectively. The accumulation schemes have negligible knowledge error as $d = \mathsf{poly}(\lambda)$. Under the Fiat-Shamir heuristic for H we can turn the NARKs and the accumulation schemes into secure schemes in the standard model.

By Theorem 1, this yields IVC and $\mathsf{IVC}_{\mathsf{CV}}$, secure IVC schemes with predicates expressed in $\mathcal{R}_{\mathsf{NP}}$.

Efficiency: We first analyze the efficiency for IVC. 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 all error terms $\mathbf{e}_1, \ldots, \mathbf{e}_{d-1}$ and commit to them. The error terms are computed by symbolically evaluating the polynomial $e(X)$ in Eq. 3 with linear functions as inputs. The recursive circuit combines a new proof $\pi.x$ with an accumulator $\mathsf{acc}.x$. The size of the accumulator instance is ℓ_{in} field elements for the input, $k-1$ 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 $d-1$ commitments for the error terms. The IVC verifier checks the correctness of the commitments and runs $\mathsf{V}_{\mathsf{sps}}$.

For $\mathsf{IVC}_{\mathsf{CV}}$, the prover needs to additionally commit to a message $\mathbf{m}_{k+1}$ with length $O(\sqrt{\ell})$; the number of error terms also increases from $d-1$ to $d+1$. Fortunately, 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 $d+1$ error terms when using CV. However, there is another separate error term $\mathbf{e}' \in \mathbb{F}^{2\sqrt{\ell}}$ for the additional $O(\sqrt{\ell})$ degree-2 checks, thus the prover needs

to commit to $E' = \mathsf{Commit}(\mathbf{e}')$. The size of the accumulator instance is ℓ_{in} field elements for the input, k field elements for the interactive-proof challenges, 1 field element for the accumulator challenge, $k+1$ commitments for the prover messages, $d+1$ field elements for the error terms of the high-degree checks, and 1 commitment for the additional error term $\mathbf{e}'$. □

3.4 Computation of Error Terms

We now give an explicit algorithm for efficiently computing the error terms, that is, computing the polynomial $e(X)$ as defined in (3) (the degree of $e(X)$ is $d' = d+2$). The algorithm has similarities with computing the round polynomials in a single round of the sumcheck protocol [23].

1. For each $i = 0$ to d define

$$e^{(i)}(X) := \sum_{a=0}^{\sqrt{\ell}-1} \sum_{b=0}^{\sqrt{\ell}-1} (X \cdot \pi.\beta_a + \mathsf{acc}.\beta_a)(X \cdot \pi.\beta'_b + \mathsf{acc}.\beta'_b) \cdot f^{\mathsf{V}_{\mathsf{sps}}}_{i,a+b\sqrt{\ell}}(\mathsf{acc} + X \cdot \pi) \tag{4}$$

2. Compute $e^{(i)}(j)$ for all $j \in [0, i+2]$. Use these evaluations to interpolate $e^{(i)}(X)$ using fast interpolation methods, e.g. an iFFT
3. Compute the coefficient form of $e(X) = \sum_{i=0}^{d} e^{(i)}(X) \cdot (\mu + X)^{d-i}$. This is done by computing the coefficients of $e^{(i)}(X) \cdot (\mu + X)^{d-i}$ for every $i \in [0, d]$ using FFTs, and recover $e(X)$ using coefficient-wise addition. The complexity is $O(d^2 \log d)$.

In the worst case, this algorithm is equivalent to evaluating the circuit at $d+2$ different inputs. However, it can perform much better in practice. The reason is that many of the n gates may only be low degree. E.g. 90% of the gates are degree 1 or 2 addition and multiplication gates, and 10% are more high degree gates. Then the prover only has to evaluate the 10% of the circuit at $d+2$ points and 90% of the circuit only at 4 points. Note that the selector polynomials are static in the classification of NP plonkup. This means that each gate has precisely the degree of the active component. This stands in contrast to relations such as high-degree Plonk, where the selectors are pre-processed, and the selectors are preprocessed witnesses. In Plonk and related systems, each gate essentially has the same degree.

Dealing with Branched Gates. In some scenarios, the NARK proof π has the property that each gate $f^{\mathsf{V}_{\mathsf{sps}}}_{i,a+b\sqrt{\ell}}(\mathsf{acc} + X \cdot \pi)$ in Formula 4 can be represented as the sum of I parts where at most one part is related to π, that is, for some gates $g_1, \ldots, g_I$ and some index $pc \in [I]$,

$$f^{\mathsf{V}_{\mathsf{sps}}}_{i,a+b\sqrt{\ell}}(\mathsf{acc} + X \cdot \pi) = g_{pc}(\mathsf{acc} + X \cdot \pi) + \sum_{j \in [I] \setminus \{pc\}} g_j(\mathsf{acc})\,.$$

In this case, for any gate $f^{\mathsf{V}_{\mathsf{sps}}}_{i,a+b\sqrt{\ell}}$, we present a caching algorithm for evaluating $f^{\mathsf{V}_{\mathsf{sps}}}_{i,a+b\sqrt{\ell}}(\mathsf{acc}+k\cdot\pi)$ at all evaluation points $k\in[0,i+2]$. The complexity is only proportional to the evaluation complexity of g_{pc} rather than $f^{\mathsf{V}_{\mathsf{sps}}}_{i,a+b\sqrt{\ell}}$.

1. For every $j\in[I]$, initialize $U_j := g_j(\mathsf{acc})$, and store $V := \sum_{j=1}^{I} U_j$.
2. Upon receiving a new NARK proof π during accumulation, for every $k\in[0,i+2]$, compute $f^{\mathsf{V}_{\mathsf{sps}}}_{i,a+b\sqrt{\ell}}(\mathsf{acc}+k\cdot\pi) = V + g_{pc}(\mathsf{acc}+k\cdot\pi) - U_{pc}$.
3. After the accumulation, let $\alpha\in\mathbb{F}$ be the folding challenge and let $U'_{pc} = g_{pc}(\mathsf{acc}+\alpha\cdot\pi)$, update $V \leftarrow V + U'_{pc} - U_{pc}$ and update $U_{pc}\leftarrow U'_{pc}$.

The algorithm is correct because V is always $\sum_{j\in[I]} g_j(\mathsf{acc})$ where acc is the current accumulator.

4 Special-Sound Subprotocols for ProtoStar

In this section, we present special-sound protocols for permutation, high-degree gate, circuit selection 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 Sect. 3.

4.1 Permutation Relation

Definition 4. *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** $\mathsf{V}(\sigma)$
	$\mathbf{w}\longrightarrow$	
		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 5. *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. $k = 1$) with verifier degree d.

4.3 Lookup Relation

Definition 6. *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 [17], and present a special-sound protocol for the lookup relation.

Lemma 4 (Lemma 5 of [17]). *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 [8], 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. $G = \mathsf{Commit}(\mathsf{ck}, \pi.\mathbf{g})$, $G' = \mathsf{Commit}(\mathsf{ck}, \mathsf{acc}.\mathbf{g})$,
2. $M = \mathsf{Commit}(\mathsf{ck}, \pi.\mathbf{m})$, $M' = \mathsf{Commit}(\mathsf{ck}, \mathsf{acc}.\mathbf{m})$,
3. $GT = \mathsf{Commit}(\mathsf{ck}, \pi.\mathbf{g} \circ \mathbf{t})$, $GT' = \mathsf{Commit}(\mathsf{ck}, \mathsf{acc}.\mathbf{g} \circ \mathbf{t})$.

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 $O(\ell)$ 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 $\mathbf{e}_i$ $(1 \leq i \leq d-1)$ 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.

Finally, we note that the algorithm above can be generalized to support polynomial $\mathbf{e}(X)$ with more general formats and with higher degrees. We refer to the full version [7] for more details.

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.*

We defer the proof to the full version [7].

The Special-Sound Protocol for Plonkup. The special-sound protocol for the Plonkup relation is the parallel composition of Π_{σ}, Π_{GATE} and Π_{LK}. We refer to the full version [7] for more detailed descriptions.

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 Sect. 4.3 to support vector lookups.

Definition 7. *Consider configuration $\mathcal{C}_{VLK} := (T, \ell, v \in \mathbb{N}, \mathbf{t})$ 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 Sect. 3.4 of [17], we can extend Lemma 4 and replace Eq. 5 with

$$\sum_{i=1}^{\ell} \frac{1}{X + w_i(Y)} = \sum_{i=1}^{T} \frac{\mathbf{m}_i}{X + t_i(Y)} \tag{6}$$

where the polynomials are defined as

$$w_i(Y) := \sum_{j=1}^{v} \mathbf{w}_{i,j} \cdot Y^{j-1}, \quad t_i(Y) := \sum_{j=1}^{v} \mathbf{t}_{i,j} \cdot Y^{j-1},$$

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		
$\mathbf{m}_i := \sum_{j=1}^{\ell} \mathbb{1}(\mathbf{w}_j = \mathbf{t}_i) \forall i \in [T]$	$\xrightarrow{\mathbf{w}, \mathbf{m}}$	
	$\xleftarrow{\beta}$	$\beta \leftarrow\$ \mathbb{F}$
	$\xrightarrow{\perp}$	
	$\xleftarrow{r}$	$r \leftarrow\$ \mathbb{F}$
Compute $[\beta_i = \beta^{i-1}]_{i=1}^v$ and $\mathbf{h} \in \mathbb{F}^\ell$, $\mathbf{g} \in \mathbb{F}^T$		
$\mathbf{h}_i := \frac{1}{w_i(\beta) + r} \forall i \in [\ell]$		
$\mathbf{g}_i := \frac{\mathbf{m}_i}{t_i(\beta) + r} \forall i \in [T]$	$\xrightarrow{[\beta_i]_{i=1}^v, \mathbf{h}, \mathbf{g}}$	
		$\sum_{i=1}^{\ell} \mathbf{h}_i \stackrel{?}{=} \sum_{i=1}^{T} \mathbf{g}_i$
		$\mathbf{h}_i \cdot [(\sum_{j=1}^{v} \mathbf{w}_{i,j} \cdot \beta_j) + r] \stackrel{?}{=} 1 \forall i \in [\ell]$
		$\mathbf{g}_i \cdot [(\sum_{j=1}^{v} \mathbf{t}_{i,j} \cdot \beta_j) + r] \stackrel{?}{=} \mathbf{m}_i \forall i \in [T]$
		$\beta_{i+1} \stackrel{?}{=} \beta_i \cdot \beta \forall i \in [v-1], \beta_1 \stackrel{?}{=} 1$

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

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

and

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

where $w_i(\beta_1, \ldots, \beta_v) := \left(\sum_{j=1}^{v} \mathbf{w}_{i,j} \cdot \beta_j\right)$ and $t_i(\beta_1, \ldots, \beta_v) := \left(\sum_{j=1}^{v} \mathbf{t}_{i,j} \cdot \beta_j\right)$. The degree of the verifier is 5. In practice, the negligible completeness error can likely be ignored without implementing these checks.

Accumulation 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 3; the number of non-zero elements in the prover message is at most $(v + 3)\ell + v$. To ensure that the accumulation procedure only requires $O(v\ell)$ operations independent of T, we can apply the same trick as in Sect. 4.3.

Special-Soundness. The perfect complete version of Π^v_{VLK} is special-sound.

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

We defer the proof to the full version [7].

4.4 Circuit Selection

We provide a sub-protocol for showing that a vector has a single one-bit (and zeros otherwise) at the location of a program counter *pc*. This is later used to select the appropriate circuit.

Definition 8. *For an integer n the relation $\mathcal{R}_{select}$ is the set of tuples $(\mathbf{b}, pc) \in \mathbb{F}^n \times \mathbb{F}$ such that $b_i = 0 \forall i \in [n] \setminus \{pc\}$ and if $pc \in [n]$ then $b_{pc} = 1$.*

Special-sound protocol Π_{select} for circuit selecting relation $\mathcal{R}_{\text{select}}$

Prover $\mathsf{P}(\mathbf{b} \in \mathbb{F}^n, pc \in \mathbb{F})$		**Verifier** V
	$\mathbf{b}, pc \longrightarrow$	
		$b_i \cdot (pc - i) \stackrel{?}{=} 0 \forall i \in [n]$
		$b_i \cdot (b_i - 1) \stackrel{?}{=} 0 \forall i \in [n]$
		$\sum_{i \in [n]} b_i \stackrel{?}{=} 1$

Complexity and security. Π_{select} is a 1-move protocol (i.e. $k = 1$); the degree of the verifier is 2.

The protocol trivially satisfies completeness. Note that the protocol is also sound: the checks $b_i \cdot (b_i - 1) = 0$ ensure that the vector $\mathbf{b}$ is Boolean; the checks $b_i \cdot (pc - i) = 0$ ensures that $b_i = 0$ if $i \neq pc$; finally, the last check guarantees that $b_{pc} = 1 - \sum_{i \in [n] \setminus \{pc\}} b_i = 1$ as $b_i = 0$ for all $i \in [n] \setminus \{pc\}$.

5 Protostar

In this section, we describe Protostar, which is built using 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 ease of exposition, we assume that the I circuits have the same number of (i) gates n, (i) gate arity c, (ii) gate degree d, (iii) gate types m, (iv) public inputs ℓ_{in} and (v) lookup gates ℓ_{lk}.

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

Definition 9. *Consider configuration* $\mathcal{C}_{mplkup} := \big(\mathsf{pp} = [n, T, c, d, m, \ell_{\mathsf{in}}, \ell_{\mathsf{lk}}]; [\mathcal{C}_i]_{i=1}^{I}; \mathbf{t}\big)$ *where the* i*th* $(1 \le i \le I)$ *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* $(\mathsf{pi}, \mathbf{w} \in \mathbb{F}^{cn})$ *is in the relation* $\mathcal{R}_{mplkup}$ *if and only if* $(\mathsf{pi}, \mathbf{w}) \in \mathcal{R}_{plonkup}$ *w.r.t. circuit configuration* $(\mathcal{C}_{pc}, \mathbf{t})$.

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 sends V vector $\mathbf{b} = (0, \ldots, 0, b_{pc} = 1, 0, \ldots, 0) \in \mathbb{F}^I$.
2. V checks that $b_i \cdot (1 - b_i) \stackrel{?}{=} 0$ and $b_i \cdot (i - pc) \stackrel{?}{=} 0$ for all $i \in [I]$, and $\sum_{i \in [I]} b_i \stackrel{?}{=} 1$.
3. P sends vector $\mathbf{m} \in \mathbb{F}^T$ such that $\mathbf{m}_i := \sum_{j \in L_{pc}} \mathbb{1}(\mathbf{w}_j = \mathbf{t}_i) \forall i \in [T]$.
4. P sends V a sparse vector $\mathbf{w}^* := (\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(I)}) \in \mathbb{F}^{Icn}$ where $\mathbf{w}^{(i)} = 0^{cn}$ for all $i \in [I] \setminus \{pc\}$ and $\mathbf{w}^{(pc)} = \mathbf{w}$.
5. V checks that

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

   Public input check: $\sum_{j=1}^{I} b_j \cdot \mathbf{w}^{(j)}[1..\ell_{\mathsf{in}}] \stackrel{?}{=} \mathsf{pi}$.

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

$$\sum_{j=1}^{I} b_j \cdot \mathsf{GT}_{j,i}\left(\mathbf{w}_i^{(j)}, \ldots, \mathbf{w}_{i+cn-n}^{(j)}\right) = 0$$

   where $\mathsf{GT}_{j,i}(x_1, \ldots, x_c) := \sum_{k=1}^{m} \mathbf{s}_{j,k}[i] \cdot G_{j,k}(x_1, \ldots, x_c)$.
6. V samples and sends P random challenge $r \leftarrow\!\!\$\, \mathbb{F}$.
7. P computes vectors $\mathbf{h} \in \mathbb{F}^{\ell_{\mathsf{lk}}}$, $\mathbf{g} \in \mathbb{F}^T$ such that

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

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

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

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

We present the special-sound protocol Π_{mplkup} for the multi-circuit Plonkup relation.

Remark 2. The public input check $\sum_{j=1}^{I} b_j \cdot \mathbf{w}^{(j)}[1..\ell_{\mathsf{in}}] \stackrel{?}{=} \mathsf{pi}$ is equivalent to $\mathbf{w}[1..\ell_{\mathsf{in}}] = \mathbf{w}_{pc}[1..\ell_{\mathsf{in}}] \stackrel{?}{=} \mathsf{pi}$ if the vector $\mathbf{b}$ passes the check at Step 2. Thus we guarantee that $\mathbf{w}[1] = pc$, and the circuit relation can add constraints on pc depending on the applications.

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

Lemma 7. *Π_{mplkup} is $2(T + \ell_{\mathsf{lk}})$-special-sound.*

We defer the proof to the full version [7].

We will now use Π_{mplkup} and our compiler described in Theorem 3 to design Protostar. Before that, we address two efficiency issues regarding supporting multiple branch circuits and combining high-degree gates with sparse lookups.

Efficient Accumulation for Supporting Many Branch Circuits. Let I be the number of branch circuits. At first glance, the message $\mathbf{w}^*$ has length $O(In)$ and seems the accumulation prover needs to take $O(In)$ time to fold the witness. Fortunately, the prover message $\mathbf{w}^* := (\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(I)}) \in \mathbb{F}^{Icn}$ is sparse: only the witness $\mathbf{w}^{(pc)}$ for the single activated branch circuit $\mathcal{C}_{pc}$ is non-zero (where $\mathbf{w}^{(pc)}$ can be determined at runtime). Thus, using the commitment to $\mathsf{acc}.\mathbf{w}^*$ and the commitments homomorphism, the complexity for the prover to fold $\mathbf{w}^*$ onto $\mathsf{acc}.\mathbf{w}^*$ is only $O(n)$.

On the other hand, the accumulation prover also needs to compute the error terms $[\mathbf{e}_j]_{j=1}^{d-1}$ described at Step 2 of Fig. 3. Note that each gate check can be split into I parts where at most one part is active, that is, $\sum_{j=1}^{I} b_j \cdot \mathsf{GT}_{j,i}(\mathbf{w}_i^{(j)}, \ldots, \mathbf{w}_{i+cn-n}^{(j)})$ can be split into I branch gates where the j-th ($1 \leq j \leq I$) branch gate is $b_j \cdot \mathsf{GT}_{j,i}(\mathbf{w}_i^{(j)}, \ldots, \mathbf{w}_{i+cn-n}^{(j)})$. Thus we can use the caching algorithm described in Sect. 3.4 to achieve $O(d|\mathcal{C}_{pc}|)$ computational complexity rather than $O(d(|\mathcal{C}_1| + \cdots + |\mathcal{C}_I|))$ where $\mathcal{C}_i$ ($1 \leq i \leq I$) is the evaluation cost of the i-th branch circuit.

Next, we address the issue of combining the high-degree gate and sparse lookup protocols with the generic transform CV in Sect. 3.3.

Efficient Accumulation of $\mathsf{CV}[\Pi_{mplkup}]$. $\mathsf{CV}[\Pi_{\text{GATE}}]$ reduces the number of degree-d verification checks in Π_{GATE} from n to 1, with the tradeoff of $O(\sqrt{n})$ additional degree-2 checks. In the resulting accumulation scheme, the error terms for high-degree gates are, thus, only of length 1. This enables using the trivial identity commitment for these error terms and thus reduces the number of group operations by the accumulation verifier. Unfortunately, applying CV 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) CV[mplkup] and b) is not composable with the sparseness optimizations for lookup described in Sect. 4.3. These optimizations make the prover computation independent of T.

Fortunately, 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 $\mathsf{CV}[\Pi_{\text{mplkup}}]$ as applying the generic transform CV only to the Π_{GATE} part of Π_{mplkup}. This means that there are $d + 1$ cross error vectors (each of length 1) for the degree $d + 2$ check in $\mathsf{CV}[\Pi_{\text{GATE}}]$; and 1 cross error vector of length $T + \ell_{\mathsf{lk}} + cn + O(\sqrt{n})$ for the rest checks—namely the low-degree checks in Π_{mplkup} and the $O(\sqrt{n})$ degree-2 checks in $\mathsf{CV}[\Pi_{\text{GATE}}]$. By leveraging the error separation technique described in Sect. 3.3, we can use the identity function to commit to the field elements and a vector commitment to commit to the long

error term. Again we leverage homomorphism as described in Sect. 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 $\mathsf{cm} = (\mathsf{Setup}, \mathsf{Commit})$*, and under the Fiat-Shamir Heuristic (Definition 3) for a hash function* H*, there exists an IVC scheme* Protostar *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* $\ell_{\mathsf{in}} = 1$*: (we omit cost terms that are negligible compared to the dominant parts)*

$\mathsf{P}_{\text{Protostar}}$ *native*	$\mathsf{P}_{\text{Protostar}}$ *recursive*	$\mathsf{V}_{\text{Protostar}}$	$\lvert\pi_{\text{Protostar}}\rvert$
$O(\lvert\mathbf{w}\rvert + \ell_{\mathsf{lk}})\mathbb{G}$ $L'(\mathcal{C}_{pc}, d+2) + 2\ell_{\mathsf{lk}}\mathbb{F}$	$3\mathbb{G}$ $d + 4\mathbb{F}$ $d + O(1)H + 1\mathsf{H}_{\mathsf{in}}$	$O(c \cdot n + T + \ell_{\mathsf{lk}})\mathbb{G}$ $n + \sum_{i=1}^{I} \mathcal{C}_i + T + \ell_{\mathsf{lk}}\mathbb{F}$	$O(c \cdot n + T + \ell_{\mathsf{lk}})$

Here $\lvert\mathbf{w}\rvert \leq cn$ *is the number of non-zero entries in the witness,* $\sum_{i=1}^{I} \mathcal{C}_i$ *is the cost of evaluating all circuits on some random input, and* $L'(\mathcal{C}_{pc}, d)$ *is the cost of computing the coefficients of the polynomial* $e(X)$ *defined in Eq. 3 using techniques from Sect. 5.*[5] H_{in} *is the cost of hashing the public input and the (constant-sized) accumulator instance.*

We defer the proof to the full version [7].

References

1. Attema, T., Fehr, S., Klooß, M.: Fiat-Shamir transformation of multi-round interactive proofs. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part I. LNCS, vol. 13747, pp. 113–142. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22318-1_5
2. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Scalable zero knowledge via cycles of elliptic curves. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 276–294. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_16
3. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKS and proof-carrying data. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 111–120. ACM Press (2013). https://doi.org/10.1145/2488608.2488623
4. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 757–788. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_25

[5] As noted in Theorem 3, $L'(\mathcal{C}_{pc}, d+2)$ is bounded by $O(nd\log^2(d))$.

5. Bonneau, J., Meckler, I., Rao, V., Shapiro, E.: Coda: decentralized cryptocurrency at scale. Cryptology ePrint Archive, Report 2020/352 (2020). https://eprint.iacr.org/2020/352
6. Bowe, S., Grigg, J., Hopwood, D.: Halo: recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021 (2019). https://eprint.iacr.org/2019/1021
7. Bünz, B., Chen, B.: Protostar: generic efficient accumulation/folding for special sound protocols. In: Cryptology ePrint Archive (2023)
8. Bünz, B., Chiesa, A., Lin, W., Mishra, P., Spooner, N.: Proof-carrying data without succinct arguments. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 681–710. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84242-0_24
9. Bünz, B., Chiesa, A., Mishra, P., Spooner, N.: Recursive proof composition from accumulation schemes. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 1–18. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64378-2_1
10. Buterin, V.: The different types of ZK EVM (2022). https://vitalik.ca/general/2022/08/04/zkevm.html. Accessed 27 Apr 2023
11. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: HyperPlonk: plonk with linear-time prover and high-degree custom gates. Cryptology ePrint Archive, Report 2022/1355 (2022). https://eprint.iacr.org/2022/1355
12. Chiesa, A., Tromer, E.: Proof-carrying data and hearsay arguments from signature cards. In: Chi-Chih, A. (ed.) ICS 2010, pp. 310–331. Yao, Tsinghua University Press (2010)
13. Chiesa, A., Tromer, E., Virza, M.: Cluster computing in zero knowledge. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 371–403. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_13
14. Eagen, L., Fiore, D., Gabizon, A.: cq: cached quotients for fast lookups. Cryptology ePrint Archive, Report 2022/1763 (2022). https://eprint.iacr.org/2022/1763
15. Gabizon, A., Williamson, Z.J.: plookup: a simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Report 2020/315 (2020). https://eprint.iacr.org/2020/315
16. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019). https://eprint.iacr.org/2019/953
17. Haböck, U.: Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive, Report 2022/1530 (2022). https://eprint.iacr.org/2022/1530
18. Kattis, A., Bonneau, J.: Proof of necessary work: succinct state verification with fairness guarantees. Cryptology ePrint Archive, Report 2020/190 (2020). https://eprint.iacr.org/2020/190
19. Khovratovich, D., Maller, M., Tiwari, P.R.: MinRoot: candidate sequential function for ethereum VDF. Cryptology ePrint Archive, Report 2022/1626 (2022). https://eprint.iacr.org/2022/1626
20. Kothapalli, A., Setty, S.: HyperNova: recursive arguments for customizable constraint systems. In: Cryptology ePrint Archive (2023)
21. Kothapalli, A., Setty, S.: SuperNova: proving universal machine executions without universal circuits. Cryptology ePrint Archive, Report 2022/1758 (2022). https://eprint.iacr.org/2022/1758

22. Kothapalli, A., Setty, S., Tzialla, I.: Nova: recursive zero-knowledge arguments from folding schemes. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, vol. 13510, pp. 359–388. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15985-5_13
23. Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. In: 31st FOCS, pp. 2–10. IEEE Computer Society Press (1990). https://doi.org/10.1109/FSCS.1990.89518
24. Mohnblatt, N.: Sangria: a folding scheme for PLONK (2023). https://github.com/geometryresearch/technical_notes/blob/main/sangria_folding_plonk.pdf. Accessed 27 Apr 2023
25. Naveh, A., Tromer, E.: PhotoProof: cryptographic image authentication for any set of permissible transformations. In: 2016 IEEE Symposium on Security and Privacy, pp. 255–271. IEEE Computer Society Press (2016). https://doi.org/10.1109/SP.2016.23
26. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9
27. Posen, J., Kattis, A.A.: Caulk+: table-independent lookup arguments. Cryptology ePrint Archive, Report 2022/957 (2022). https://eprint.iacr.org/2022/957
28. Setty, S., Angel, S., Gupta, T., Lee, J.: Proving the correct execution of concurrent services in zero-knowledge. In: 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2018), pp. 339–356 (2018)
29. Setty, S., Thaler, J., Wahby, R.: Customizable constraint systems for succinct arguments. Cryptology ePrint Archive (2023)
30. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 1–18. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_1
31. Wikström, D.: Special soundness in the random oracle model. Cryptology ePrint Archive, Report 2021/1265 (2021). https://eprint.iacr.org/2021/1265
32. Xiong, A.L., et al.: VERI-ZEXE: decentralized private computation with universal setup. Cryptology ePrint Archive, Report 2022/802 (2022). https://eprint.iacr.org/2022/802
33. Zapico, A., Buterin, V., Khovratovich, D., Maller, M., Nitulescu, A., Simkin, M.: Caulk: lookup arguments in sublinear time. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022, pp. 3121–3134. ACM Press (2022). https://doi.org/10.1145/3548606.3560646
34. Zapico, A., Gabizon, A., Khovratovich, D., Maller, M., Ràfols, C.: Baloo: nearly optimal lookup arguments. Cryptology ePrint Archive, Report 2022/1565 (2022). https://eprint.iacr.org/2022/1565
35. Zhang, Y.X., Vark, A.: Origami - a folding scheme for Halo2 lookups (2023). https://hackmd.io/@aardvark/rkHqa3NZ2. Accessed 12 July 2023