ECLIPSE: Enhanced Compiling method for Pedersen-committed zkSNARK Engines*

Diego F. Aranha[1], Emil Madsen Bennedsen[2], Matteo Campanelli[1], Chaya Ganesh[3], Claudio Orlandi[1], and Akira Takahashi[1]

[1] Aarhus University, Aarhus, Denmark
{dfaranha, matteo, orlandi, takahashi}@cs.au.dk
[2] Concordium, Denmark
masik7@gmail.com
[3] Indian Institute of Science, India
chaya@iisc.ac.in

July 8, 2021

Abstract. We provide new constructions for zero-knowledge commit-and-prove SNARKs (CP-SNARKs) with a universal updatable SRS. Informally, a commit-and-prove argument system is one that can efficiently prove relations over committed inputs. They have many applications, including allowing for efficient composition of proof systems with different strength points.
We first show a general technique to compile Algebraic Holographic Proofs (AHP) with special "decomposition" properties into an efficient CP-SNARK with universal and updatable SRS. We require that the polynomials in an AHP can be easily decomposed into components that refer to the committed part of the witness and the rest of the witness respectively.
We then show that some of the most efficient AHP constructions—Marlin, PLONK, and Sonic—satisfy our compilation requirements. To obtain succinct instantiations of our protocols we rely on recent advancements in compressed Σ-protocol theory (Attema and Cramer, Crypto '20). Our constructions retain the succinct proof size of the underlying AHP and only impose an additional proof size that grows logarithmically with the size of the committed component of the witness.

* Research supported by: the Concordium Blockchain Research Center, Aarhus University, Denmark; the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC);

Table of Contents

1 Introduction

Zero-knowledge proof systems [GMR85] have a rich history in cryptography and theory of computation [GMW86, For87, BGG+90] supporting numerous cryptographic constructions. Examples of early applications of zero-knowledge proofs include identification schemes [FFS87], CCA-secure public-key encryption [NY90], signature schemes [CS97], anonymous credentials [CL01], secure multi-party computation [GMW87], and many others. More recently, cryptocurrencies such as Zcash [BCG+14] use zero-knowledge proofs to provide integrity while maintaining privacy. In applications like cryptocurrencies and anonymous credentials, practical deployment demands small proof sizes and fast verification.

Succinct Zero-knowlege proofs. There has been a series of works on constructing zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs). Many recent applications of zero-knowledge (such as cryptocurrencies [BCG+14]), demand small proof sizes and fast verification, as provided by zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs). The seminal paper of [GGPR13] proposed a pairing-based zk-SNARK for general NP statements based on the NP complete langauge of Quadratic Span Programs (QSP) for Boolean circuits and Quadratic Arithmetic Programs (QAP) for arithmetic circuits. This built on previous works of [IKO07, Gro10, Lip12] and led to several works [BCI+13, PHGR13, BCG+13, Lip13, BCTV14, Gro16] which have proofs that are very short and have fast verification time. This line of work on (pre-processing) zkSNARKs has seen rapid progress with many works proposing significant improvements in efficiency of different parameters of interest like proof size, verifier efficiency, complexity of setup, prover efficiency, among others.

Preprocessing zk-SNARKs, which are the most attractive proof system when applications demand verification efficiency, typically rely on a trusted setup to produce a structured reference string (SRS). If the trusted setup is compromised, it becomes possible to break the soundness property of the proof system. One solution is to use a secure multi-party computation protocol (MPC) to conduct the setup [BGM17], and as long as at least one party is honest, the setup remains secure. However, pre-processing SNARKs have setup that depends on the relation, and it becomes infeasible to perform an MPC for every new relation an application needs to prove. Recently, in [GKM+18] a zk-SNARK with an updatable SRS that allows participants to dynamically update the SRS was proposed. Even though this does not avoid the problem of trusted setup, the security now depends on one contributor deleting the contributed randomness. Moreover, the scheme only requires a universal SRS that allows to prove statements about all circuits of some bounded size. However, the size of this universal updatable SRS is quadratic in the number of multiplication gates of the circuit representing the statement. In [MBKM19], the authors construct Sonic, a zkSNARK that is universal and updatable with a linear-sized SRS. Many recent constructions of updatable SRS zkSNARKs [BFS20, CHM+20, GWC19] follow a modular approach where an information-theoretic protocol is constructed in an abstract model like Probabilistically Checkable Proof (PCP), linear PCP, Interactive Oracle Proof (IOP) etc., and then the information-theoretic protocol is compiled via a cryptographic compiler to obtain an argument system. Marlin [CHM+20] uses an IOP abstraction called algebraic holographic proofs (AHP), and the DARK compiler [BFS20] uses an abstraction called polynomial IOPs (PIOPs). In these abstractions, the prover provides oracle access to a set of polynomials, and the verifier sends random challenges. Then, the verifier asks for evaluations of these polynomials and decides to accept or not based on the answers. PLONK [GWC19] uses an abstraction called idealized low degree protocols (ILDPs) that proceeds in a similar way except that at the end of the protocol, the verifier checks a set of polynomial identities over the oracles sent by the prover.

Σ-protocols. Σ-protocols are proof systems that are efficient for proving algebraic statements about discrete logarithms, roots, or polynomial relationships among values [Sch90, GQ88, CDS94, CS97]. Proofs based on Σ-protocols are extremely efficient for statements that come up in cryptographic constructions: statements that are efficiently representable as algebraic functions over some group $\mathbb{G}$ where the discrete-logarithm problem is hard. For example, a prover may want to convince the verifier that she knows an x such that $g^x = y$ for publicly known values $g, y \in \mathbb{G}$. They yield short proof sizes, require a constant number of public-key operations, and do not impose trusted setup requirements. Moreover, they can be made non-interactive using the efficient Fiat-Shamir transformation [FS87].

While Σ-protocols are efficient in practice, they only apply to a restricted set of languages. Proving statements about a cryptographic hash function or a block cipher represented by a Boolean or arithmetic circuit using a straightforward Σ-protocol results in the proof size and the proving/verification time scaling linearly with the size of circuit.

Recent work on compressed Σ-protocol theory [AC20] is a strengthening of Σ-protocols that compress the communication complexity from linear to logarithmic. The underlying pivot of the compressed proto-

col is a standard Σ-protocol for opening linear forms on Pedersen vector commitments, i.e., a Σ-protocol for proving that a committed vector $\mathbf{x}$ satisfies $L(\mathbf{x}) = y$ for a public scalar y and public linear form L.

1.1 Our Contributions

Consider a composite computation that naturally presents different components, like a Boolean circuit for a hash function, and algebraic representation for group exponentiation. A general-purpose zero-knowledge proof system for such a computation requires a single homogeneous representation, thus incurring a high cost in performance. Ideally, one would like to take advantage of the nuances of a computation and choose the best proof system for each component of the computation. Motivated by applications where it is desirable to use SNARKs and Σ-protocols for proving a single composite statement, we show how to combine state-of-the-art updatable SRS SNARK with state-of-the-art compressed Σ-protocol proofs by presenting a general compiler from AHP to commit-and-prove zkSNARKs (CP-SNARKs).

- *Compiler from AHP to CP-SNARK.* We present a compiler that takes an AHP which is the information-theoretic protocol underlying many existing zkSNARKs and compiles it into a CP-SNARK. Our compiler is similar in spirit to compilers of [BFS20, CHM+20, CFF+20] that convert information-theoretic protocols to succinct arguments. We first abstract out a set of properties that AHP should satisfy to apply our "split–and–link" paradigm, leading to efficient CP-SNARKs (outlined below in Sect. 1.3).
- *Concrete instantiations.* We then apply our compiler to the AHPs of Marlin, PLONK and Sonic to obtain concrete CP-SNARKs.[4] This immediately allows us to prove that the inputs (and/or outputs) used in the zk-SNARK for an arithmetic circuit/Rank 1 constraint system statement are the same as the values inside an algebraic commitment. This helps to hide intermediate outputs of a composite statement by committing to it, thus allowing switching between the algebraic (Σ-protocols) and arithmetic world (zk-SNARK). One instance of an application is proving knowledge of input x such that $\mathsf{Com}(x) = y$ and $\mathsf{Com}(H(x)) = z$ for public values y and z where H is a hash function and Com is the Pedersen commitment scheme. One can use standard Σ-protocols to prove statements about committed values y, z. Thus, using the commitment as an anchor, this immediately yields a NIZK for a composite statement that combines SNARK with updatable SRS with compressed Σ-protocols. In order to make the argument for the composite statement succinct, we use recent advances in compressed Σ-protocol theory. We cast the statement about consistency with Pedersen commitments as statements about knowledge of pre-image of group homomorphisms. This allows us to apply the compression technique of [AC20] that achieves logarithmic communication for the canonical Σ-protocol and the amortization technique that proves many statements efficiently. Thus, our linking protocol that needs to prove ℓ statements, where each statement is about equality of vectors of size d, achieves communication complexity $O(\log(\ell d))$, so the overall proof (the size of the SNARK together with the size of the linking proof) is still succinct.

Our constructions improve on the efficiency of existing CP-SNARKs (see Table 1). In particular they improve on the proof size of Lunar for small values of d and large values of ℓ. This scenario occurs, for example, in the delegated credentials setting that we outline below in Section 1.2. Moreover, compared to [AGM18] which tackled a similar problem relying on naïve Σ-protocols and a specific QAP-based SNARK construction with non-updatable SRS, our approach achieves better asymptotic efficiency as well as further generality.

1.2 Applications

Anonymous and Delegated Credentials. Consider the application of making digital certificates anonymous: one would like to prove knowledge of a message m and a signature σ, where σ is a valid signature on message m with respect to some public verification key. While there is a large body of work on anonymous credentials, very few of the techniques can turn the commonly used X.509 certificates into anonymous credentials. The main challenge is that the statement being considered is a composite statement containing both Boolean (hash function) and algebraic (group operations) components, since the message is hashed before being signed. Efficient NIZK for composite statements that use a zk-SNARK for the circuit part and Σ-protocols for the algebraic would yield a more efficient proof system.

[4] The reason why we apply our compiler to all three proof systems is that Marlin, PLONK and Sonic are a sort of rock-paper-scissor for AHPs. Since they use different models of computations it might be possible to prove some statements more efficient with one rather than the others. We believe that our techniques are general enough to extend to future AHPs.

	$\|\pi\|$	Prove (time)	Verify (time)
This work	$O(\log(\ell \cdot d))$	$O(n + \ell \cdot d)$	$O(\ell \cdot d)$
Lunar [CFF+20]	$O(\ell)$	$O(n + \ell \cdot d)$	$O(\ell)$
LegoUAC [CFQ19]	$O\left(\ell \log^2(n)\right)$	$O(n) + \ell \cdot \tilde{O}(d)$	$O\left(\ell \log^2(n)\right)$

Table 1. Efficiency comparison among CP-SNARK constructions with universal and updatable SRS. Proving time expresses group operations. The first line refers to our compiler applied to AHPs with suitable decomposition properties (See Section 3). In the above we denote by n the number of constraints in an R1CS system, by ℓ the number of input commitments and by d the size of each committed vectors. (The same asymptotics apply also to other constraints systems with slight variations though. For example, they apply to the AHPs in PLONK if n above refers to the total number of gates).

Our amortization techniques can be useful in a setting of "delegated credentials". Each citizen or member of an organization can have associated a bundle of properties (credentials), e.g., credit and employment history or digital certificates issued by governments. We assume these properties are of size d and are fingerprinted through a (compressing) commitment and that each of these users delegates the storage of these properties to a service. Every time the user needs to prove a statement on these credentials with respect to the public commitment, it can issue an order to the service. Instead of providing a single proof per user, a service can wait to accumulate ℓ orders and provide a single proof for all of them. Our construction allows to prove each of the ℓ orders for credentials of size d with communication complexity only $O(\log(\ell d))$.

Blockchains. Credential systems are used for balancing privacy and accountability on the blockchain [DGK+21], and the flexibility of choosing a suitable proof system for each statement component allows for flexibility of using suitable cryptographic primitives (like signature schemes) in the design without compromising on efficiency of the proof system. Many cryptocurrencies also implement confidential transactions [Max15] where the amounts are hidden inside commitments. To enable verification of such a transaction, they include a zero-knowledge proof that the sum of values in committed inputs is greater than the sum of committed outputs and that all values are positive. Most implementations of confidential transactions use range proofs over committed values. A state-of-the-art zkSNARK on committed values provides an efficient candidate for such applications.

Stitching proofs. Consider having to prove a large statement represented by a circuit whose size exceeds the upper bound of the universal SRS size. We can view the large circuit as containing many different smaller subcircuits, prove each sub-circuit, and stitch proofs using committed input and output at each layer (by proving equality of input of a sub-circuit and the output of the previous sub-circuit). For instance, at a high level, one of the statements in ZCash is of the form: knowledge of x_i's such that $H(x_1||H(x_2||\cdots H(x_k))) = y$. The value of k is large and therefore the SRS generated for proving this statement is large as well. If a universal updatable SRS does not support this k, a better alternative to generating (and updating) a new SRS is instead to use the SRS to prove sub-circuits and compose them. Each sub-circuit proves knowledge of x, y such that $H(x||H(y)) = z$, and many such proofs are composed by proving equality of the output and input of successive sub-circuits using the commitments. This can be extended to a general system using SRSs for small size circuits $C_1, \ldots, C_k$ allowing NIZKs for arbitrary composition of these circuits for proof of a larger statement without having to generate a new larger SRS.

1.3 Technical Overview

We follow the blueprint of the modular approach for designing efficient arguments that consists of an information theoretic protocol in an abstract model (PCP, linear PCP, IOP etc.), and then compiling the information-theoretic protocol via a cryptographic compiler to obtain an argument system. Many recent constructions of zkSNARKs [BFS20, CHM+20, GWC19] follow this approach where the information theoretic object is an algebraic variant of IOP, and the cryptographic primitive in the compiler is a polynomial commitment scheme. This is then transformed into a non-interactive argument by applying Fiat-Shamir. We consider KZG-like polynomial commitment schemes [KZG10] that essentially consists of an "evaluation of the polynomial at a secret point given in the exponent" as the commitment.

The high-level idea of our compiler to obtain a CP-SNARK is as follows: We look at the polynomial oracles sent by the AHP prover that "encode" the witness; we call these *witness-carrying polynomials*.

Our transformation requires the AHP to have *decomposable* witness-carrying polynomials; at a high level, this allows the prover to commit to different parts of the witness independently. This property is already satisfied by AHPs underlying Marlin, Sonic and PLONK. The prover in the compiled argument will "split" the witness into a committed part and non-committed part, and encode them separately. The argument prover commits to the polynomial oracles sent by the AHP prover and sends this commitment to the argument verifier. We now exploit the homomorphic property of the polynomial commitment scheme so that the verifier can locally put together an encoding of the combined witness polynomial given the encodings of the split witness. We then use a *linking protocol* to prove consistency of the split witness in this separate commitment of the AHP with values inside external commitments. When the polynomial commitment scheme is instantiated with KZG-like commitment, the linking protocol is essentially an efficient Σ-protocol that shows equality of exponents in Pedersen commitments with the values of the split witness encoded in the polynomial commitment.

Finally, to instantiate our compiler with the AHP underlying PLONK, Marlin and Sonic to obtain concrete CP-SNARKs, we design suitable linking protocols for the specific witness encoding used in that particular AHP. We need additional techniques to ensure that the prover cannot exploit the split witness to break soundness. Marlin and PLONK use the Lagrange interpolation basis to encode the witness vector, and our linking protocol ensures that only the allowed Lagrange basis are used in each of the split witness polynomials; that is, there is no "overlap" of the witness vector across the split. Sonic uses a constraint system with respect to bivariate polynomial equations similar to the one used in [BCC+16, BBB+18] to capture satisfiability of a circuit, where the values on the circuit wires are encoded as the coefficients of a polynomial. Our linking protocol for Sonic ensures that the coefficients are only used in the allowed degree bounds; that is, there is no "overlap" of the coefficients across the split polynomials.

We believe that this modularity of split witness encoding in the AHP combined with a linking protocol will lead to CP-SNARK constructions from other SNARKs that rely on AHPs and KZG-type or other suitable homomorphic polynomial commitment schemes.

1.4 Related Work

There are few examples in the literature of efficient zero-knowledge proof systems for composite statements like those we consider in this paper.

The first paper in this important line of work [CGM16] presents a zero-knowledge proof that can be used to prove that $F(x) = 1$ given a Pedersen commitment to x, where F is represented as a Boolean circuit. They provide an efficient way of combining the garbled-circuit based proof of [JKO13] for circuit-based statements with Σ-protocols for algebraic parts. However, this is inherently interactive which is inherited from the interactivity of [JKO13] where the verifier uses private coins.

In [BHH+19], the authors show how to extend the MPC-in-the-head techniques of ZKBoo [GMO16] and ZKB++ [CDG+17] to allow algebraic statements on Pedersen commitments. While allowing for non-interactive proofs via the Fiat-Shamir transform, this approach results in larger proof sizes. In [AGM18], protocols combining zk-SNARKs with Σ-protocols are presented. This overcomes the disadvantage of interactivity, and also gives a system suitable for applications that require short proofs. The techniques used however, work for zk-SNARKs that are based on Quadratic Arithmetic Programs (QAP). While this encompasses efficient systems like [GGPR13, PHGR13, Gro16], they all fall into the class of SNARKs that require a trusted setup.

Bulletproofs [BBB+18] can be used to prove statements on algebraically committed inputs, and be made non-interactive using Fiat-Shamir. Unfortunately, proof sizes scale logarithmically and the verification time scales linearly with the size of the circuit.

LegoSNARK [CFQ19] is a framework for commit-and-prove zkSNARKs (CP-SNARKs) that gives general composition tools to build new CP-SNARKs from proof gadgets in a modular way. The commit-and-prove can be leveraged by using a variety of existing schemes and making them interoperable. The construction LegoUAC in [CFQ19] is a CP-SNARK with a universal and updatable SRS, but it produces proofs of size $\log^2(N)$ for a size bound N on the relation.

Lunar [CFF+20] obtains CP-SNARKs with a universal and updatable SRS with proofs of constant size. It also presents proof systems for "linking" committed inputs to the polynomial commitments used in AHP-based arguments. Our work and that in Lunar achieve better efficiency in different settings. Consider a circuit with ℓ committed inputs where each input is a vector of (maximum) size d. The proof size of constructions in Lunar grows linearly in ℓ, and independently of d, while CP-PLONK and CP-Marlin derived from our compiler incur overhead of $O(\log(\ell d))$ (see Table 1). Hence our work achieves better asymptotic proof size when d is small and $\ell > 1$. On the other hand, for larger d, the proofs in Lunar are shorter.

2 Preliminaries

Notation. For positive integers a and b such that $a < b$ we use the integer interval notation $[a, b]$ to denote $\{a, a+1, \ldots, b\}$; we use $[b]$ as shorthand for $[1, b]$. A finite field is denoted by $\mathbb{F}$. We denote by κ a security parameter. When we explicitly specify the random tape ρ for a randomized algorithm $\mathcal{A}$, then we write $a \leftarrow \mathcal{A}(\mathsf{srs}; \rho)$ to indicate that $\mathcal{A}$ outputs a given input srs and random tape ρ. For a pair of randomized algorithms $\mathcal{A}$ and $\mathcal{E}_{\mathcal{A}}$, we often use the handy notation $(a; x) \leftarrow (\mathcal{A} || \mathcal{E}_{\mathcal{A}})(\mathsf{srs})$ which denotes that $\mathcal{A}$ outputs a on input srs, and $\mathcal{E}_{\mathcal{A}}$ outputs x given the same input srs, and $\mathcal{A}$'s random tape. We denote by $\Pr\left[A \,:\, B\right]$ the conditional probability of an event A under the condition B.

2.1 Indexed relations

Definition 1 (Indexed relation [CHM+20]). *An* indexed relation $\mathcal{R}$ *is a set of triples* $(\mathsf{i}, \mathsf{x}, \mathsf{w})$ *where* i *is the index,* x *is the instance, and* w *is the witness; the corresponding* indexed language $\mathcal{L}(\mathcal{R})$ *is the set of pairs* (i, x) *for which there exists a witness* w *such that* $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$*. Given a size bound* $\mathsf{N} \in \mathbb{N}$*, we denote by* $\mathcal{R}_{\mathsf{N}}$ *the restriction of* $\mathcal{R}$ *to triples* $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$ *with* $|\mathsf{i}| \leq \mathsf{N}$*.*

2.2 Zero-knowledge Arguments of Knowledge with preprocessing

A zero-knowledge proof (or argument)[5] for $\mathcal{L}$ allows a prover P to convince a verifier V that $x \in \mathcal{L}$ for a common input x without revealing w. A proof of knowledge captures not only the truth of a statement $x \in \mathcal{L}$, but also that the prover is in "possession" of a witness w.

Definition 2 (Preprocessing Argument with Universal SRS [CHM+20]). *A Preprocessing Argument with Universal SRS is a tuple* $\mathsf{ARG} = (\mathcal{S}, \mathcal{I}, \mathcal{P}, \mathcal{V})$ *of four algorithms.* $\mathcal{S}$ *is a probabilistic polynomial-time setup algorithm that given a bound* $\mathsf{N} \in \mathbb{N}$ *samples a structured reference string* srs *supporting indices of size up to* N*. The indexer algorithm* $\mathcal{I}$ *is deterministic and, given oracle access to* srs *produces a proving index key and a verifier index key, used respectively by* $\mathcal{P}$ *and* $\mathcal{V}$*. The latter two are probabilistic polynomial-time interactive algorithms.*

Completeness *For all size bounds* $\mathsf{N} \in \mathbb{N}$ *and efficient* $\mathcal{A}$*,*

$$\Pr\left(\begin{array}{l}(\mathsf{i}, \mathsf{x}, \mathsf{w}) \notin \mathcal{R}_{\mathsf{N}} \vee \\ \langle \mathcal{P}(\mathsf{ipk}, \mathsf{x}, \mathsf{w}), \mathcal{V}(\mathsf{ivk}, \mathsf{x}) \rangle = 1\end{array} : \begin{array}{r}\mathsf{srs} \leftarrow \mathcal{S}(1^{\kappa}, \mathsf{N}) \\ (\mathsf{i}, \mathsf{x}, \mathsf{w}) \leftarrow \mathcal{A}(\mathsf{srs}) \\ (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}^{\mathsf{srs}}(\mathsf{i})\end{array}\right) = 1$$

Knowledge Soundness *For every* $\mathsf{N} \in \mathbb{N}$ *and efficient adversary* $\tilde{\mathcal{P}} = (\tilde{\mathcal{P}}_1, \tilde{\mathcal{P}}_2)$ *there exists an efficient extractor* $\mathcal{E}$ *such that*

$$\Pr\left(\begin{array}{l}(\mathsf{i}, \mathsf{x}, \mathsf{w}) \notin \mathcal{R}_{\mathsf{N}} \wedge \\ \langle \tilde{\mathcal{P}}_2(\mathsf{st}), \mathcal{V}(\mathsf{ivk}, \mathsf{x}) \rangle = 1\end{array} : \begin{array}{r}\mathsf{srs} \leftarrow \mathcal{S}(1^{\kappa}, \mathsf{N}) \\ (\mathsf{i}, \mathsf{x}, \mathsf{st}) \leftarrow \tilde{\mathcal{P}}_1(\mathsf{srs}) \\ \mathsf{w} \leftarrow \mathcal{E}(\mathsf{srs}) \\ (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}^{\mathsf{srs}}(\mathsf{i})\end{array}\right) = \mathsf{negl}(\lambda)$$

Above we assumed the extractor takes in input the same random tape as the malicious prover.

Perfect Zero-Knowledge *There exists an efficient simulator* $\mathsf{Sim} = (\mathsf{Setup}, \mathsf{Prove})$ *such that for every efficient adversary* $\tilde{\mathcal{V}} = (\tilde{\mathcal{V}}_1, \tilde{\mathcal{V}}_2)$ *it holds that*

$$\Pr\left(\begin{array}{l}(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{N}} \wedge \\ \langle \mathcal{P}(\mathsf{ipk}, \mathsf{x}, \mathsf{w}), \tilde{\mathcal{V}}_2(\mathsf{st}) \rangle = 1\end{array} : \begin{array}{r}\mathsf{srs} \leftarrow \mathcal{S}(1^{\kappa}, \mathsf{N}) \\ (\mathsf{i}, \mathsf{x}, \mathsf{w}, \mathsf{st}) \leftarrow \tilde{\mathcal{V}}_1(\mathsf{srs}) \\ (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}^{\mathsf{srs}}(\mathsf{i})\end{array}\right) =$$

$$\Pr\left(\begin{array}{l}(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{N}} \wedge \\ \langle \mathsf{Sim.Prove}(\mathsf{trap}, \mathsf{i}, \mathsf{x}), \tilde{\mathcal{V}}_2(\mathsf{st}) \rangle = 1\end{array} : \begin{array}{r}(\mathsf{srs}, \mathsf{trap}) \leftarrow \mathsf{Sim.Setup}(1^{\kappa}, \mathsf{N}) \\ (\mathsf{i}, \mathsf{x}, \mathsf{w}, \mathsf{st}) \leftarrow \tilde{\mathcal{V}}_1(\mathsf{srs})\end{array}\right)$$

Succinctness *We call the argument succinct if the communication complexity between prover and verifier is bounded by* $\mathsf{poly}(\kappa) \cdot \mathsf{polylog}(|x| + |w|)$*.*

[5] We use proof and argument as synonymous in this paper, as we are only interested in computational soundness.

We have the following two optional requirements on the arguments defined above. We say that an argument is *public-coin* if all the messages from the verifier are uniformly random strings of a bounded length. We say it is *updatable* if there exists an update algorithm that can be run by anyone at any time and to update the SRS. this algorithm guarantees security as long as at least one of the (sequential) updates have been carried out honestly.

2.3 Σ-Protocols and Pedersen Vector Commitment

Σ-protocols are interactive proof systems consisting of three rounds. In a Σ-protocol, the prover sends a message a, the verifier replies with a random bit string c, and the prover responds with z. The verifier decides to accept or reject based on the transcript (a, c, z). A Σ-protocol can be efficiently compiled into a non-interactive zero-knowledge proof of knowledge (in the random oracle model) through the Fiat-Shamir transform [FS87]. Throughout the paper, we use the Pedersen commitment scheme [Ped92] as the algebraic commitment, which gives unconditional hiding and computational binding properties based on the hardness of computing the discrete logarithm in a group $\mathbb{G}$ of prime order q. Given two random generators $G, H \in \mathbb{G}$ such that $\log_G H$ is unknown, a value $x \in \mathbb{Z}_q$ is committed to by choosing r randomly from $\mathbb{Z}_q$, and computing $G^x H^r$. We write $\mathsf{Com}_q(x; r)$ to denote a Pedersen commitment to x with randomness r in a group of order q, and omit the subscript when the group is clear. A Pedersen commitment to a vector $\mathbf{x} \in \mathbb{Z}_q^d$ is computed as $\mathsf{Com}(\mathbf{x}; r) = \mathbf{G}^{\mathbf{x}} H^r = (\prod_{i=1}^{d} G_i^{x_i}) H^r$ where $\mathbf{G} = (G_1, \ldots, G_d)$ are randomly chosen generators with unknown relative discrete logarithms.

2.4 Algebraic Holographic Proofs

Below we recall the definition of AHP from Marlin.

Definition 3 (AHP [CHM+20]). *An* Algebraic Holographic Proofs (AHP) *over a field family $\mathcal{F}$ for an indexed relation $\mathcal{R}$ is specified by a tuple*

$$\mathsf{AHP} = (\mathsf{k}, \mathsf{s}, \mathsf{d}, \mathsf{I}, \mathsf{P}, \mathsf{V})$$

where $\mathsf{k}, \mathsf{s}, \mathsf{d} : \{0,1\}^* \to \mathbb{N}$ are polynomial-time computable functions and $\mathsf{I}, \mathsf{P}, \mathsf{V}$ are three algorithms known as the indexer, prover, *and* verifier. *The parameter k specifies the number of interaction rounds, s specifies the number of polynomials in each round, and d specifies degree bounds on these polynomials. The protocol proceeds as follows:*

- ***Offline phase*** *The indexer I receives as input a field $\mathbb{F} \in \mathcal{F}$ and index i for $\mathcal{R}$, and outputs $\mathsf{s}(0)$ polynomials $p_{0,1}, \ldots, p_{0,s(0)} \in \mathbb{F}[X]$ of degrees at most $\mathsf{d}(|\mathsf{i}|, 0, 1), \ldots, \mathsf{d}(|\mathsf{i}|, 0, \mathsf{s}(0))$ respectively. Note that the offline phase does not depend on any particular instance or witness, and merely considers the task of encoding the given index i.*
- ***Online phase*** *Given an instance x and witness w such that $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$, the prover P receives $(\mathbb{F}, \mathsf{i}, \mathsf{x}, \mathsf{w})$ and the verifier V receives $(\mathbb{F}, \mathsf{x})$ and oracle access to the polynomials output by $\mathsf{I}(\mathbb{F}, \mathsf{i})$. The prover P and the verifier V interact over $\mathsf{k} = \mathsf{k}(|\mathsf{i}|)$ rounds. For $i \in [\mathsf{k}]$, in the i-th round of interaction, the verifier V sends a message $\rho_i \in \mathbb{F}^*$ to the prover P; then the prover P replies with $\mathsf{s}(i)$ oracle polynomials $p_{i,1}, \ldots, p_{i,s(i)} \in \mathbb{F}[X]$. After k interactions, the verifer outpus additional randomness $\rho_{\mathsf{k}+1} \in \mathbb{F}^*$ which serves as auxiliary input to V in subsequent phases. We note that $\rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{k}+1} \in \mathbb{F}^*$ are public and uniformly random strings.*
- ***Query phase*** *Let $\mathbf{p} = (p_{i,j})_{i \in [\mathsf{k}], j \in [\mathsf{s}(i)]}$ be a vector consisting of all polynomials sent by the prover P. The verifier may query any of the polynomials it has received any number of times. Concretely, V executes a subroutine Q_V that receives $(\mathbb{F}, \mathsf{x}; \rho_1, \ldots, \rho_{\mathsf{k}+1})$ and outputs a query set Q consisting of tuples $((i, j), z)$ to be interpreted as "query $p_{i,j}$ at $z \in \mathbb{F}$". We denote a vector consisting of query answers $\mathbf{p}(Q)$.*
- ***Decision phase*** *The verifier outputs "accept" or "reject" based on the answers to the queries (and the verifier's randomness). Concretely, V executes a subroutine D_V that receives $(\mathbb{F}, \mathsf{x}, \mathbf{p}(Q); \rho_1, \ldots, \rho_{\mathsf{k}+1})$ as input, and outputs the decision bit.*

  The function d determines which provers to consider for the completeness and soundness properties of the proof system. In more detail, we say that a (possibly malicious) prover $\tilde{\mathsf{P}}$ is admissible *for AHP if, on every interaction with the verifier V, it holds that for every round $i \in [\mathsf{k}]$ and oracle index $j \in [s(i)]$ we have $\deg(p_{i,j}) \leq \mathsf{d}(|\mathsf{i}|, i, j)$. The honest prover P is required to be admissible under this definition.*

We require an AHP *to satisfy completeness, (knowledge) soundness and zero-knowledge as defined below.*

Completeness. An AHP is complete if for all $\mathbb{F} \in \mathcal{F}$ and any $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$, the checks returned by $\mathsf{V}^{\mathsf{I}(\mathbb{F},\mathsf{i})}(\mathbb{F}, \mathsf{x})$ after interacting with (honest) $\mathsf{P}(\mathbb{F}, \mathsf{i}, \mathsf{x}, \mathsf{w})$ are always satisfied.

Soundness. An AHP is ϵ-sound if for every field $\mathbb{F} \in \mathcal{F}$, relation-instance tuple $(\mathsf{i}, \mathsf{x}) \notin L_{\mathcal{R}}$ and prover P^* we have $\Pr[\langle \mathsf{P}^*, \mathsf{V}^{\mathsf{I}(\mathbb{F},\mathsf{i})}(\mathbb{F}, \mathsf{x})\rangle = 1] \leq \epsilon$.

Knowledge Soundness. An AHP is ϵ-knowledge-sound if there exists a polynomial-time knowledge extractor $\mathcal{E}$ such that for any prover P^*, field $\mathbb{F} \in \mathcal{F}$, relation i, instance x and auxiliary input z:

$$\Pr\left[(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R} : \mathsf{w} \leftarrow \mathcal{E}^{\mathsf{P}^*}(\mathbb{F}, \mathsf{i}, \mathsf{x}, z)\right] \geq \Pr[\langle \mathsf{P}^*(\mathbb{F}, \mathsf{i}, \mathsf{x}, z), \mathsf{V}^{\mathsf{I}(\mathbb{F},\mathsf{i})}(\mathbb{F}, \mathsf{x})\rangle = 1] - \epsilon$$

where $\mathcal{E}$ has oracle access to P^*, i.e., it can query the next message function of P^* (and rewind it) and obtain all the messages and polynomials returned by it.

Zero-Knowledge. The property of $(\mathsf{b}, \mathsf{C})-$Zero-Knowledge for AHPs models the existence of a simulator that can interact with a malicious verifier and can effectively simulate under two conditions: there is a bound b on the number of evaluation queries asked by the verifier; these queries need to satisfy an admissible test modelled a a circuit C. We say an AHP is zero-knowledge for some bound $\mathsf{b} = \mathsf{poly}(\lambda)$ and some efficient checker circuit C. We refer the reader to Section 4 in [CHM+20] for formal details.

Public coins and non-adaptive queries. In the remainder of this work, we only consider AHPs that are public coin and non-adaptive: the messages of the verifier are random elements and its checks are independent of the prover's messages.

2.5 Polynomial Commitment

Below we recall the definition of standard polynomial commitment scheme. The definition is taken verbatim from Section 6.1 of [CHM+20].

Definition 4 (Polynomial Commitment Scheme). *A polynomial commitment scheme over a field family $\mathcal{F}$ is a tuple* $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Trim}, \mathsf{Com}, \mathsf{Open}, \mathsf{Check})$ *such that*

- $\mathsf{Setup}(1^\kappa, D) \to \mathsf{pp}$. *On input a security parameter κ, and a maximum degree bound $D \in \mathbb{N}$,* Setup *samples public parameters* pp. *The parameters contain the description of a finite field $\mathbb{F} \in \mathcal{F}$.*
- $\mathsf{Trim}^{\mathsf{pp}}(1^\kappa, \mathbf{d}) \to (\mathsf{ck}, \mathsf{rk})$. *Given oracle access to public parameters* pp*, and on input a security parameter κ, and degree bounds $\mathbf{d}$,* Trim *deterministically computes a key pair* $(\mathsf{ck}, \mathsf{rk})$ *that is specialized to $\mathbf{d}$.*
- $\mathsf{Com}_{\mathsf{ck}}(\mathbf{p}, \mathbf{d}; \boldsymbol{\omega}) \to \mathbf{c}$. *On input* ck*, univariate polynomials $\mathbf{p} = (p_i)_{i=1}^n$ over the field $\mathbb{F}$ with $\deg(p_i) \leq d_i \leq D$,* Com *outputs commitments $\mathbf{c} = (c_i)_{i=1}^n$ to the polynomials $\mathbf{p}$. The randomness $\boldsymbol{\omega}$ is used if the commitments $\mathbf{c}$ are hiding.*
- $\mathsf{Open}_{\mathsf{ck}}(\mathbf{p}, \mathbf{d}, Q, \xi; \boldsymbol{\omega}) \to \pi$. *On input* ck*, univariate polynomials $\mathbf{p}$, degree bounds $\mathbf{d}$, a query set Q consisting of $(i, z) \in [n] \times \mathbb{F}$, and opening challenge ξ,* Open *outputs an evaluation proof π. The randomness must equal the one previously used in* Com.
- $\mathsf{Check}_{\mathsf{rk}}(\mathbf{c}, \mathbf{d}, Q, \mathbf{v}, \pi, \xi) \in \{0, 1\}$. *On input* rk*, commitments $\mathbf{c}$, degree bounds $\mathbf{d}$, query set Q, alleged evaluations $\mathbf{v} = (v_{(i,z)})_{(i,z)\in Q}$, evaluation proof π, and opening challenge ξ,* Check *outputs 1 iff π attests that, for every $(i, z) \in Q$, the polynomial p_i evaluates to $v_{(i,z)}$ at z.*

We recall a set of basic properties that the KZG scheme [KZG10] and its variant described in Marlin already satisfy.

Completeness. For every maximum degree bound $D \in \mathbb{N}$ and efficient adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l} \deg(\mathbf{p}) \leq \mathbf{d} \leq D \\ \implies \mathsf{Check}_{\mathsf{rk}}(\mathbf{c}, \mathbf{d}, Q, \mathbf{v}, \pi, \xi) \end{array} : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, D) \\ (\mathbf{p}, \mathbf{d}, Q, \xi, \boldsymbol{\omega}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{ck}, \mathsf{rk}) \leftarrow \mathsf{Trim}^{\mathsf{pp}}(1^\kappa, \mathbf{d}) \\ \mathbf{c} \leftarrow \mathsf{Com}(\mathsf{ck}, \mathbf{p}, \mathbf{d}; \boldsymbol{\omega}) \\ \mathbf{v} \leftarrow \mathbf{p}(Q) \\ \pi \leftarrow \mathsf{Open}(\mathsf{ck}, \mathbf{p}, \mathbf{d}, Q, \xi; \boldsymbol{\omega}) \end{array}\right] = 1$$

Succinctness. We require the commitments and the evaluation proofs to be of size independent of the degree of the polynomials, that is $|\mathbf{c}| = n \cdot \mathsf{poly}(\lambda), |\pi| = |Q| \cdot \mathsf{poly}(\lambda), |\mathsf{rk}| = n \cdot \mathsf{poly}(\lambda)$. We also require the verifier Check to run in time $(n + |Q|)n \cdot \mathsf{poly}(\lambda)$.

Extractability. From any adversary that can satisfactorily prove evaluations $\mathbf{v}$ and degree bounds $\mathbf{d}$ over polynomial commitments $\mathbf{c}$ we should be able to extract: *(i)* polynomials $\mathbf{p}$ consistent with the proofs, *(ii)* randomness $\boldsymbol{\omega}$ through which $\mathbf{c}$ opens to $\mathbf{p}$. The complete formal definition is quite involved; we refer the reader to [CHM+20, Definition 6.2] for details.

Polynomial Binding. We require that it is infeasible for any adversary to open the same commitment to two different polynomials. Formally, for every maximum degree bound $D \in \mathbb{N}$, security parameter κ and efficient adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{rr} \mathbf{p}_1 \neq \mathbf{p}_2 & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, D) \\ \wedge\ \mathbf{c} = \mathsf{Com}(\mathsf{ck}, \mathbf{p}_1, \mathbf{d}; \boldsymbol{\omega}_1) & : \ (\mathbf{c}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{d}, \boldsymbol{\omega}_1, \boldsymbol{\omega}_2) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \wedge\ \mathbf{c} = \mathsf{Com}(\mathsf{ck}, \mathbf{p}_2, \mathbf{d}; \boldsymbol{\omega}_2) & (\mathsf{ck}, \mathsf{rk}) \leftarrow \mathsf{Trim}^{\mathsf{pp}}(1^\kappa, \mathbf{d}) \end{array}\right] \leq \mathsf{negl}(\kappa)$$

Hiding. We require the existence of a stateful simulator $\mathsf{Sim} = (\mathsf{Sim.Setup}, \mathsf{Sim.Commit}, \mathsf{Sim.Open})$ such that an adversary cannot distinguish whether it is interacting with an honest execution or a simulated one. We refer the reader to [CHM+20] for the full definition.

Homomorphism. A PC is additively homomorphic if for every $D \in \mathbb{N}$, every $\mathbf{d}$ such that $d_i \leq D$, every query set Q, every opening challenge ξ, every $\mathbf{p}_1, \mathbf{p}_2, \boldsymbol{\omega}_1, \boldsymbol{\omega}_2$ that are consistent with the degree bound $\mathbf{d}$,[6]

$$\Pr\left[\mathbf{c}_1 + \mathbf{c}_2 = \mathsf{Com}_{\mathsf{ck}}(\mathbf{p}_1 + \mathbf{p}_2, \mathbf{d}; \boldsymbol{\omega}_1 + \boldsymbol{\omega}_2) \ : \ \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, D); \\ (\mathsf{ck}, \mathsf{rk}) = \mathsf{Trim}^{\mathsf{pp}}(1^\kappa, \mathbf{d}) \\ \mathbf{c}_1 = \mathsf{Com}_{\mathsf{ck}}(\mathbf{p}_1, \mathbf{d}; \boldsymbol{\omega}_1) \\ \mathbf{c}_2 = \mathsf{Com}_{\mathsf{ck}}(\mathbf{p}_2, \mathbf{d}; \boldsymbol{\omega}_2) \end{array}\right] = 1$$

2.5.1 The KZG scheme.

Below we recall the polynomial commitment scheme due to Kate–Zaverucha–Goldberg [KZG10], denoted by $\mathsf{PC}_{\mathsf{KZG}}$. The scheme is proven extractable under the strong Diffie–Hellman (SDH) assumption in the *algebraic group model (AGM)* [FKL18], polynomial binding under the discrete-log assumption, and perfectly hiding [CHM+20, KZG10]. For simplicity we omit challenge ξ used for batch opening as well as the Trim function, and set $\mathsf{ck} = \mathsf{rk} = \mathsf{pp}$. See Appendix B of [CHM+20] for details of such optimization techniques.

- $\mathsf{Setup}(1^\kappa, D) \to (g, g^\chi, \ldots, g^{\chi^D}, g, g^{\gamma\chi}, \ldots, g^{\gamma\chi^D}, h^\chi)$ where it determines a bilinear group public parameters $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$, with $g \in \mathbb{G}_1$ and $\chi, \gamma \in \mathbb{F}$ are randomly chosen. We denote exponentiation in $\mathbb{G}_i$ by $[\cdot]_i$.
- $\mathsf{Com}_{\mathsf{ck}}(p, D; \omega) \to [p(\chi) + \gamma\omega(\chi)]_1$, where $\omega \in \mathbb{F}_{<D}[X]$ is a random masking polynomial.
- $\mathsf{Open}_{\mathsf{ck}}(p, D, z; \omega)$ computes $W(X) = \frac{p(X)-p(z)}{X-z}$, $\bar{W}(X) = \frac{\omega(X)-\omega(z)}{X-z}$, $\mathit{\Pi} := [W(\chi) + \gamma\bar{W}(\chi)]_1$, $\bar{v} := \bar{W}(z)$ and outputs $\pi := (\mathit{\Pi}, \bar{v})$.
- $\mathsf{Check}_{\mathsf{rk}}(c, D, z, v, \pi)$ checks $e(\mathit{\Pi}, [\chi]_2/[z]_2) \stackrel{?}{=} e(C/([v]_1 \cdot [\gamma\bar{v}]_1), h)$.

3 AHP-to-CP-SNARK compiler

In this section, we present our general compiler from AHPs to commit-and-prove SNARKs.

3.1 Additional Preliminaries for Compiler

Auxiliary Commitment Scheme AC We will assume a commitment scheme AC for Auxiliary Commitments. They are "auxiliary" in the sense that they are used as auxiliary inputs to parts of the witness. We assume AC to satisfy the standard properties of (computational) binding and (computational or otherwise) hiding. As we explicitly support a *vector* $\mathbf{x} \in \mathbb{F}^d$ as committed message, the definition is specialized for a vector commitment scheme. Specifically we assume $\mathsf{AC} = (\mathsf{Gen}, \mathsf{Com})$ such that $\mathsf{AC.Gen}(1^\lambda, d) \to \mathsf{ack}$ is a randomized algorithm returning a commitment key ack for messages of dimension $d \in \mathbb{N}$, and $\mathsf{AC.Com}_{\mathsf{ack}}(\mathbf{x}; r)$ is a committing algorithm returning a commitment $\hat{c}$ on input $\mathbf{x} \in \mathbb{F}^d$ for $d = \mathsf{poly}(\lambda)$ and some randomness r.

[6] Note that we are implicitly assuming that commitment randomness is given in the form of polynomials, while the definition of PC in [CHM+20] considers it as a random seed to derive such masking polynomials.

Remark 1. In our concrete instantiations, we use the Pedersen vector commitment scheme (Sect. 2.3) as AC.

Commit-and-Prove Relation Our goal is to construct a general compiler that turns AHP for $\mathcal{R}$ into ARG for the relation over commitments $\mathcal{R}_{\mathsf{com}}$. Throughout we assume an indexed relation where the witness can be represented as a vector in $\mathbb{F}^n$.

Definition 5 (Commit-and-prove relation). *Let $\mathcal{R}$ be an indexed relation, AC a commitment scheme as defined above and ack an auxiliary commitment key in the range of $\mathsf{AC.Gen}$. We define the corresponding* commit-and-prove relation

$$\mathcal{R}_{\mathsf{com}} = \left\{ \begin{array}{c} ((\mathsf{i}, m, \ell, d, I_{\mathsf{com}}, (I_k)_{k\in[\ell]}, \mathsf{ack}), \\ (\mathsf{x}, (\hat{c}_k)_{k\in[\ell]}), ((\mathsf{w}_i)_{i\in[n]}, (r_k)_{k\in[\ell]})) \end{array} : \begin{array}{c} (\mathsf{i}, \mathsf{x}, (\mathsf{w}_i)_{i\in[n]}) \in \mathcal{R} \wedge \\ I_{\mathsf{com}} \subset [n] \ \wedge \ |I_{\mathsf{com}}| = \ell d \wedge \\ I_{\mathsf{com}} = \bigcup_{k\in[\ell]} I_k \ \wedge \ |I_k| = d \wedge \\ \hat{c}_k = \mathsf{AC.Com}_{\mathsf{ack}}((\mathsf{w}_i)_{i\in I_k}; r_k) \end{array} \right\}$$

3.2 Additional properties for AHP

We present basic properties that the underlying AHPs of PLONK, Marlin and Sonic already satisfy. First we describe our variant of Definition 3.3 from [CFF+20]: straight-line extractability for algebraic holographic proofs. We note that our definition is in the AHP model, while that in [CFF+20] is for Polynomially Holographic Proofs. The reason why we explicitly define witness-carrying polynomials is that our compiler needs to identify a minimum set of polynomials containing enough information about the whole witness, with which auxiliary commitments are shown to be consistent. Note that we also restrict WitExt to be deterministic so that it can be essentially seen as a witness decoding algorithm that works for both honest and malicious provers once and for all.

Definition 6 (AHP with S-straight-line extractor). *Fix AHP for indexed relation $\mathcal{R}$ and index set $S \subseteq \{(i,j) \ : \ i \in [\mathsf{k}], j \in [\mathsf{s}(i)]\}$. An AHP is* ϵ-knowledge sound with S-straight-line extractor *if there exists an efficient deterministic extractor WitExt such that for any P^*, every field $\mathbb{F} \in \mathcal{F}$, every index i and instance x,*

$$\Pr[(\mathsf{i}, \mathsf{x}, \mathsf{WitExt}(\{p_{i,j}(X)\}_{(i,j)\in S})) \in \mathcal{R}] \geq \Pr[\langle \mathsf{P}^*, \mathsf{V}^{\mathsf{I}(\mathbb{F},\mathsf{i})}\rangle(\mathbb{F}, \mathsf{x}) = 1] - \epsilon$$

where $\{p_{i,j}(X)\}_{(i,j)\in S}$ is a subset of the polynomials output by P^* in an execution of $\langle \mathsf{P}^*, \mathsf{V}^{\mathsf{I}(\mathbb{F},\mathsf{i})}\rangle(\mathbb{F}, \mathsf{x})$. Let W be a smallest set such that there exists an efficient extractor satisfying the condition above. Then we say that $\{p_{i,j}(X)\}_{(i,j)\in W}$ are witness-carrying polynomials of AHP. *If all witness-carrying polynomials are sent during the same round $\mathsf{k_w} \leq \mathsf{k}$, we call $\mathsf{k_w}$ a* witness-committing round.

Definition 7 (Disjoint witness-carrying polynomials). *We say that witness-carrying polynomials are* disjoint *if there exists some disjoint index sets $I_{i,j}$ such that $[n] = \bigcup_{(i,j)\in W} I_{i,j}$ and the corresponding WitExt independently invokes $\mathsf{WitExt}_{i,j}$ on $p_{i,j}$ to obtain $(\mathsf{w}_\iota)_{\iota\in I_{i,j}}$.*

Remark 2. Let $\mathsf{n_w} = |W|$. For Marlin and Sonic we have $\mathsf{n_w} = 1$ and $\mathsf{k_w} = 1$; for PLONK we have $\mathsf{n_w} = 3$ and $\mathsf{k_w} = 1$ and disjoint witness-carrying polynomials. In our compiler formalization, we always assume that W is such that $\mathsf{k_w}$ is minimum, and that AHP has a witness-committing round.

The following two definitions are needed to guarantee completeness of our compiler.

Definition 8 (Unique extraction). *We say that an S-straight-line extractor WitExt performs* unique extraction*, if for any honest prover P, every $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$, $\mathsf{WitExt}(\{p_{i,j}(X)\}_{(i,j)\in S}) = \mathsf{w}$, where $\{p_{i,j}(X)\}_{(i,j)\in S}$ is a subset of the polynomials output by P in an execution of $\langle \mathsf{P}(\mathsf{w}), \mathsf{V}^{\mathsf{I}(\mathbb{F},\mathsf{i})}\rangle(\mathbb{F}, \mathsf{x})$.*

Definition 9 (Decomposable witness-carrying polynomials). *Let W be an index set of witness-carrying polynomials of AHP. We say that polynomials $(p_{i,j}(X))_{(i,j)\in W}$ of AHP are* decomposable *if there exists an efficient function $\mathsf{Decomp}((p_{i,j}(X))_{(i,j)\in W}, I) \to (p^{(1)}_{i,j}(X), p^{(2)}_{i,j}(X))_{(i,j)\in W}$ such that it satisfies the following properties for any $I \subset [n]$.*

- Additive decomposition: $p_{i,j}(X) = p^{(1)}_{i,j}(X) + p^{(2)}_{i,j}(X)$ *for* $(i,j) \in W$.
- Degree preserving: $\deg(p^{(1)}_{i,j}(X))$ *and* $\deg(p^{(2)}_{i,j}(X))$ *are at most* $\deg(p_{i,j}(X))$ *for* $(i,j) \in W$.

– Non-overlapping: *Let* $\mathsf{w} = \mathsf{WitExt}((p_{i,j}(X))_{(i,j)\in W})$, $\mathsf{w}^{(1)} = \mathsf{WitExt}((p^{(1)}_{i,j}(X))_{(i,j)\in W})$, *and* $\mathsf{w}^{(2)} = \mathsf{WitExt}((p^{(2)}_{i,j}(X))_{(i,j)\in W})$. *Then*

$$(\mathsf{w}_i)_{i\in I} = (\mathsf{w}^{(1)}_i)_{i\in I} \qquad (\mathsf{w}_i)_{i\notin I} = (\mathsf{w}^{(2)}_i)_{i\notin I} \qquad (\mathsf{w}^{(1)}_i)_{i\notin I} = 0 \qquad (\mathsf{w}^{(2)}_i)_{i\in I} = 0$$

3.3 Our compiler

In order to prove the relation above, our compiler will use a commit-and-prove NIZKAoK subprotocol $\mathsf{CP}_{\mathtt{lnk}}$ for following relation.

Definition 10 (Commitment-linking relation). *Suppose* AHP *with* W*-straight-line extractor and witness carrying polynomials, a polynomial commitment scheme* PC*, and an auxiliary commitment scheme* AC *are fixed. We define the linking relation*

$$\mathcal{R}_{\mathtt{lnk}} = \left\{ \begin{array}{c} ((n,\ell,d,I_{\mathsf{com}},(I_k)_{k\in[\ell]},\mathsf{ck},\mathsf{ack}),\\ ((\hat{c}_k)_{k\in[\ell]},\mathbf{v},Q,\\ (c^{\mathsf{com}}_{i,j}(X),c^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W},\\ ((p^{\mathsf{com}}_{i,j}(X),p^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W},\\ (\omega^{\mathsf{com}}_{i,j}(X),\omega^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W},\\ (r_k)_{k\in[\ell]})) \end{array} : \begin{array}{c} I_{\mathsf{com}}\subset[n] \ \wedge\ |I_{\mathsf{com}}| = \ell d \ \wedge\\ I_{\mathsf{com}} = \bigcup_{k\in[\ell]} I_k \ \wedge\ |I_k| = d \ \wedge\\ c^{\mathsf{com}}_{i,j} = \mathsf{PC.Com}_{\mathsf{ck}}(p^{\mathsf{com}}_{i,j}(X),\mathsf{d}(|\mathsf{i}|,i,j);\omega^{\mathsf{com}}_{i,j}) \ \wedge\\ c^{\mathsf{mid}}_{i,j} = \mathsf{PC.Com}_{\mathsf{ck}}(p^{\mathsf{mid}}_{i,j}(X),\mathsf{d}(|\mathsf{i}|,i,j);\omega^{\mathsf{mid}}_{i,j}) \ \wedge\\ \hat{c}_k = \mathsf{AC.Com}_{\mathsf{ack}}((\mathsf{w}_i)_{i\in I_k};r_k) \ \textit{where}\\ \mathsf{w} = \mathsf{WitExt}((p^{\mathsf{com}}_{i,j}(X)+p^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}) \ \wedge\\ v_{((i,j),z)} = p^{\mathsf{com}}_{i,j}(z)+p^{\mathsf{com}}_{i,j}(z) \ \textit{for all}\ ((i,j),z)\in Q \ \textit{such that}\ (i,j)\in W \end{array} \right\}$$

Remark 3. Although the correctness of polynomial evaluation (i.e., the condition "$v_{((i,j),z)} = p^{\mathsf{com}}_{i,j}(z) + p^{\mathsf{com}}_{i,j}(z)$") is part of $\mathcal{R}_{\mathtt{lnk}}$, we remark that this is redundant since it is to be proven by the opening algorithm of PC outside $\mathsf{CP}_{\mathtt{lnk}}$ anyway. Looking ahead, security proof of our compiler indeed holds even without showing such a condition within $\mathsf{CP}_{\mathtt{lnk}}$. We rather include this for the ease of proving knowledge soundness of $\mathsf{CP}_{\mathtt{lnk}}$; in concrete instantiations, an extractor of $\mathsf{CP}_{\mathtt{lnk}}$ typically needs to extract what is committed to $c^{\mathsf{mid}}_{i,j}$ by internally invoking an extractor of PC, which however is only guaranteed to succeed if the evaluation proof is valid. Hence, by letting $\mathsf{CP}_{\mathtt{lnk}}$ take care of evaluation proof by default we can easily make such an argument go through. In later sections our $\mathsf{CP}_{\mathtt{lnk}}$ for Sonic takes advantage of this generalization, while the ones for PLONK and Marlin don't since they create a special evaluation proof independent of the AHP query phase.

Intuition about the compiler. The compiler in Figure 1 is close to those in Marlin [CHM+20], Lunar [CFF+20] and DARK [BFS20]. One important difference is the use of polynomial decomposition where the prover will commit separately to each of the "parts" of the witness-carrying polynomials. This separate commitment will allow efficiently proving the commitment-linking relation.

Theorem 1. *Let* $\mathcal{F}$ *be a field family and* $\mathcal{R}$ *be an indexed relation. Consider the following components:*

– $\mathsf{AHP} = (\mathsf{k},\mathsf{s},\mathsf{d},\mathsf{I},\mathsf{P},\mathsf{V})$ *is a knowledge sound AHP for* $\mathcal{R}$ *with* W*-straight-line unique extractor, and with a decomposition function* Decomp *for witness-carrying polynomials* $(p_{i,j}(X))_{(i,j)\in W}$*;*

– $\mathsf{PC} = (\mathsf{Setup},\mathsf{Com},\mathsf{Open},\mathsf{Check})$ *is a homomorphic polynomial commitment over* $\mathcal{F}$ *with binding and extractability;*

– $\mathsf{CP}_{\mathtt{lnk}} = (\mathcal{S}_{\mathtt{lnk}},\mathcal{P}_{\mathtt{lnk}},\mathcal{V}_{\mathtt{lnk}})$ *is non-interactive argument of knowledge for* $\mathcal{R}_{\mathtt{lnk}}$ *(Definition 10)*

Then the construction of $\mathsf{ARG} = (\mathcal{S},\mathcal{I},\mathcal{P},\mathcal{V})$ *in Fig. 1 is a preprocessing argument system for the relation* $\mathcal{R}_{\mathsf{com}}$*. If* $\mathsf{CP}_{\mathtt{lnk}}$ *is zero-knowledge and* AHP *is zero-knowledge as defined in Definition 3, then* ARG *is also zero-knowledge.*

Moreover, if witness-carrying polynomials are disjoint and $I_{\mathsf{com}}\subset I_{i^*,j^*}$ *for some* $(i^*,j^*)\in W$*, then the above claim holds even if* $\mathsf{CP}_{\mathtt{lnk}}$ *shows a variant of* $\mathcal{R}_{\mathtt{lnk}}$ *such that all "*$(i,j)\in W$*" are replaced by* (i^*,j^*) *and* WitExt *is replaced by* $\mathsf{WitExt}_{i^*,j^*}$*.*

Remark 4. While in the description of our compiler we generically commit all polynomials with the same type of polynomial commitments, our instantiations use some ad-hoc tweaks. In particular, we need to commit to the witness carrying polynomials using a special version of KZG (see for example the input format of commitments in Figure 5) different than the one we use for the rest of the oracle polynomials. Note that this is a standard optimization trick already used in previous works, e.g., [CHM+20],[GWC19],[MBKM19], and we are still able to satisfy the security requirements of the general compiler this way.

Proof. **Completeness.** It follows from properties of the Decomp function, uniqueness of extraction, and homomorphism of PC. Concretely, since PC is homomorphic and decomposition of polynomials is additive and degree-preserving, it holds that

$$\begin{aligned} c^{\mathsf{com}}_{i,j} + c^{\mathsf{mid}}_{i,j} &= \mathsf{PC.Com}_{\mathsf{ck}}(p^{\mathsf{com}}_{i,j}(X) + p^{\mathsf{mid}}_{i,j}(X), \mathsf{d}(|\mathsf{i}|, i, j); \omega^{\mathsf{com}}_{i,j} + \omega^{\mathsf{mid}}_{i,j}) \\ &= \mathsf{PC.Com}_{\mathsf{ck}}(p_{i,j}(X), \mathsf{d}(|\mathsf{i}|, i, j); \omega_{i,j}). \end{aligned}$$

Hence $\mathcal{V}$ always accepts evaluation of $p_{i,j}(X)$ during $\mathsf{PC.Check}_{\mathsf{rk}}$. Moreover, due to uniqueness of extraction and properties of Decomp, if the instance-witness pair is in $\mathcal{R}_{\mathsf{com}}$ then we have that the inputs to $\mathsf{CP}_{\mathtt{lnk}}$ prover satisfy relation $\mathcal{R}_{\mathtt{lnk}}$. In particular,

$$\mathsf{WitExt}((p^{\mathsf{com}}_{i,j}(X) + p^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}) = \mathsf{WitExt}((p_{i,j}(X))_{(i,j)\in W}) = \mathsf{w}.$$

Knowledge soundness. It follows from homomorphism and binding of PC, knowledge soundness of $\mathsf{CP}_{\mathtt{lnk}}$ and W-straight-line extractability of AHP. Our goal is to extract a pair of witness $((\mathsf{w}_i)_{i\in[n]}, (r_k)_{k\in[\ell]})$ that satisfies relation $\mathcal{R}_{\mathsf{com}}$, given index $(\mathsf{i}, m, \ell, d, I_{\mathsf{com}}, (I_k)_{k\in[\ell]}, \mathsf{ack})$ and statement $(\mathsf{x}, (\hat{c}_k)_{k\in[\ell]})$. Namely, $(\mathsf{w}_i)_{i\in[n]}$ such that $(\mathsf{i}, \mathsf{x}, (\mathsf{w}_i)_{i\in[n]}) \in \mathcal{R}$ and randomness r_k for commitment $\hat{c}_k$ such that its opening is consistent with $(\mathsf{w}_i)_{i\in I_k}$. At the high-level the extractor $\mathcal{E}_{\mathsf{ARG}}$ works as follows:

1. Extract the polynomials from the polynomial commitments sent at each round through the extractor for the polynomial commitments;
2. From these, for each $(i, j) \in W$ reconstruct the witness-carrying polynomials as $\tilde{p}_{i,j}(X)$;
3. On the other hand, extract auxiliary commitment randomness $(\tilde{r}_k)_{k\in[\ell]}$ as well as decomposed witness-carrying polynomials $(p^{\mathsf{com}}_{i,j}(X), p^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}$ such that $p_{i,j}(X) = p^{\mathsf{com}}_{i,j}(X) + p^{\mathsf{mid}}_{i,j}(X)$, by invoking the linking extractor.
4. Extract witness $(\tilde{\mathsf{w}}_i)_{i\in[n]}$ from the W-straight-line extractor as $\mathsf{WitExt}(\tilde{p}_{i,j}(X))_{(i,j)\in W}$;
5. Return $((\tilde{\mathsf{w}}_i)_{i\in[n]}, (\tilde{r}_k)_{k\in[\ell]})$.

A more detailed version of the proof follows.
Suppose that $\tilde{\mathcal{P}}$ convinces $\mathcal{V}$ of ARG with non-negligible probability. Assuming the existence of extractors $\mathcal{E}_{\mathsf{PC}}$ for PC and $\mathcal{E}_{\mathtt{lnk}}$ for $\mathsf{CP}_{\mathtt{lnk}}$, we show the existence of another extractor $\mathcal{E}_{\mathsf{ARG}}$ that outputs a valid witness $\tilde{\mathsf{w}}$ for $\mathcal{R}_{\mathsf{com}}$ with non-negligible probability, given access to $\tilde{\mathcal{P}}$.

- First we construct an adversary $\mathcal{A}_{\mathsf{PC}}$ against the extractability game for PC. The $\mathcal{A}_{\mathsf{PC}}$ receives ck and random coins as input, and internally invokes $\tilde{\mathcal{P}}$ to obtain a set of commitments $(\tilde{c}_{i,j})_{i\in[\mathsf{k}], j\in[\mathsf{s}(i)]}$, where for $(i, j) \in W$ it holds that $\tilde{c}_{i,j} = \tilde{c}^{\mathsf{com}}_{i,j} + \tilde{c}^{\mathsf{mid}}_{i,j}$.
- We then invoke an extractor $\mathcal{E}_{\mathsf{PC}}$ who, given the same input as that of $\mathcal{A}_{\mathsf{PC}}$, outputs a set of polynomials $\tilde{\mathbf{p}} := (\tilde{p}_{i,j})_{i\in[\mathsf{k}], j\in[\mathsf{s}(i)]}$. If the cheating prover $\tilde{\mathcal{P}}$ convinces the ARG verifier $\mathcal{V}$, then the evaluation proof $\tilde{\pi}_{\mathsf{Eval}}$ is valid w.r.t. the alleged evaluations $\tilde{\mathbf{v}} := (\tilde{v}_{i,j})_{i\in[\mathsf{k}], j\in[\mathsf{s}(i)]}$. Hence if $\mathcal{E}_{\mathsf{PC}}$ fails to extract polynomials (i.e., $\tilde{\mathbf{p}}(Q) \neq \tilde{\mathbf{v}}$), then $\mathcal{A}_{\mathsf{PC}}$ wins the extractability game, which, however, happens with negligible probability under our assumption. So below we assume that with overwhelming probability $\tilde{\mathbf{p}}(Q) = \tilde{\mathbf{v}}$.
- Second we construct another adversary $\mathcal{A}_{\mathtt{lnk}}$ against the knowledge soundness game for $\mathsf{CP}_{\mathtt{lnk}}$. The $\mathcal{A}_{\mathtt{lnk}}$ receives a statement for $\mathcal{R}_{\mathtt{lnk}}$ and random coins as input, and internally invokes $\tilde{\mathcal{P}}$ to obtain a linking proof $\tilde{\pi}_{\mathtt{lnk}}$.
- We then invoke another extractor $\mathcal{E}_{\mathtt{lnk}}$ who, given the same input as that of $\mathcal{A}_{\mathtt{lnk}}$, outputs the corresponding witness $((\tilde{p}^{\mathsf{com}}_{i,j}(X), \tilde{p}^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}, (\tilde{\omega}^{\mathsf{com}}_{i,j}(X), \tilde{\omega}^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}, (\tilde{r}_k)_{k\in[\ell]})$. If the cheating prover $\tilde{\mathcal{P}}$ convinces the ARG verifier $\mathcal{V}$, then linking proof $\tilde{\pi}_{\mathtt{lnk}}$ is valid w.r.t. the commitments $(\hat{c}_k)_{k\in[\ell]}$ and $(\tilde{c}^{\mathsf{com}}_{i,j}(X), \tilde{c}^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}$. Hence if $\mathcal{E}_{\mathtt{lnk}}$ fails to extract the witness, then $\mathcal{A}_{\mathtt{lnk}}$ wins the knowledge soundness game, which, however, happens with negligible probability under our assumption.
- Now we construct a cheating prover $\tilde{\mathsf{P}}$ for AHP. The $\tilde{\mathsf{P}}$ internally invokes $\mathcal{E}_{\mathsf{PC}}$ and $\mathcal{E}_{\mathtt{lnk}}$ to obtain $\tilde{\mathbf{p}}$ and $((\tilde{p}^{\mathsf{com}}_{i,j}(X), \tilde{p}^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}, (\tilde{\omega}^{\mathsf{com}}_{i,j}(X), \tilde{\omega}^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}, (\tilde{r}_k)_{k\in[\ell]})$. The polynomials of the latter satisfy relation $\mathcal{R}_{\mathtt{lnk}}$, i.e.,

$$\begin{aligned} \tilde{c}^{\mathsf{com}}_{i,j} &= \mathsf{PC.Com}_{\mathsf{ck}}(\tilde{p}^{\mathsf{com}}_{i,j}(X), \mathsf{d}(|\mathsf{i}|, i, j); \tilde{\omega}^{\mathsf{com}}_{i,j}) \\ \tilde{c}^{\mathsf{mid}}_{i,j} &= \mathsf{PC.Com}_{\mathsf{ck}}(\tilde{p}^{\mathsf{mid}}_{i,j}(X), \mathsf{d}(|\mathsf{i}|, i, j); \tilde{\omega}^{\mathsf{mid}}_{i,j}) \end{aligned}$$

Due to the homomorphic property of PC, we have

$$\tilde{c}^{\mathsf{com}}_{i,j} + \tilde{c}^{\mathsf{mid}}_{i,j} = \mathsf{PC.Com_{ck}}(\tilde{p}^{\mathsf{com}}_{i,j}(X) + \tilde{p}^{\mathsf{mid}}_{i,j}(X), \mathsf{d}(|\mathsf{i}|, i, j); \tilde{\omega}^{\mathsf{com}}_{i,j} + \tilde{\omega}^{\mathsf{mid}}_{i,j})$$

If $\tilde{p}^{\mathsf{com}}_{i,j}(X) + \tilde{p}^{\mathsf{mid}}_{i,j}(X) \neq \tilde{p}_{i,j}(X)$ for some $(i,j) \in W$ (recall that the latter was extracted by $\mathcal{E}_{\mathsf{PC}}$), then $\tilde{\mathsf{P}}$ aborts, which only happens with negligible probability as the ability to find such polynomials breaks binding of PC w.r.t. $\tilde{c}^{\mathsf{com}}_{i,j} + \tilde{c}^{\mathsf{mid}}_{i,j}$. Hence we may assume that $\tilde{p}^{\mathsf{com}}_{i,j}(X) + \tilde{p}^{\mathsf{mid}}_{i,j}(X) = \tilde{p}_{i,j}(X)$. In that case, note that $\mathcal{R}_{\mathtt{lnk}}$ relation also guarantees for every $k \in [\ell]$

$$\hat{c}_k = \mathsf{AC.Com_{ack}}((\tilde{\mathsf{w}}_i)_{i \in I_k}; \tilde{r}_k)$$

where $\tilde{\mathsf{w}} = \mathsf{WitExt}((\tilde{p}^{\mathsf{com}}_{i,j}(X) + \tilde{p}^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}) = \mathsf{WitExt}((\tilde{p}_{i,j}(X))_{(i,j)\in W})$.

To sum up, as long as (1) $\mathcal{E}_{\mathsf{PC}}$ is successful, i.e., $\tilde{\mathsf{P}}$ outputs polynomials $\tilde{\mathbf{p}}$ which form correct opening to $\tilde{\mathbf{c}}$, (2) $\mathcal{E}_{\mathtt{lnk}}$ is successful, i.e., $\tilde{\mathsf{P}}$ internally obtains polynomials satisfying $\mathcal{R}_{\mathtt{lnk}}$, and (3) witness-carrying polynomials extracted by $\mathcal{E}_{\mathsf{PC}}$ and $\mathcal{E}_{\mathtt{lnk}}$ are identical, it holds that V accepts whenever $\mathcal{V}$ accepts. This indicates that $\tilde{\mathsf{P}}$ convinces V with non-negligible probability if $\tilde{\mathcal{P}}$ convinces $\mathcal{V}$.

We finally let $\mathcal{E}_{\mathsf{ARG}}$ invoke the W-straight-line extractor WitExt of AHP on witness-carrying polynomials $(\tilde{p}_{i,j}(X))_{(i,j)\in W}$ outputted by $\tilde{\mathsf{P}}$. By definition of the extractor $(\tilde{\mathsf{w}}_i)_{i\in[n]} = \mathsf{WitExt}((\tilde{p}_{i,j}(X))_{(i,j)\in W})$ satisfies $(\mathsf{i}, \mathsf{x}, (\tilde{\mathsf{w}}_i)_{i\in[n]}) \in \mathcal{R}$. Moreover, the committed part of witness $(\tilde{\mathsf{w}}_i)_{i\in I_k}$ is guaranteed to form correct opening to $\hat{c}_k$ with extracted randomness $\tilde{r}_k$, thanks to the linking relation $\mathcal{R}_{\mathtt{lnk}}$. This implies that a pair of extracted witness $((\tilde{\mathsf{w}}_i)_{i\in[n]}, (\tilde{r}_k)_{k\in[\ell]})$ satisfies $\mathcal{R}_{\mathsf{com}}$.

We argue a special case where witness carrying-polynomials are disjoint. In that case, we assume $\mathsf{CP}_{\mathtt{lnk}}$ only guarantees that $(\tilde{\mathsf{w}}_\iota)_{\iota\in I_{i^*,j^*}} = \mathsf{WitExt}_{i^*,j^*}(\tilde{p}_{i^*,j^*})$ are consistent with auxiliary commitments. This still retains knowledge soundness, since when WitExt is invoked on all extracted witness-carrying polynomials at the end of $\mathcal{E}_{\mathsf{ARG}}$, we know that WitExt invokes $\mathsf{WitExt}_{i,j}$ independently on each $\tilde{p}_{i,j}$ to obtain $(\tilde{\mathsf{w}}_\iota)_{\iota\in I_{i,j}}$ and index sets $I_{i,j}$ are disjoint.

Zero knowledge. Our proof closely follows that of the compiler in [CHM+20] (Theorem 8.4). We provide an overview and we stress when our proof diverges from theirs.

We construct a simulator $\mathsf{Sim}_{\mathsf{ARG}}$ by using the simulators $\mathsf{Sim}_{\mathsf{PC}}$ from the polynomial commitment (hiding property), the zero-knowledge simulator $\mathsf{Sim}_{\mathtt{lnk}}$ of $\mathsf{CP}_{\mathtt{lnk}}$ and the zero-knowledge simulator $\mathsf{Sim}_{\mathsf{AHP}}$ of AHP.

Below we require that $\mathsf{CP}_{\mathtt{lnk}}$ is zero-knowledge with simulator $\mathsf{Sim}_{\mathtt{lnk}}$. Zero-knowledge for non-interactive proof systems is standard and is a straightforward extension of the one we define for interactive-arguments (see for example [Gro16]).

Consider a (stateful) malicious verifier $\tilde{V}$. After receiving the srs it outputs a tuple (indexer, statement, witness):

$$((\mathsf{i}, m, \ell, d, I_{\mathsf{com}}, (I_k)_{k\in[\ell]}, \mathsf{ack}), (\mathsf{x}, (\hat{c}_k)_{k\in[\ell]}), ((\mathsf{w}_i)_{i\in[n]}, (r_k)_{k\in[\ell]})).$$

During the online (proving) stage the input of $\mathsf{Sim}_{\mathsf{ARG}}$ consists of the statement $((\mathsf{i}, m, \ell, d, I_{\mathsf{com}}, (I_k)_{k\in[\ell]}, \mathsf{ack}), (\mathsf{x}, (\hat{c}_k)_{k\in[\ell]}))$ as well as the following elements computed by the setup simulator:

- the integer D computed as in the protocol setup for size bound N;
- the output of $\mathsf{Sim}_{\mathsf{PC}}.\mathsf{Setup}$ (to obtain simulated parameters for polynomial commitment)
- the output of $\mathsf{Sim}_{\mathtt{lnk}}.\mathsf{Setup}$ (to obtain simulated parameters for $\mathsf{CP}_{\mathtt{lnk}}$)).

For each round $i \in [k]$, the simulator:

- receives challenge ρ_i from the verifier and forwards it to the AHP simulator $\mathsf{Sim}_{\mathsf{AHP}}(\mathsf{x})$.
- samples commitment randomness and use $\mathsf{Sim}_{\mathsf{PC}}$ to simulate all the commitments to oracle polynomials in that round. This step is the same for both branches of Step 3 in the Online Phase of Figure 1 (both witness-carrying polynomials and not).
- sends commitments to verifier.

After the online phase the simulator runs $\mathsf{Sim}_{\mathtt{lnk}}$ and sends the output to the verifier. Then:

- after receiving ρ_{k+1} from the verifier, it runs the (honest) query algorithm to obtain a list of polynomials queries Q from the transcript;
- checks that they are admissible using the checker circuit C (see definition of AHP zero-knowledge in Definition 3)
- obtains simulated evaluations. In order to do this, it can run the indexer on input i to actually obtain polynomials $p_{0,j}$-s.For the evaluation points of the online phase, it forwards the query list Q to $\mathsf{Sim}_{\mathsf{AHP}}$.

Finally $\mathsf{Sim}_{\mathsf{ARG}}$ simulates the evaluation proofs as follows:

- It receives an opening challenge ξ
- it simulates the evaluation proofs for polynomials through $\mathsf{Sim}_{\mathsf{PC}}$

We now argue this simulated view is indistinguishable from that of a malicious verifier. Recall from definition of zero-knowledge for AHPs that the $\mathsf{Sim}_{\mathsf{AHP}}$ can produce an indistinguishable transcript whenever the protocol carries out at most b queries that are admissible (i.e., they satisfy checker circuit C). Since this is the case for our protocol we can invoke this property. It is then straightforward to argue that $\mathsf{Sim}_{\mathsf{PC}}$ leaks nothing more about these evaluations because of the hiding property of the polynomial commitments. Invoking the zero-knowledge property of $\mathsf{CP}_{\mathtt{lnk}}$ completes the proof. □

4 Compressed Σ-protocol for Equality

We describe how to construct an efficient protocol proving equality of committed vectors, following the framework due to Attema, Cramer and Fehr [AC20, ACF20]. This allows us to instantiate $\mathsf{CP}_{\mathtt{lnk}}$ with proof size of only $O(\log(\ell d))$ when ℓ Pedersen commitments are received as inputs.

4.1 AmComEq: Amortization of ℓ commitment equality proofs

In our application, we would like to show equality of vectors within a single commitment containing vector of size ℓd (corresponding to a polynomial commitment) and ℓ chunks of vector of size d in multiple Pedersen commitments. Concretely, our goal is to give an efficient protocol for relation

$$\mathcal{R}_{\mathsf{AmComEq}} = \left\{ \begin{array}{c} ((\boldsymbol{g}, \mathbf{h}, \mathbf{G}, \mathbf{H}, d, d', d'', \ell), \\ (C, \hat{C}_1, \ldots, \hat{C}_\ell), \\ (\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_\ell)) \end{array} : \begin{array}{c} C = \boldsymbol{g}^{\mathbf{w}} \mathbf{h}^{\boldsymbol{\alpha}}, \hat{C}_i = \mathbf{G}^{\mathbf{w}_i} \mathbf{H}^{\boldsymbol{\beta}_i}, \\ \boldsymbol{g} \in \mathbb{G}^{\ell d}, \mathbf{G} \in \mathbb{G}^{d}, \mathbf{h} \in \mathbb{G}^{d'}, \mathbf{H} \in \mathbb{G}^{d''}, \\ \mathbf{w}_i \in \mathbb{Z}_q^d, \boldsymbol{\alpha} \in \mathbb{Z}_q^{d'}, \boldsymbol{\beta}_i \in \mathbb{Z}_q^{d''}, \mathbf{w} = [\mathbf{w}_1, \ldots, \mathbf{w}_\ell] \end{array} \right\} \tag{1}$$

where we assume d' and d'' are small constants (for concrete instantiations in later sections, we only need $d' \leq 4$ and $d'' = 1$). Our starting point is a naïve ComEq Σ-protocol proving equality of vectors committed in two Pedersen commitments, with proof size of $O(d)$ (see Appendix A). To avoid invoking ComEq individually for many commitments we first amortize the statements. The main idea of amortization is to introduce additional challenge $x \in \mathbb{Z}_q$ and use it to take a random linear combination in the exponent. A similar idea has appeared in many contexts, e.g., amortization of many range proofs in $\mathsf{Bulletproofs}$ [BBB+18] and batch verification of EdDSA signatures. Note that the protocol below can be seen as a slight variant of direct instantiation of the technique described by Attema–Cramer–Fehr [ACF20, §3.4]. For completeness, in Fig. 11 we include a version derived by invoking their amortization of multiple group homomorphisms in a black-box way. The advantage of our $\mathsf{AmComEq}$ over Fig. 11 is that it allows to save ℓ group exponentiations on verifier's side (i.e., computation of $\tilde{\mathbf{H}}$), by letting the prover precompute amortization of commitment randomness $\boldsymbol{\beta}_i$. However, the proof sizes are identical.

Note also that the protocol is 4-round where the first message is a challenge, which does not really fit into the format of standard Fiat–Shamir transform [FS87]. However, one can easily make it applicable by either introducing additional round where the prover first sends a dummy randomness, or let them send A before receiving challenge x.

Theorem 2. $\mathsf{AmComEq}$ *is a four-move protocol for the relation* $\mathcal{R}_{\mathsf{AmComEq}}$. *It is perfectly complete, computationally* $(\ell, 2)$*-special sound if finding non-trivial discrete-log relation for the generators* $[\boldsymbol{g}, \mathbf{h}]$ *is hard, and special HVZK. Moreover, the communication costs are:*

- $\mathcal{P} \to \mathcal{V}$*:* 2 *elements of* $\mathbb{G}$ *and* $\ell d + d' + d''$ *elements of* $\mathbb{Z}_q$.
- $\mathcal{V} \to \mathcal{P}$*:* 2 *elements of* $\mathbb{Z}_q$.

Proof. **Completeness.** It follows by inspection.

$(\ell, 2)$-special soundness. For every execution $j \in [\ell]$, we fix the first challenge x_j. Given two accepting transcripts $(x_j, A_j, \hat{A}_j, e_j, \mathbf{z}_j, \boldsymbol{\omega}_j, \boldsymbol{\Omega}_j)$ and $(x_j, A_j, \hat{A}_j, e'_j, \mathbf{z}'_j, \boldsymbol{\omega}'_j, \boldsymbol{\Omega}'_j)$ for the same x_j but with distinct e_j and e'_j, we extract valid witness w.r.t C from the first verification condition $\boldsymbol{g}^{\mathbf{z}_j} \mathbf{h}^{\boldsymbol{\omega}_j} = A_j C^{e_j}$ and $\boldsymbol{g}^{\mathbf{z}'_j} \mathbf{h}^{\boldsymbol{\omega}'_j} = A_j C^{e'_j}$:

$$\tilde{\mathbf{w}}_j = (\mathbf{z}_j - \mathbf{z}'_j)/(e_j - e'_j), \quad \tilde{\boldsymbol{\alpha}}_j = (\boldsymbol{\omega}_j - \boldsymbol{\omega}'_j)/(e_j - e'_j)$$

Protocol ECLIPSE compiler

Setup $\mathcal{S}(1^\kappa, \mathsf{N}, d)$**.** The setup $\mathcal{S}$ on input a security parameter $\kappa \in \mathbb{N}$ and size bound $\mathsf{N} \in \mathbb{N}$, uses N to compute a maximum degree bound D, samples $\mathsf{pp} \leftarrow \mathsf{PC.Setup}(1^\kappa, D)$, samples $\mathsf{ack} \leftarrow \mathsf{AC.Setup}(1^\kappa, d)$, and then outputs $\mathsf{srs} := (\mathsf{pp}, \mathsf{ack})$. The integer D is computed to be the maximum degree bound in AHP for indices of size N. In other words,

$$D := \max\{\mathsf{d}(\mathsf{N}, i, j) | i \in \{0, 1, \ldots, \mathsf{k}(\mathsf{N})\}, j \in \{1, \ldots, \mathsf{s}(i)\}\}$$

Indexer $\mathcal{I}^{\mathsf{srs}}(\mathsf{i}, I_{\mathsf{com}}, (I_k)_{k\in[\ell]})$**.** The indexer $\mathcal{I}$ upon input i, commitment index sets I_{com}, $(I_k)_{k\in[\ell]}$ and given oracle access to srs, deduces the field $\mathbb{F} \in \mathcal{F}$ contained in $\mathsf{srs} = (\mathsf{pp}, \mathsf{ack})$, runs the AHP indexer I on $(\mathbb{F}, \mathsf{i})$ to obtain $\mathsf{s}(0)$ polynomials $(p_{0,j})_{j=1}^{\mathsf{s}(0)} \in \mathbb{F}[X]$ of degrees at most $(\mathsf{d}(|\mathsf{i}|, 0, j))_{j=1}^{\mathsf{s}(0)}$. Then it proceeds by computing $(\mathsf{ck}, \mathsf{rk}) := \mathsf{PC.Trim}^{\mathsf{pp}}(\mathbf{d})$, where $\mathbf{d} = (\mathsf{d}(|\mathsf{i}|, i, j))_{i\in[\mathsf{k}], j\in[\mathsf{s}(i)]}$, and generating (de-randomized) commitments to index polynomials $(c_{0,j})_{j=1}^{\mathsf{s}(0)} = \mathsf{PC.Com}_{\mathsf{ck}}((p_{0,j})_{j=1}^{\mathsf{s}(0)})$. The indexer outputs $\mathsf{ipk} := (\mathsf{ck}, \mathsf{i}, (p_{0,j})_{j=1}^{\mathsf{s}(0)}, (c_{0,j})_{j=1}^{\mathsf{s}(0)}, \mathsf{ack}, I_{\mathsf{com}}, (I_k)_{k\in[\ell]})$ and $\mathsf{ivk} := (\mathsf{rk}, (c_{0,j})_{j=1}^{\mathsf{s}(0)}, \mathsf{ack}, I_{\mathsf{com}}, (I_k)_{k\in[\ell]})$.

Input. The ARG prover $\mathcal{P}$ receives $(\mathsf{ipk}, (\mathsf{x}, (\hat{c}_k)_{k\in[\ell]}), ((\mathsf{w}_i)_{i\in[n]}, (r_k)_{k\in[\ell]}))$ and the verifier $\mathcal{V}$ receives $(\mathsf{ivk}, (\mathsf{x}, (\hat{c}_k)_{k\in[\ell]}))$. If for any $k \in [\ell]$, $\hat{c}_k \neq \mathsf{AC.Com}_{\mathsf{ack}}((\mathsf{w}_i)_{i\in I_{\mathsf{com}}}; r_k)$, then $\mathcal{P}$ aborts.

Online phase. For every round $i \in [\mathsf{k}]$, $\mathcal{P}$ and $\mathcal{V}$ run the i-th round of interaction between the AHP prover $\mathsf{P}(\mathbb{F}, \mathsf{i}, \mathsf{x}, \mathsf{w})$ and verifier $\mathsf{V}(\mathbb{F}, \mathsf{x})$.

1. $\mathcal{V}$ receives random challenge $\rho_i \in \mathbb{F}$ from V, and forwards it to $\mathcal{P}$.
2. $\mathcal{P}$ forwards ρ_i to P, which replies with polynomials $p_{i,1}, \ldots, p_{i,\mathsf{s}(i)} \in \mathbb{F}[X]$ with $\deg(p_{i,j}) \leq \mathsf{d}(|\mathsf{i}|, i, j)$.
3. $\mathcal{P}$ computes and outputs commitments as follows.
   - If $i = \mathsf{k}_{\mathsf{w}}$ (i.e. witness-committing round), then $\mathcal{P}$ first decomposes witness-carrying polynomials as
   $$(p^{\mathsf{com}}_{i,j}(X), p^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W} := \mathsf{Decomp}((p_{i,j}(X))_{(i,j)\in W}, I_{\mathsf{com}})$$
   such that $p_{i,j}(X) = p^{\mathsf{com}}_{i,j}(X) + p^{\mathsf{mid}}_{i,j}(X)$.
   - For every $(i, j) \in W$, $\mathcal{P}$ sends
   $$c^{\mathsf{com}}_{i,j} := \mathsf{PC.Com}_{\mathsf{ck}}(p^{\mathsf{com}}_{i,j}(X), \mathsf{d}(|\mathsf{i}|, i, j); \omega^{\mathsf{com}}_{i,j})$$
   $$c^{\mathsf{mid}}_{i,j} := \mathsf{PC.Com}_{\mathsf{ck}}(p^{\mathsf{mid}}_{i,j}(X), \mathsf{d}(|\mathsf{i}|, i, j); \omega^{\mathsf{mid}}_{i,j})$$
   to $\mathcal{V}$, where $\omega^{\mathsf{com}}_{i,j}$ and $\omega^{\mathsf{mid}}_{i,j}$ are uniformly sampled masking polynomials according the polynomial commitment scheme. $\mathcal{P}$ lets $\omega_{i,j} := \omega^{\mathsf{com}}_{i,j} + \omega^{\mathsf{mid}}_{i,j}$. $\mathcal{V}$ computes $c_{i,j} := c^{\mathsf{com}}_{i,j} + c^{\mathsf{mid}}_{i,j}$.
   - For every $(i, j) \notin W$, $\mathcal{P}$ sends
   $$c_{i,j} := \mathsf{PC.Com}_{\mathsf{ck}}(p_{i,j}(X), \mathsf{d}(|\mathsf{i}|, i, j); \omega_{i,j})$$
   to $\mathcal{V}$.

After k rounds of interaction, $\mathcal{V}$ obtains an additional challenge $\rho_{\mathsf{k}+1} \in \mathbb{F}^*$ from the AHP verifier V, used in the next phase. Let $\mathbf{c} := (c_{i,j})_{i\in[\mathsf{k}], j\in[\mathsf{s}(i)]}$, $\mathbf{p} := (p_{i,j})_{i\in[\mathsf{k}], j\in[\mathsf{s}(i)]}$, $\boldsymbol{\omega} := (\omega_{i,j})_{i\in[\mathsf{k}], j\in[\mathsf{s}(i)]}$ and $\mathbf{d} := (\mathsf{d}(|\mathsf{i}|, i, j))_{i\in[\mathsf{k}], j\in[\mathsf{s}(i)]}$.

Query phase.

1. $\mathcal{V}$ sends $\rho_{\mathsf{k}+1} \in \mathbb{F}^*$ that represents randomness for the query phase of $\mathsf{V}(\mathbb{F}, \mathsf{x})$ to $\mathcal{P}$.
2. $\mathcal{P}$ uses the query algorithm of V to compute the query set $Q := \mathsf{Q}_{\mathsf{V}}(\mathbb{F}, \mathsf{x}; \rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{k}+1})$.
3. $\mathcal{P}$ replies with answers $\mathbf{v} := \mathbf{p}(Q)$.
4. $\mathcal{V}$ samples and sends an opening challenge $\xi \in \mathbb{F}$ to $\mathcal{P}$.
5. $\mathcal{P}$ replies with an evaluation proof to demonstrate correctness of all claimed evaluations.
$$\pi_{\mathsf{Eval}} := \mathsf{PC.Open}_{\mathsf{ck}}(\mathbf{p}, \mathbf{d}, Q, \xi; \boldsymbol{\omega})$$

Linking phase. $\mathcal{P}$ invokes

$$\mathsf{CP}_{\mathtt{lnk}}.\mathcal{P}(((\hat{c}_k)_{k\in[\ell]}, \mathbf{v}, Q, (c^{\mathsf{com}}_{i,j}(X), c^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}), ((p^{\mathsf{com}}_{i,j}(X), p^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}, (\omega^{\mathsf{com}}_{i,j}(X), \omega^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}, (r_k)_{k\in[\ell]}))$$

to obtain and send linking proof $\pi_{\mathtt{lnk}}$.

Decision phase. $\mathcal{V}$ accepts if and only if the following conditions hold:

- the decision algorithm of V accepts the answers, i.e., $\mathsf{D}_{\mathsf{V}}(\mathbb{F}, \mathsf{x}, \mathbf{v}, \rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{k}+1}) = 1$;
- the alleged answers pass the test, i.e., $\mathsf{PC.Check}_{\mathsf{rk}}(\mathbf{c}, \mathbf{d}, Q, \mathbf{v}, \pi_{\mathsf{Eval}}, \xi) = 1$;
- the alleged linking proof is verified, i.e.,
$$\mathsf{CP}_{\mathtt{lnk}}.\mathcal{V}(((\hat{c}_k)_{k\in[\ell]}, \mathbf{v}, Q, (c^{\mathsf{com}}_{i,j}(X), c^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}), \pi_{\mathtt{lnk}}) = 1;$$

Fig. 1. Compiler from AHP to Interactive AoK for $\mathcal{R}_{\mathsf{com}}$. The differences with the Marlin compiler are marked in red.

Protocol AmComEq

1. $\mathcal{V}$ sends random challenge $x \in \mathbb{Z}_q$. Both parties compute
$$\tilde{\mathbf{G}} = [\mathbf{G}, \mathbf{G}^x, \ldots, \mathbf{G}^{x^{\ell-1}}] \in \mathbb{G}^{\ell d}.$$
2. $\mathcal{P}$ samples random $\mathbf{r} \in \mathbb{Z}_q^{\ell d}$, $\boldsymbol{\delta} \in \mathbb{Z}_q^{d'}$, $\boldsymbol{\gamma} \in \mathbb{Z}_q^{d''}$, and sends
$$A = \boldsymbol{g}^{\mathbf{r}}\mathbf{h}^{\boldsymbol{\delta}} \qquad \hat{A} = \tilde{\mathbf{G}}^{\mathbf{r}}\mathbf{H}^{\boldsymbol{\gamma}}$$
3. $\mathcal{V}$ sends random challenge $e \in \mathbb{Z}_q$.
4. $\mathcal{P}$ sends
$$\mathbf{z} = \mathbf{r} + e\mathbf{w}, \quad \boldsymbol{\omega} = \boldsymbol{\delta} + e\boldsymbol{\alpha}, \quad \boldsymbol{\Omega} = \boldsymbol{\gamma} + e\sum_{i=1}^{\ell} \boldsymbol{\beta}_i x^{i-1}$$
5. $\mathcal{V}$ checks
$$\boldsymbol{g}^{\mathbf{z}}\mathbf{h}^{\boldsymbol{\omega}} \stackrel{?}{=} AC^e, \quad \tilde{\mathbf{G}}^{\mathbf{z}}\mathbf{H}^{\boldsymbol{\Omega}} \stackrel{?}{=} \hat{A}\prod_{i=1}^{\ell}(\hat{C}_i^{x^{i-1}})^e$$

Fig. 2. Four-move protocol for amortized equality of many vector Pedersen commitments.

such that $C = \boldsymbol{g}^{\tilde{\mathbf{w}}_j}\mathbf{h}^{\tilde{\boldsymbol{\alpha}}_j}$. For some distinct execution paths, one may extract different witnesses. However, if there's any pair of such witnesses, one is able to find a non-trivial discrete-log relation for the vector $[\boldsymbol{g}, \mathbf{h}]$. Hence under the assumption stated in the theorem it is guaranteed that for every execution path $j \in [\ell]$ the same witness is extracted with overwhelming probability, i.e., $(\tilde{\mathbf{w}}, \tilde{\boldsymbol{\alpha}}) = (\tilde{\mathbf{w}}_1, \tilde{\boldsymbol{\alpha}}_1) = \ldots = (\tilde{\mathbf{w}}_\ell, \tilde{\boldsymbol{\alpha}}_\ell)$.

Now we show that each i-th slot of $\tilde{\mathbf{w}} = [\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_\ell]$ corresponds to what is committed in $\hat{C}_i$. First, we get a value in the form of $(\boldsymbol{\Omega}_j - \boldsymbol{\Omega}'_j)/(e_j - e'_j)$ from j-th execution path for $j \in [\ell]$. Thus we can extract $(\tilde{\boldsymbol{\beta}}_i)_{i\in[\ell]}$ such that $(\boldsymbol{\Omega}_j - \boldsymbol{\Omega}'_j)/(e_j - e'_j) = \sum_{i=1}^{\ell} \tilde{\boldsymbol{\beta}}_i x_j^{i-1}$ as these equations uniquely define a degree-ℓ polynomial $\beta(X) = \sum_{i=1}^{\ell} \tilde{\boldsymbol{\beta}}_i X^{i-1}$. From the second verification condition, we get in total ℓ equations of the form

$$\prod_{i=1}^{\ell} \hat{C}_i^{x_j^{i-1}} = \tilde{\mathbf{G}}^{(\mathbf{z}_j - \mathbf{z}'_j)/(e_j - e'_j)}\mathbf{H}^{(\boldsymbol{\Omega}_j - \boldsymbol{\Omega}'_j)/(e_j - e'_j)} = \mathbf{G}^{\sum_{i=1}^{\ell} \tilde{\mathbf{w}}_i x_j^{i-1}}\mathbf{H}^{\sum_{i=1}^{\ell} \tilde{\boldsymbol{\beta}}_i x_j^{i-1}}. \tag{2}$$

for every $j \in [\ell]$. Let us rewrite $\mathbf{G} = G^{s_1} \ldots G^{s_d}$, $\mathbf{H} = G^{t_1} \ldots G^{t_{d''}}$ and $\hat{C}_i = G^{u_i}$ using some arbitrary generator $G \in \mathbb{G}$. Then Eq. (2) can be rewritten as follows.

$$G^{\sum_{i=1}^{\ell} u_i x_j^{i-1}} = G^{\sum_{i=1}^{\ell} (s_1\tilde{\mathbf{w}}_{i,1} + \ldots + s_d\tilde{\mathbf{w}}_{i,d} + t_1\tilde{\boldsymbol{\beta}}_{i,1} + \ldots + t_{d''}\tilde{\boldsymbol{\beta}}_{i,d''})x_j^{i-1}}. \tag{3}$$

As x_j for $j \in [\ell]$ are distinct with each other, we have ℓ evaluations for the polynomial $u(X) = \sum_{i=1}^{\ell} u_i X^{i-1}$. Hence $u(X)$ can be uniquely determined as

$$u(X) = \sum_{i=1}^{\ell} (s_1\tilde{\mathbf{w}}_{i,1} + \ldots + s_d\tilde{\mathbf{w}}_{i,d} + t_1\tilde{\boldsymbol{\beta}}_{i,1} + \ldots + t_{d''}\tilde{\boldsymbol{\beta}}_{i,d''})X^{i-1} \mod q. \tag{4}$$

Recalling that $\hat{C}_i = G^{u_i}$, we get

$$\hat{C}_i = G^{(s_1\tilde{\mathbf{w}}_{i,1} + \ldots + s_d\tilde{\mathbf{w}}_{i,d} + t_1\tilde{\boldsymbol{\beta}}_{i,1} + \ldots + t_{d''}\tilde{\boldsymbol{\beta}}_{i,d''})} = \mathbf{G}^{\tilde{\mathbf{w}}_i}\mathbf{H}^{\tilde{\boldsymbol{\beta}}_i}. \tag{5}$$

Hence we conclude that every $\hat{C}_i$ indeed contains the witness $(\tilde{\mathbf{w}}_i, \tilde{\boldsymbol{\beta}}_i)$.

Special HVZK. Given challenge x and e, the simulator samples random $\mathbf{z} \in \mathbb{Z}_q^{\ell d}$, $\boldsymbol{\omega} \in \mathbb{Z}_q^{d'}$, $\boldsymbol{\Omega} \in \mathbb{Z}_q^{d''}$, and then the other messages can be perfectly simulated as follows.

$$A := \boldsymbol{g}^{\mathbf{z}}\mathbf{h}^{\boldsymbol{\omega}}C^{-e}, \quad \hat{A} := \tilde{\mathbf{G}}^{\mathbf{z}}\mathbf{H}^{\boldsymbol{\Omega}}\prod_{i=1}^{\ell}(\hat{C}_i^{x^{i-1}})^{-e}$$

Protocol CompDLEq

Let $\boldsymbol{g} = [\boldsymbol{g}_L, \boldsymbol{g}_R]$, $\mathbf{G} = [\mathbf{G}_L, \mathbf{G}_R]$, $\mathbf{z} = [\mathbf{z}_L, \mathbf{z}_R]$ where each sub-vector is of dimension $d/2$.

1. $\mathcal{P}$ sends shifted commitments

$$L = \boldsymbol{g}_R^{\mathbf{z}_L}, \quad R = \boldsymbol{g}_L^{\mathbf{z}_R},$$
$$\hat{L} = \mathbf{G}_R^{\mathbf{z}_L}, \quad \hat{R} = \mathbf{G}_L^{\mathbf{z}_R}$$

2. $\mathcal{V}$ sends random challenge $c \in \mathbb{Z}_q$.
3. $\mathcal{P}$ computes

$$\mathbf{z}' = \mathbf{z}_L + c\mathbf{z}_R$$

and both parties compute

$$Y' = LY^c R^{c^2}, \quad \hat{Y}' = \hat{L}\hat{Y}^c \hat{R}^{c^2}$$
$$\boldsymbol{g}' = \boldsymbol{g}_L^c \odot \boldsymbol{g}_R, \quad \mathbf{G}' = \mathbf{G}_L^c \odot \mathbf{G}_R.$$

If $d > 2$ then they invoke CompDLEq for the next instance

$$((\boldsymbol{g}', \mathbf{G}', d/2), (Y', \hat{Y}'), \mathbf{z}').$$

Otherwise, $\mathcal{P}$ sends $\mathbf{z}'$ and $\mathcal{V}$ checks that

$$\boldsymbol{g}'^{\mathbf{z}'} \stackrel{?}{=} Y', \quad \mathbf{G}'^{\mathbf{z}'} \stackrel{?}{=} \hat{Y}'.$$

Fig. 3. Compressed Σ-protocol for equality of vector discrete logs

4.2 CompAmComEq: Recursive compression

The major drawback of AmComEq is that its proof size is still linear in the vector dimension ℓd, due to the response vector $\mathbf{z} \in \mathbb{Z}_q^{\ell d}$. Notice however that once the rest of transcript $x, A, \hat{A}, e, \boldsymbol{\omega}, \boldsymbol{\Omega}$ is fixed, it should be sufficient to prove knowledge of $\mathbf{z}$ such that $\boldsymbol{g}^{\mathbf{z}} = Y := AC^e \mathbf{h}^{-\boldsymbol{\omega}}$ and $\tilde{\mathbf{G}}^{\mathbf{z}} = \hat{Y} := \hat{A} \prod_{i=1}^{\ell} (\hat{C}_i^{x^{i-1}})^e \mathbf{H}^{-\boldsymbol{\Omega}}$, instead of sending $\mathbf{z}$. This is where the *compressed Σ-protocol theory* [AC20, ACF20, ACR20, ACK21] comes into play. That is, the last move of AmComEq can invoke another protocol CompDLEq of proof size $O(\log(\ell d))$, for the relation

$$\mathcal{R}_{\mathsf{DLEq}} = \left\{ ((\boldsymbol{g}, \tilde{\mathbf{G}}, \ell d), (Y, \hat{Y}), \mathbf{z}) \ : \ Y = \boldsymbol{g}^{\mathbf{z}}, \hat{Y} = \tilde{\mathbf{G}}^{\mathbf{z}} \right\}. \tag{6}$$

The protocol CompDLEq for $\mathcal{R}_{\mathsf{DLEq}}$ is described in Fig. 3. From [AC20, Theorem 2] we immediately get the following result.

Theorem 3. CompDLEq *is a $(2\mu+1)$-move protocol for the relation $\mathcal{R}_{\mathsf{DLEq}}$, where $\mu = \lceil \log_2(\ell d) \rceil - 1$. It is perfectly complete and unconditionally $(k_1, \ldots, k_\mu)$-special sound, where $k_i = 3$ for all $i \in [1, \mu]$. Moreover, the communication costs are:*

- *$\mathcal{P} \to \mathcal{V}$: $4\lceil \log_2(\ell d) \rceil - 4$ elements of $\mathbb{G}$ and 2 elements of $\mathbb{Z}_q$.*
- *$\mathcal{V} \to \mathcal{P}$: $\lceil \log_2(\ell d) \rceil - 1$ elements of $\mathbb{Z}_q$.*

Let CompAmComEq be a protocol identical to AmComEq, except that its last move is replaced by CompDLEq. Then we obtain the following.

Corollary 1. CompAmComEq *is a $(2\mu+4)$-move protocol for the relation $\mathcal{R}_{\mathsf{AmComEq}}$, where $\mu = \lceil \log_2(\ell d) \rceil - 1$. It is perfectly complete and computationally $(\ell, 2, k_1, \ldots, k_\mu)$-special sound if finding non-trivial discrete-log relation for the generators $[\boldsymbol{g}, \mathbf{h}]$ is hard, where $k_i = 3$ for all $i \in [1, \mu]$. Moreover, the communication costs are:*

- *$\mathcal{P} \to \mathcal{V}$: $4\lceil \log_2(\ell d) \rceil - 2$ elements of $\mathbb{G}$ and $2 + d' + d''$ elements of $\mathbb{Z}_q$.*
- *$\mathcal{V} \to \mathcal{P}$: $\lceil \log_2(\ell d) \rceil + 1$ elements of $\mathbb{Z}_q$.*

Protocol $\mathsf{AHP}_{\mathsf{PLONK}}$

Offline phase. The indexer I receives as input $\mathbb{F} \in \mathcal{F}$ and $\mathsf{i} = (\mathbb{F}, n, m, l, \mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O, \mathbf{q}_M, \mathbf{q}_C, \sigma, \mathcal{T}_\mathcal{C})$, and computes the following polynomial oracles as described in the text: selector polynomials $(q_L, q_R, q_O, q_M, q_C)$; preprocessed polynomials for permutation argument $(S_{L,\mathsf{ID}}, S_{R,\mathsf{ID}}, S_{O,\mathsf{ID}}, S_{L,\sigma}, S_{R,\sigma}, S_{O,\sigma})$; vanishing polynomial of $\mathbb{H}$, $v_\mathbb{H}(X) = X^n - 1$.

Input. P receives $(\mathbb{F}, \mathsf{i}, (\mathsf{w}_i)_{i\in[l]}, (\mathsf{w}_i)_{i\in[l+1,3n]})$ and V receives $(\mathbb{F}, (\mathsf{w}_i)_{i\in[l]})$ and oracle access to the polynomials output by $\mathsf{I}(\mathbb{F}, \mathsf{i})$.

Online phase: first round. P computes $f_{\mathsf{pub}}(X), f_L(X), f_R(X), f_O(X)$ as described in Eq. (7) and sends $(f_L(X), f_R(X), f_O(X))$ to V.

Online phase: second round. Upon receiving challenges $\beta, \gamma \in \mathbb{F}$ from the V, P computes $h_{\mathsf{ID}}(X)$, $h_\sigma(X)$ and a permutation polynomial $s(X)$ as described in Eqs. (26) and (27). Then P sends an oracle polynomial $s(X)$ to V.

Online phase: third round. Upon receiving challenge $\alpha \in \mathbb{F}$ from the V, P computes

$$\begin{aligned} F_\mathcal{C}(X) &= q_L(X)f_L(X) + q_R(X)f_R(X) + q_O(X)f_O(X) \\ &\quad + q_M(X)f_L(X)f_R(X) + q_C(X) + f_{\mathsf{pub}}(X) \\ F_1(X) &= h_{\mathsf{ID}}(X)s(X) - h_\sigma(X)s(\zeta X) \\ F_2(X) &= L_1(X)(s(X) - 1) \\ T(X) &= \frac{F_\mathcal{C}(X) + F_1(X)\cdot\alpha + F_2(X)\cdot\alpha^2}{v_\mathbb{H}(X)} \end{aligned}$$

and sends an oracle polynomial $T(X)$ to V.

Query phase. V queries online oracles $(f_L(X), f_R(X), f_O(X), s(X), T(X))$ and all offline oracles with a random query point $z \in \mathbb{F}$. Moreover, it makes an additional query to the permutation polynomial $s(X)$ with ζz.

Decision phase. V first computes $f_{\mathsf{pub}}(X)$ as described in the text. Then V constructs $F_\mathcal{C}(z)$ (see (18)), $F_1(z)$ and $F_2(z)$ based on the outputs of polynomial oracles. It then checks that $(F_\mathcal{C}(z) + F_1(z)\cdot\alpha + F_2(z)\cdot\alpha^2) = T(z)\cdot v_\mathbb{H}(z)$.

Fig. 4. AHP for $\mathcal{R}'_{\mathsf{PLONK}}$

5 Instantiation with PLONK

In this section we apply our ECLIPSE compiler to PLONK. We first go over the essential part of the PLONK protocol, using the language of AHP. More detailed preliminaries are provided in Appendix B.

5.1 PLONK AHP

We consider a arithmetic circuit with fan-in two over $\mathbb{F}$, consisting of n gates. The PLONK AHP essentially proves knowledge of left, right and output wire values for every gate $i \in [n]$ in the circuit, such that they are also consistent with the constraints determined by the circuit topology. The per-gate constraints are specified by *selector vectors* $\mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O, \mathbf{q}_M, \mathbf{q}_C \in \mathbb{F}^n$. We call $\mathcal{C} = (n, m, \mathbf{L}, \mathbf{R}, \mathbf{O}, \mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O, \mathbf{q}_M, \mathbf{q}_C)$ *constraint systems*.

$\mathsf{AHP}_{\mathsf{PLONK}}$ relies on a multiplicative subgroup $\mathbb{H} = \{\zeta, \zeta^2, \ldots, \zeta^n\} \subset \mathbb{F}^*$ generated by an nth primitive root of unity $\zeta \in \mathbb{F}^*$. It follows that an associated vanishing polynomial $v_\mathbb{H}(X) = X^n - 1$ splits completely in $\mathbb{F}[X]$, i.e., $X^n - 1 = \prod_{i=1}^n (X - \zeta^i)$. Then we have the corresponding Lagrange basis $L_i(X) \in \mathbb{F}_{<n}[X]$ for $i \in [n]$ such that $L_i(\zeta^i) = 1$ and $L_i(\zeta^j) = 1$ for $j \neq i$.

During the first round of $\mathsf{AHP}_{\mathsf{PLONK}}$ (Fig. 4), the prover sends the following polynomials encoding both statement and witness $((\mathsf{w}_i)_{i\in[l]}, (\mathsf{w}_i)_{i\in[l+1,3n]})$:

$$f_L(X) = \sum_{i\in[n]} \mathsf{w}_i L_i(X) \qquad f_R(X) = \sum_{i\in[n]} \mathsf{w}_{n+i} L_i(X) \qquad f_O(X) = \sum_{i\in[n]} \mathsf{w}_{2n+i} L_i(X) \tag{7}$$

To achieve zero-knowledge these polynomials are masked by blinding terms $(b_{L,1}X + b_{L,2})v_\mathbb{H}(X)$, $(b_{R,1}X + b_{R,2})v_\mathbb{H}(X)$ and $(b_{O,1}X + b_{O,2})v_\mathbb{H}(X)$ where each coefficient is randomly sampled by the AHP prover.

5.2 CP-PLONK

Our goal is to turn $\mathsf{AHP}_{\mathsf{PLONK}}$ into CP-PLONK with our compiler. We first describe a commit-and-prove variant of relation $\mathcal{R}'_{\mathsf{PLONK}}$. We assume without loss of generality that every committed witness $(\mathsf{w}_i)_{i\in I_{\mathsf{com}}}$ is left input to gate i. Then we use the following disjoint witness index sets: $I_{\mathsf{pub}} = [l], I_{\mathsf{com}} =$

$[l+1, l+\ell d], I_{\mathsf{mid}} = [l+\ell d+1, n]$, assuming that $\mathsf{w}_{l+1}, \ldots, \mathsf{w}_{l+\ell d}$ are ℓd witness values committed in advance. Moreover, suppose every vector compound of d values $(\mathsf{w}_i)_{i \in I_k}$, where $I_k = [l+1+d(k-1), l+dk]$, is committed into kth auxiliary commitment $\hat{C}_k$ for $k \in [\ell]$. Then we have $I_{\mathsf{com}} = \bigcup_{k \in [\ell]} I_k$.

Definition 11 (CP-PLONK indexed relation). *The indexed relation $\mathcal{R}_{\mathsf{CP\text{-}PLONK}}$ is the set of all triples*

$$((\mathbb{F}, n, m, l, \mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O, \mathbf{q}_M, \mathbf{q}_C, \sigma, \mathcal{T}_{\mathcal{C}}, I_{\mathsf{com}}, (I_k)_{k \in [\ell]}, \mathsf{ack}), ((\mathsf{w}_i)_{i \in [l]}, (\hat{C}_k)_{k \in [\ell]}), ((\mathsf{w}_i)_{i \in [l+1,3n]}, (r_k)_{k \in [\ell]}))$$

such that

$$\begin{aligned}
\forall i \in [n]: &\quad \mathsf{w}_i = \mathsf{w}_{\sigma(i)} \\
\forall i \in [l]: &\quad (\mathbf{q}_L)_i \cdot \mathsf{w}_i + (\mathbf{q}_R)_i \cdot \mathsf{w}_{n+i} + (\mathbf{q}_O)_i \cdot \mathsf{w}_{2n+i} + (\mathbf{q}_M)_i \mathsf{w}_i \mathsf{w}_{n+i} + (\mathbf{q}_C)_i - \mathsf{w}_i = 0 \\
\forall i \in [l+1, n]: &\quad (\mathbf{q}_L)_i \cdot \mathsf{w}_i + (\mathbf{q}_R)_i \cdot \mathsf{w}_{n+i} + (\mathbf{q}_O)_i \cdot \mathsf{w}_{2n+i} + (\mathbf{q}_M)_i \mathsf{w}_i \mathsf{w}_{n+i} + (\mathbf{q}_C)_i = 0 \\
\forall k \in [\ell]: &\quad \hat{c}_k = \mathsf{AC.Com}_{\mathsf{ack}}((\mathsf{w}_i)_{i \in I_k}; r_k)
\end{aligned}$$

5.2.1 Applying our compiler We show that $\mathsf{AHP}_{\mathsf{PLONK}}$ as well as the polynomial commitment scheme meets the requirements of Theorem 1.

- Decomp takes $\mathsf{n_w} = 3$ (blinded) witness-carrying polynomials (f_L, f_R, f_O) and $I_{\mathsf{com}} \subset [n]$, parses f_L as $\sum_{i \in [n]} \mathsf{w}_i L_i(X) + (b_1 X + b_2) v_{\mathbb{H}}(X)$, and decompose them as follows.

$$\begin{aligned}
f_{L,\mathsf{com}}(X) &:= \sum_{i \in I_{\mathsf{com}}} \mathsf{w}_i L_i(X) + (\rho_1 X + \rho_2) v_{\mathbb{H}}(X) \qquad f_{R,\mathsf{com}}(X) := 0 \qquad f_{O,\mathsf{com}}(X) := 0 \\
f_{L,\mathsf{mid}}(X) &:= \sum_{i \in [n] \setminus I_{\mathsf{com}}} \mathsf{w}_i L_i(X) + (\lambda_1 X + \lambda_2) v_{\mathbb{H}}(X) \quad f_{R,\mathsf{mid}}(X) := f_R(X) \quad f_{O,\mathsf{mid}}(X) := f_O(X)
\end{aligned}$$

  where ρ_i's are randomly chosen and $\lambda_i := b_i - \rho_i$. Clearly, the decomposition is additive and degree-preserving.

- WitExt takes witness-carrying polynomials (f_L, f_R, f_O) and uniquely extracts witness vectors for every $i \in [n]$

$$\mathsf{w}_i = f_L(\zeta^i) \qquad \mathsf{w}_{n+i} = f_R(\zeta^i) \qquad \mathsf{w}_{2n+i} = f_O(\zeta^i)$$

  As it's independently extracting witness values within disjoint index sets $I_L = [n]$, $I_R = [n+1, 2n]$, and $I_O = [2n+1, 3n]$, respectively, we have that f_L, f_R and f_O are disjoint (see Definition 7).

- As PLONK retains zero-knowledge by blinding witness-carrying polynomials, but without hiding commitment, we use de-randomized version of $\mathsf{PC_{KZG}.Com_{ck}}$ (see Sect. 2.5.1) that takes polynomial $f \in \mathbb{F}_{<D}[X]$ and outputs $[f(\chi)]_1$. Clearly, this is an additively homomorphic commitment scheme. Its binding and extractability were formally shown in Appendix B-D of [CHM+20]. As mentioned in [GWC19] and from how WitExt works, the knowledge soundness of PLONK holds only by enforcing degree bound to the maximum degree D for committed polynomials so the plain KZG construction should suffice for compiling $\mathsf{AHP}_{\mathsf{PLONK}}$.

We now define a suitable commitment-linking protocol $\mathsf{CP}_{\mathtt{lnk}}$ in Fig. 5. Since witness-carrying polynomials are disjoint it is enough to provide linking w.r.t a polynomial f_L. The main idea is to (1) prove consistency between $f_{L,\mathsf{com}}$ and auxiliary commitments $\hat{C}_k$ with the $\mathsf{AmComEq}$ protocol from previous section, and (2) force the prover to show f_{mid} vanishes at all points in $\mathbb{H}_{\mathsf{com}} = \{\zeta^i\}_{i \in I_{\mathsf{com}}}$. The latter is in particular crucial for WitExt to successfully output a witness vector consistent with auxiliary commitments, even after taking the sum of $f_{L,\mathsf{com}}$ and $f_{L,\mathsf{mid}}$. This step only incurs constant overhead in the evaluation proof thanks to the batch evaluation technique proposed in [BDFG20]. On the other hand, the consistency between f_{com} and ℓ vector Pedersen commitments $\hat{C}_k = \mathbf{G}^{(\mathsf{w}_i)_{i \in I_k}} H^{r_k}$ for $k \in [\ell]$ are handled by $\mathsf{CompAmComEq}$ protocol (see Sect. 4).

Lemma 1. *Assuming extractability of $\mathsf{PC}_{\mathsf{KZG}}$ and argument of knowledge of $\mathsf{CompAmComEq}$, the protocol $\mathsf{CP}_{\mathtt{lnk}}$ (Fig. 5) is an argument of knowledge.*

Proof. First, the extractor $\mathcal{E}_{\mathtt{lnk}}$ obtains $f_{L,\mathsf{mid}}(X) \in \mathbb{F}_{<D}[X]$ such that $[f_{L,\mathsf{mid}}(\chi)]_1 = C_{L,\mathsf{mid}}$ and $f_{L,\mathsf{mid}}(\zeta^i) = 0$ for $i \in I_{\mathsf{com}}$, by internally invoking an extractor $\mathcal{E}_{\mathsf{KZG}}$, which succeeds with overwhelming probability as long as a malicous prover $\mathcal{P}_{\mathtt{lnk}}$ convinces the verifier.

Second, $\mathcal{E}_{\mathtt{lnk}}$ invokes an extractor $\mathcal{E}_{\mathsf{ComEq}}$ for the $\mathsf{CompAmComEq}$ protocol, which outputs $(\mathsf{w}_i)_{i\in I_{\mathsf{com}}}$ and $(r_k)_{k\in[\ell]}$ such that $\hat{C}_k = \mathsf{AC.Com}_{\mathsf{ack}}((\mathsf{w}_i)_{i\in I_k}; r_k)$ for $k \in [\ell]$, and $C_{L,\mathsf{com}} = [\sum_{i\in I_{\mathsf{com}}} \mathsf{w}_i L_i(\chi) + (\rho_1\chi + \rho_2)v_{\mathbb{H}}(\chi)]_1$. So we obtain $f_{L,\mathsf{com}}(X) = \sum_{i\in I_{\mathsf{com}}} \mathsf{w}_i L_i(X) + (\rho_1 X + \rho_2)v_{\mathbb{H}}(X)$.

Let $f_L(X) := f_{L,\mathsf{com}}(X) + f_{L,\mathsf{mid}}(X)$. Due to the 0-evaluation proof output by $\mathcal{P}_{\mathtt{lnk}}$, it holds that $f_L(X)$ and $f_{L,\mathsf{com}}(X)$ agree on $\mathbb{H}_{\mathsf{com}}$, i.e., $f_L(\zeta^i) = f_{L,\mathsf{com}}(\zeta^i) + f_{L,\mathsf{mid}}(\zeta^i) = f_{L,\mathsf{com}}(\zeta^i) = \mathsf{w}_i$ for each $i \in I_{\mathsf{com}}$ (recall that the term $(\rho_1 X + \rho_2)v_{\mathbb{H}}(X)$ vanishes anyway). Hence if WitExt is invoked on f_L it does extract witness $(\mathsf{w}_i)_{i\in I_{\mathsf{com}}}$ consistent with $(\hat{C}_k)_{k\in[\ell]}$, which is guaranteed by $\mathcal{E}_{\mathsf{ComEq}}$.

Lemma 2. *Assuming zero knowledge of Fiat–Shamir-transformed* $\mathsf{CompAmComEq}$*, the protocol* $\mathsf{CP}_{\mathtt{lnk}}$ *is zero-knowledge in the SRS model.*

Proof. To simulate π_{ComEq} we simply invoke the zero-knowledge simulator for $\mathsf{CompAmComEq}$ made non-interactive with Fiat–Shamir [FS87]. To simulate the evaluation proof Π the simulator uses the trapdoor χ used for generating the commitment key to compute $\Pi := C_{L,\mathsf{mid}}^{1/v_{\mathbb{H}_{\mathsf{com}}}(\chi)}$.

Protocol $\mathsf{CP}_{\mathtt{lnk}}$ for PLONK

Preprocessing. Both parties receive $\mathsf{srs}, I_{\mathsf{com}}$ and precompute $[v_{\mathbb{H}_{\mathsf{com}}}(\chi)]_2$ such that

$$v_{\mathbb{H}_{\mathsf{com}}}(X) = \prod_{a\in\mathbb{H}_{\mathsf{com}}} (X - a), \quad \mathbb{H}_{\mathsf{com}} = \{\zeta^i \,:\, i \in I_{\mathsf{com}}\} \subset \mathbb{H}$$

Input. Both $\mathcal{P}_{\mathtt{lnk}}$ and $\mathcal{V}_{\mathtt{lnk}}$ receives $(\mathsf{ck}, \mathsf{ack}, (\hat{C}_k)_{k\in[\ell]}, (C_{L,\mathsf{com}}, C_{L,\mathsf{mid}}))$ as statements. The $\mathcal{P}_{\mathtt{lnk}}$ has as input witness $(f_{L,\mathsf{com}}(X), f_{L,\mathsf{mid}}(X), (r_k)_{k\in[\ell]})$ such that

$$f_{L,\mathsf{com}}(X) = \sum_{i\in I_{\mathsf{com}}} \mathsf{w}_i L_i(X) + (\rho_1 X + \rho_2)v_{\mathbb{H}}(X) \qquad f_{L,\mathsf{mid}}(X) = \sum_{i\in[n]\setminus I_{\mathsf{com}}} \mathsf{w}_i L_i(X) + (\lambda_1 X + \lambda_2)v_{\mathbb{H}}(X)$$

$$\hat{C}_k = \mathbf{G}^{(\mathsf{w}_i)_{i\in I_k}} H^{r_k} \qquad C_{L,\mathsf{com}} = [f_{L,\mathsf{com}}(\chi)]_1 \qquad C_{L,\mathsf{mid}} = [f_{L,\mathsf{mid}}(\chi)]_1$$

Prove.

- Compute a proof $\pi_{\mathsf{CompAmComEq}}$ of the following statement where $g_i := [L_i(\chi)]_1$ for $i \in I_{\mathsf{com}}$, $\boldsymbol{g} := (g_i)_{i\in I_{\mathsf{com}}}$, $h_1 = [\chi v_{\mathbb{H}}(\chi)]_1$, $h_2 = [v_{\mathbb{H}}(\chi)]_1$.

$$\mathsf{CompAmComEq} : \mathsf{PK}\{((\mathsf{w}_i)_{i\in I_{\mathsf{com}}}, (r_k)_{k\in[\ell]}, \rho_1, \rho_2) \,:\, \hat{C}_k = \mathbf{G}^{(\mathsf{w}_i)_{i\in I_k}} H^{r_k} \wedge C_{L,\mathsf{com}} = \boldsymbol{g}^{(\mathsf{w}_i)_{i\in I_{\mathsf{com}}}} h_1^{\rho_1} h_2^{\rho_2}\}$$

- Compute evaluation proof $W(X) = \frac{f_{L,\mathsf{mid}}(X)}{v_{\mathbb{H}_{\mathsf{com}}}(X)}$ and $\Pi := [W(\chi)]_1$. Output $\pi_{\mathtt{lnk}} = (\Pi, \pi_{\mathsf{CompAmComEq}})$.

Verify. Given $\pi_{\mathtt{lnk}}$, verify $\pi_{\mathsf{CompAmComEq}}$ and check that $f_{L,\mathsf{mid}}$ vanishes on $\mathbb{H}_{\mathsf{com}}$:

$$e(C_{L,\mathsf{mid}}, h) = e(\Pi, [v_{\mathbb{H}_{\mathsf{com}}}(\chi)]_2)$$

Fig. 5. Commitment-linking protocol for PLONK

6 Instantiation with Marlin

Preliminaries. For a finite field $\mathbb{F}$ and a subset $\mathbb{S} \subseteq \mathbb{F}$, we denote by $v_{\mathbb{S}}(X)$ the vanishing polynomial of $\mathbb{S}$ that is the unique non-zero monic polynomial of degree at most $|\mathbb{S}|$ that is zero everywhere on $\mathbb{S}$. For a matrix $M \in \mathbb{F}^{n\times n}$ we denote the number of its nonzero entries by $\|M\|$. For two vectors u and v, we denote by $u \circ v$ their Hadamard (component-wise) product. For a function $f : \mathbb{S} \to \mathbb{F}$, we denote by $\hat{f}$, the univariate polynomial over $\mathbb{F}$ with degree less than $|\mathbb{S}|$ that agrees with f, that is, $\hat{f}(k) = f(k)$ for all $k \in \mathbb{S}$. For an $n \times n$ matrix M with rows/columns indexed by elements of $\mathbb{S}$, we denote by $\hat{M}(X, Y)$, the polynomial of individual degree less than n such that $\hat{M}(s, t)$ is the (s, t)th entry of M for all $s, t \in \mathbb{S}$.

Define the bivariate polynomial $u_{\mathbb{S}}(X, Y)$

$$u_{\mathbb{S}}(X, Y) := \frac{v_{\mathbb{S}}(X) - v_{\mathbb{S}}(Y)}{X - Y}$$

such that $u_{\mathbb{S}}(X, X) = |\mathbb{S}|X^{|\mathbb{S}|-1}$ is the formal derivative of the vanishing polynomial $v_{\mathbb{S}}(X)$. We have that $u_{\mathbb{S}}(X, Y)$ vanishes on the square $\mathbb{S} \times \mathbb{S}$, except on the diagonal. It takes $u_{\mathbb{S}}(a, a)_{a\in\mathbb{S}}$ on the diagonal.

Univariate sumcheck [BCR+19]. Given a multiplicative subgroup $\mathbb{S}$ of $\mathbb{F}$, a polynomial $f(X)$ sums to σ over $\mathbb{S}$ if and only if $f(X)$ can be written as $h(X)v_{\mathbb{S}}(X) + Xg(X) + \sigma/|\mathbb{S}|$ for some $h(X)$ and $g(X)$ where the degree of $\deg(g) < |\mathbb{S}| - 1$.

Definition 12 (R1CS indexed relation). *R1CS (Rank-1 constraint satisfiability) indexed relation is the set of tuples* $(\mathsf{i}, \mathsf{x}, \mathsf{w}) = \big((\mathbb{F}, l, n, m, A, B, C), x, w\big)$ *for* $l, n, m \in \mathbb{N}, l \leq n, A, B, C \in \mathbb{F}^{n \times n}, x \in \mathbb{F}^l, w \in \mathbb{F}^{n-l}$, $m \geq \max\{\|A\|, \|B\|, \|C\|\}$ $z \coloneqq (x, w)$ *is a vector in* $\mathbb{F}^n$ *such that* $Az \circ Bz = Cz$.

We assume efficiently computable bijections $\phi_{\mathbb{H}} : \mathbb{H} \to [|\mathbb{H}|]$ and $\phi_{\mathbb{K}} : \mathbb{K} \to [|\mathbb{K}|]$, and denote the first k elements in H and the remaining elements, via sets $\mathbb{H}[\leq k] \coloneqq \{h \in \mathbb{H} : 1 \leq \phi_{\mathbb{H}}(h) \leq k\}$ and $\mathbb{H}[> k] \coloneqq \{h \in \mathbb{H} : k < \phi_{\mathbb{H}}(h) \leq |\mathbb{H}|\}$ respectively. We then denote the first part of the vector $z \in \mathbb{F}^n$ as the public component $x \in \mathbb{F}^l$ and the second part as witness component $w \in \mathbb{F}^{n-l}$.

6.1 Marlin AHP

We now describe the Marlin AHP. In the preprocessing phase, the indexer I is given as input a field $\mathbb{F}$, subsets $\mathbb{H}, \mathbb{K}$ of $\mathbb{F}$, and matrices $A, B, C \in \mathbb{F}^{n \times n}$ representing the R1CS instance. The output of the preprocessing phase is three univariate polynomials $\{\hat{\mathsf{row}}_M, \hat{\mathsf{col}}_M, \hat{\mathsf{val}}_M\}$ of degree less than $|\mathbb{K}|$ for each matrix $M \in A, B, C$, such that the following polynomial is a low-degree extension of M.

$$\hat{M}(X, Y) \coloneqq \sum_{k \in K} u_{\mathbb{H}}(X, \hat{\mathsf{row}}_M(k)) u_{\mathbb{H}}(Y, \hat{\mathsf{col}}_M(k)) \hat{\mathsf{val}}_M(k)$$

The three polynomials $\hat{\mathsf{row}}_M, \hat{\mathsf{col}}_M, \hat{\mathsf{val}}_M$ are the unique low-degree extensions of the functions $\mathsf{row}_M, \mathsf{col}_M, \mathsf{val}_M : K \to \mathbb{F}$ that denote the row index, column index and value of the non-zero entries of the matrix M respectively. Let $\hat{M}(X, Y)$ be the unique low-degree extension of M that agrees with the matrix M everywhere on the domain $\mathbb{H} \times \mathbb{H}$. The prover P receives as input the instance $x \in \mathbb{F}^l$, a witness $w \in \mathbb{F}^{n-l}$. The verifier V receives as input x, and obtains oracle access to the nine polynomials output at the end of the preprocessing phase.

Let $\hat{x}(X)$ and and $\hat{w}(X)$ be polynomials of degree at most l and $n - l$ that agree with the instance x on $\mathbb{H}[\leq l]$, and with the shifted witness on $\mathbb{H}[> l]$ respectively, where the shifted witness $\bar{w}$ is such that $\bar{w} : \mathbb{H}[> l] \to \mathbb{F}$,

$$\forall \gamma,\ \bar{w}(\gamma) \coloneqq \frac{w(\gamma) - \hat{x}(\gamma)}{v_{\mathbb{H}[\leq l]}(\gamma)}$$

Let $z \coloneqq (x, w)$ denote the full assignment. The prover computes the linear combinations $z_A \coloneqq Az, z_B \coloneqq Bz, z_C \coloneqq Cz$, and sets polynomials $\hat{z}_A(X), \hat{z}_B(X), \hat{z}_C(X) \in \mathbb{F}^{|\mathbb{H}|}(X)$.

P needs to prove that z_A, z_B, z_C are obtained as the specified linear combinations of z, and that $z_A \circ z_B = z_C$. Note that the polynomial $\hat{z}(X) \coloneqq \hat{w}(X) v_{\mathbb{H}[\leq l]}(X) + \hat{x}(X)$ agrees with z on $\mathbb{H}$. P sends the polynomial $h_0(X)$ such that $\hat{z}_A(X)\hat{z}_B(X) - \hat{z}_C(X) = h_0(X) v_{\mathbb{H}}(X)$, a random $s(X) \in \mathbb{F}$ together with its sum over $\mathbb{H}$, $\sigma_1 \coloneqq \sum_{k \in \mathbb{H}} s(k)$. V samples $\alpha, \eta_A, \eta_B, \eta_C$ randomly from $\mathbb{F}$ and send them to the prover. P and V engage in a univariate sumcheck protocol to prove that the polynomial $q_1(X)$ defined below sums to σ_1 on H.

$$q_1(X) \coloneqq s(X) + u_{\mathbb{H}}(\alpha, X) \Bigg(\sum_{M \in \{A,B,C\}} \eta_M \hat{z}_M(X) \Bigg) - \Bigg(\sum_{M \in \{A,B,C\}} \eta_M r_M(\alpha, X) \Bigg) \hat{z}(X)$$

where $r_M(X, Y) \coloneqq \sum_{k \in \mathbb{H}} u_{\mathbb{H}}(X, k) \hat{M}(k, Y)$. This is done via three sequential sumchecks. The AHP is given in Fig. 6.

Protocol Protocol $\mathsf{AHP}_{\mathsf{Marlin}}$

Offline phase. The indexer I is given as input a field $\mathbb{F} \in \mathcal{F}$, subsets $\mathbb{H}, \mathbb{K}$ of $\mathbb{F}$, and matrices $A, B, C \in \mathbb{F}^{n \times n}$ representing the R1CS instance, and outputs three univariate polynomial oracles $\{\hat{\mathsf{row}}_M, \hat{\mathsf{col}}_M, \hat{\mathsf{val}}_M\}$ of degree less than $|\mathbb{K}|$ for each matrix $M \in A, B, C$, such that the following polynomial is a low-degree extension of M.

$$\hat{M}(X, Y) \coloneqq \sum_{k \in \mathbb{K}} u_{\mathbb{H}}(X, \hat{\mathsf{row}}_M(k)) u_{\mathbb{H}}(Y, \hat{\mathsf{col}}_M(k)) \hat{\mathsf{val}}_M(k)$$

Input. P receives $(\mathbb{F}, \mathbb{H}, \mathbb{K}, A, B, C, \mathsf{i}, (z_i)_{i \in [l]}, (z_i)_{i \in [l+1, n]})$, and V receives $(\mathbb{F}, \mathbb{H}, \mathbb{K}, (z_i)_{i \in [l]})$ and oracle access to the nine polynomials output by $\mathsf{I}(\mathbb{F}, \mathsf{i})$.

Online phase: first round. P sends the oracle polynomials $\hat{w}(X) \in \mathbb{F}^{<n-l}(X), h_0(X), \hat{z}_A(X), \hat{z}_B(X), \hat{z}_C(X) \in \mathbb{F}^n(X)$. It samples a random $s(X) \in \mathbb{F}^{2n}(X)$ and sends polynomial oracle $s(X)$ together with $\sigma_1 \in \mathbb{F}$ where $\sigma_1 \coloneqq \sum_{k \in \mathbb{H}} s(k)$, and $\hat{z}_A(X)\hat{z}_B(X) - \hat{z}_C(X) = h_0(X) v_{\mathbb{H}}(X)$.

Online phase: second round. Upon receiving challenges $\alpha, \eta_A, \eta_B, \eta_C \in \mathbb{F}$ from V, P sends oracle polynomials $g_1(X), h_1(X) \in \mathbb{F}^n(X)$ to V, where

$$s(X) + u_{\mathbb{H}}(\alpha, X)\Big(\sum_{M \in \{A,B,C\}} \eta_M \hat{z}_M(X)\Big) - \Big(\sum_{M \in \{A,B,C\}} \eta_M r_M(\alpha, X)\Big)\hat{z}(X) = h_1(X) v_{\mathbb{H}}(X) + X g_1(X) + \sigma_1/|\mathbb{H}|$$

Online phase: third round. Upon receiving challenge $\beta_1 \in \mathbb{F}$ from the V, P sends oracle polynomials $g_2(X), h_2(X) \in \mathbb{F}^n(X)$ and $\sigma_2 \in \mathbb{F}$ to V, where $\sigma_2 \coloneqq \sum_{k \in \mathbb{H}} u_{\mathbb{H}}(\alpha, k) \sum_{M \in \{A,B,C\}} \eta_M \hat{M}(k, \beta_1)$,

$$u_{\mathbb{H}}(\alpha, X) \sum_{M \in \{A,B,C\}} \eta_M \hat{M}(X, \beta_1) = h_2(X) v_{\mathbb{H}}(X) + X g_2(X) + \sigma_2/|\mathbb{H}|$$

Online phase: fourth round. Upon receiving challenge $\beta_2 \in \mathbb{F}$ from the V, P sends oracle polynomials $g_3(X), h_3(X) \in \mathbb{F}^n(X)$ and $\sigma_3 \in \mathbb{F}$ to V, where where $\sigma_3 \coloneqq \sum_{k \in \mathbb{K}} \sum_{M \in \{A,B,C\}} \eta_M \frac{v_{\mathbb{H}}(\beta_2) v_{\mathbb{H}}(\beta_1) \hat{\mathsf{val}}_M(k)}{(\beta_2 - \hat{\mathsf{row}}_M(k))(\beta_1 - \hat{\mathsf{col}}_M(k))}$,

$$h_3(X) v_{\mathbb{K}}(X) = a(X) - b(X)(X g_3(X) + \sigma_3/|\mathbb{K}|)$$

$$a(X) = \sum_{M \in \{A,B,C\}} \eta_M v_{\mathbb{H}}(\beta_2) v_{\mathbb{H}}(\beta_1) \hat{\mathsf{val}}_M(X) \prod_{L \in \{A,B,C\} \setminus \{M\}} (\beta_2 - \hat{\mathsf{row}}_L(X))(\beta_1 - \hat{\mathsf{col}}_L(X))$$

$$b(X) = \prod_{M \in \{A,B,C\}} (\beta_2 - \hat{\mathsf{row}}_M(X))(\beta_1 - \hat{\mathsf{col}}_M(X))$$

Query phase. V queries the oracles $\hat{w}(X), \hat{z}_A(X), \hat{z}_B(X), \hat{z}_C(X), h_0(X), s(X), h_1(X), g_1(X)$ at β_1; $h_2(X), g_2(X)$ at β_2; $h_3(X), g_3(X)$ and all offline oracles $\{\hat{\mathsf{row}}_M, \hat{\mathsf{col}}_M, \hat{\mathsf{val}}_M\}$ for each $M \in A, B, C$ at a random query point $\beta_3 \in \mathbb{F}$.

Decision phase. V accepts if the following tests pass:

- $h_3(\beta_3) v_{\mathbb{K}}(\beta_3) = a(\beta_3) - b(\beta_3)(\beta_3 g_3(\beta_3) + \sigma_3/|\mathbb{K}|)$
- $h_2(\beta_2) v_{\mathbb{H}}(\beta_2) + \beta_2 g_2(\beta_2) + \sigma_2/|\mathbb{H}| = u_{\mathbb{H}}(\alpha, \beta_2)\sigma_3$
- $s(\beta_1) + u_{\mathbb{H}}(\alpha, \beta_1)(\sum_M \eta_M \hat{z}_M(\beta_1)) - \sigma_2 \hat{z}(\beta_1) = h_1(\beta_1) v_{\mathbb{H}}(\beta_1) + \beta_1 g_1(\beta_1) + \sigma_1/|\mathbb{H}|$
- $\hat{z}_A(\beta_1)\hat{z}_B(\beta_1) - \hat{z}_C(\beta_1) = h_0(\beta_1) v_{\mathbb{H}}(\beta_1)$

Fig. 6. AHP for $\mathcal{R}_{R1CS}$

6.2 CP-Marlin

Protocol $\mathsf{CP}_{\mathtt{lnk}}$ for Marlin

The common inputs are srs.

Preprocessing. Both parties receive $\mathsf{srs}, I_{\mathsf{com}}$ and precompute $[v_{\mathbb{H}[\leq l+d\ell]}(\chi)]_2$ such that

$$v_{\mathbb{H}[\leq l+d\ell]}(X) = \prod_{a \in \mathbb{H}_{\mathsf{com}}} (X - a), \quad \mathbb{H}_{\mathsf{com}} = \{\mathbb{H}[i] : i \in I_{\mathsf{com}} \cup I_{\mathsf{pub}}\} \subset \mathbb{H}$$

Input. Both $\mathcal{P}_{\mathtt{lnk}}$ and $\mathcal{V}_{\mathtt{lnk}}$ receive $(\mathsf{x}, \mathsf{ck}, (\hat{C}_k)_{k\in[\ell]}, (I_k)_{k\in[\ell]}[w_{\mathsf{com}}(\chi)]_1, [w_{\mathsf{mid}}(\chi)]_1)$ as common inputs. $\mathcal{P}_{\mathtt{lnk}}$ has as input witness vector $\mathbf{w}$ and commitment randomness r_k such that for shifted witness $\bar{\mathbf{w}} : \mathbb{H}[> l] \to \mathbb{F}$ defined as in the AHP, $w_{\mathsf{com}}(X) \in \mathbb{F}^{d\ell}[X]$ is the polynomial of degree less than $d\ell$ that agrees with $\bar{\mathbf{w}}$ on $\mathbb{H}[> l, \leq l + d\ell]$, and $w_{\mathsf{mid}}(X) \in \mathbb{F}^{n-l-d\ell}[X]$ is the polynomial of degree less than $m - l - d\ell$ that agrees with $\bar{\mathbf{w}}$ on $\mathbb{H}[> l + d\ell]$, and $\hat{C}_k = \mathbf{G}^{(w_i)_{i\in I_k}} H^{r_k}$.

Prove.

- Compute a proof $\pi_{\mathsf{CompAmComEq}}$ of the following statement,

$$\mathsf{PK}\left\{((w)_{i\in I_{\mathsf{com}}}, (r_k)_{k\in[\ell]}) \ : \ \hat{C}_k = \mathbf{G}^{(w_i)_{i\in I_k}} H^{r_k} \text{ for } k \in [\ell] \wedge [w_{\mathsf{com}}(\chi)]_1 = \textstyle\prod_{i\in I_{\mathsf{com}}} g_i^{w_i}\right\}$$

  where $g_i = [L_i]_1$, L_i are the Lagrange polynomials $\{L_{\mathbb{H}[\leq l+d\ell],\alpha}(X)\}_{\alpha\in\mathbb{H}[\leq l+d\ell]}$
- Compute a proof to show that $w_{\mathsf{mid}}(X)$ vanishes at $\mathbb{H}[\leq l + d\ell]$. Set

$$\mathit{\Pi} := [W(\chi)]_1$$

  where $W(X) = \frac{w_{\mathsf{mid}}(X)}{v_{\mathbb{H}[\leq l+d\ell]}}$. Output $\pi_{\mathsf{link}} = (\mathit{\Pi}, \pi_{\mathsf{CompAmComEq}})$.

Verify. Given srs, common inputs, and $\pi_{\mathtt{lnk}}$, parse $\pi_{\mathtt{lnk}}$ as $(\mathit{\Pi}, \pi_{\mathsf{CompAmComEq}})$.

- Check that w_{mid} vanishes at $\mathbb{H}_{\mathsf{com}}$:

$$e(c_{\mathsf{mid}}, h) = e(\mathit{\Pi}, [v_{\mathbb{H}[\leq l+d\ell]}(X)]_2)$$

- Verify $\pi_{\mathsf{CompAmComEq}}$.

Fig. 7. Commitment-linking protocol for Marlin

We now turn $\mathsf{AHP}_{\mathsf{Marlin}}$ into CP-Marlin by applying our compiler. We begin by giving a commit-and-prove relation for R1CS.

Relation for CP-Marlin. We define an extended relation to accommodate consistency of partial witness wire values and commitment. We use the index sets: $I_{\mathsf{pub}} = [l], I_{\mathsf{com}} = [l + 1, l + d\ell], I_{\mathsf{mid}} = [l + d\ell + 1, n]$, assuming that $z_{l+1}, \ldots, z_{l+d\ell}$ are $d\ell$ values committed to in advance. Moreover, every d values are batched into a single commitment, that is, every vector compound of d wires $(z_i)_{i\in I_k}$, for $I_k = [l+1+d(k-1), l+dk]$, is committed to in the kth auxiliary commitment $\hat{C}_k = \mathbf{G}^{(z_i)_{i\in I_k}} H^{r_k}$ for $k \in [\ell]$. Then we have $I_{\mathsf{com}} = \bigcup_{k\in[\ell]} I_k$. For values $\mathbf{z} \in \mathbb{F}^n$, we call $(w_i)_{i\in[l]} := (z_i)_{i\in[l]}$ *public input* and $(w_i)_{i\in[l+1,n]} := (z_i)_{i\in[l+1,n]}$ *witness*, respectively.

Definition 13 (CP-Marlin indexed relation). *The indexed relation $\mathcal{R}_{\mathsf{CP\text{-}Marlin}}$ is the set of all triples*

$$(\mathsf{i}, \mathsf{x}, \mathsf{w}) = ((\mathbb{F}, \mathbb{H}, \mathbb{K}, n, m, l, \ell, d, A, B, C), ((w_i)_{i\in[l]}, (\hat{C}_k)_{k\in[\ell]}), ((w_i)_{i\in[l+1,n]}, (r_k)_{k\in[\ell]}))$$

such that, for $\mathbf{w} := (w_i)_{i\in[n]} \in \mathbb{F}^n$

$$A\mathbf{w} \circ B\mathbf{w} = C\mathbf{w}, \ \textit{and} \ \forall k \in [\ell], \hat{C}_k = \mathsf{AC.Commit}_{\mathsf{ack}}((w_i)_{i\in I_k}; r_k).$$

Applying our compiler. We now show that $\mathsf{AHP}_{\mathsf{Marlin}}$ and the polynomial commitment scheme $\mathsf{PC}_{\mathsf{KZG}}$ [KZG10] meet the requirements of Theorem 1.

– Unique witness extraction: WitExt takes $\hat{w}(X)$, evaluates $\hat{w}(X)$ on every $k \in \mathbb{H}[> l]$ and constructs a vector of values $\mathbf{w} \in \mathbb{F}^{n-l}$. It is easy to see that WitExt satisfies unique extraction (Definition 8).

– Decomposable witness-carrying polynomials: Decomp takes $\hat{w}(X)$ and $I_{\mathsf{com}} \subset [n]$, and outputs $w_{\mathsf{com}} \in \mathbb{F}^{\ell d}[X], w_{\mathsf{mid}} \in \mathbb{F}^{n-l-\ell d}[X]$ computed as follows: Let $w = (w_i)_i \in \mathbb{F}^{n-l}$ be the vector such that $\hat{w}(X)$ agrees with w on $\mathbb{H}[> l]$. Let w_{com} and w_{mid} be polynomials that agree with w on $\mathbb{H}[> l \leq l + d\ell]$ and $\mathbb{H}[> l + d\ell]$ respectively.

  Then for $I_{\mathsf{com}}, I_{\mathsf{mid}}$ such that $I_{\mathsf{com}} \cup I_{\mathsf{mid}} = [n]$ and $I_{\mathsf{com}} \cap I_{\mathsf{mid}} = \varnothing$, Decomp clearly has the degree preserving property, and the non-overlapping property. For the additive decomposition, note that $w_{\mathsf{com}}(X) + w_{\mathsf{mid}}(X) = \hat{w}(X)$.

– $\mathsf{PC}_{\mathsf{KZG}}.\mathsf{Com}_{\mathsf{ck}}$ with maximum degree bound D takes a polynomial $f \in \mathbb{F}_{<D}[X]$ and outputs $[f(\chi)]_1$. Clearly, this is an additively homomorphic commitment scheme. As mentioned in §9.2 of [CHM+20] and as it's clear from how WitExt works, the knowledge soundness of Marlin holds only by enforcing degree bound to the maximum degree D for committed polynomials so the plain KZG construction should suffice for compiling $\mathsf{AHP}_{\mathsf{Marlin}}$.

We now present a suitable commitment-linking protocol $\mathsf{CP}_{\mathtt{lnk}}$ in Fig. 7. The key idea is to have the prover commit to an encoding of the assignment in index sets I_{com} and I_{mid} into separate polynomials, and then show that $w_{\mathsf{mid}}(X)$ vanishes at $\mathbb{H}[\leq l + d\ell]$, together with the consistency of w_{com} with vector Pedersen commitments $\hat{C}_k = \mathbf{G}^{(\mathsf{w}_i)_{i \in I_k}} H^{r_k}$ for $k \in [\ell]$ via $\mathsf{CompAmComEq}$ protocol (see Sect. 4). We assume that $I_{\mathsf{com}} = \bigcup_{k \in [\ell]} I_k$, I_k's are disjoint with each other and of same cardinality $d = |I_k|$.

Knowledge soundness of the commitment-linking protocol follows from the same argument as the knowledge soundness for the protocol in Fig 5. Instantiating our compiler from Fig. 1 with the $\mathsf{CP}_{\mathtt{lnk}}$ from Fig. 7 immediately yields an argument system for $\mathcal{R}_{\mathsf{CP\text{-}Marlin}}$.

7 Instantiation with Sonic

While the proving time of Marlin is an entire order of magnitude better than Sonic, and Marlin's verifier requires fewer pairings and fewer exponentiations, for applications that use batched verifications, Sonic remains the state-of-the-art. Applications like cryptocurrency transactions take advantage of batching where each verifier is not just given a single proof but many proofs of the same statement. This optimization works in the *helped* scenario, where an untrusted third party can aggregate such proofs in a single batch for faster verification.

Sonic is a zk-SNARK system in the universal SRS setting that can be used to prove any statement represented as an arithmetic circuit. While Sonic was orginally not presented in the language of AHPs, it was later recharacterized as an IOP by Bünz, Fisch and Szepieniec, which is essentially equivalent to the AHP framework [BFS20, §1.2].

The construction in Sonic relies on a special construction of polynomial commitments (a modifiation of KZG) that forces the prover to commit to a Laurent polynomial with no constant term.

7.1 Sonic AHP

We first describe the system of constraints used by Sonic. The vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ of length n, represent left inputs, right inputs and outputs respectively of the multiplication gates.

$$\mathbf{a} \odot \mathbf{b} = \mathbf{c}$$

Let $\mathbf{u}_q, \mathbf{v}_q, \mathbf{w}_q \in \mathbb{F}^n$ be fixed vectors for the qth linear constraint with instance values $k_q \in \mathbb{F}$. There are Q linear constraints of the form,

$$\mathbf{a} \cdot \mathbf{u}_q + \mathbf{b} \cdot \mathbf{v}_q + \mathbf{c} \cdot \mathbf{w}_q = k_q$$

The n multiplication constraints are compressed into one equation by introducing the formal indeterminate Y.

$$\sum_{i=1}^{n} (a_i b_i - c_i) Y^i = 0 \qquad \sum_{i=1}^{n} (a_i b_i - c_i) Y^{-i} = 0$$

The Q linear constraints are compressed,

$$\sum_{q=1}^{Q}(\mathbf{a}\cdot\mathbf{u}_q+\mathbf{b}\cdot\mathbf{v}_q+\mathbf{c}\cdot\mathbf{w}_q-k_q)Y^{q+n}=0$$

Define polynomials

$$u_i(Y)=\sum_{q=1}^{Q}Y^{q+n}u_{q,i} \qquad v_i(Y)=\sum_{q=1}^{Q}Y^{q+n}v_{q,i}$$
$$w_i(Y)=-Y^i-Y^{-i}+\sum_{q=1}^{Q}Y^{q+n}w_{q,i} \qquad k(Y)=\sum_{q=1}^{Q}Y^{q+n}k_q$$

Combining the multiplicative and linear constraints,

$$\mathbf{a}\cdot\mathbf{u}(Y)+\mathbf{b}\cdot\mathbf{v}(Y)+\mathbf{c}\cdot\mathbf{w}(Y)+\sum_{i=1}^{n}a_ib_i(Y^i+Y^{-i})-k(Y)=0 \tag{8}$$

The above holds at all points if the constraint system is satisfied. If the constraint system is not satisfied, the above will fail to hold with high probability for a large enough field. Now, the left hand side of the above is embedded into the constant term of a polynomial $t(X,Y)$ in another indeterminate X. A polynomial $r(X,Y)$ is designed such that $r(X,Y)=r(XY,1)$

$$r(X,Y)=\sum_{i=1}^{n}(a_iX^iY^i+b_iX^{-i}Y^{-i}+c_iX^{-n-i}Y^{-n-i}) \tag{9}$$
$$s(X,Y)=\sum_{i=1}^{n}\left(u_i(Y)X^{-i}+v_i(Y)X^i+w_i(Y)X^{i+n}\right) \tag{10}$$
$$t(X,Y)=r(X,1)(r(X,Y)+s(X,Y))-k(Y) \tag{11}$$

Note that the coefficient of X^0 in $t(X,Y)$ coincides with the left hand side of Eq. (8). We are now set out to define the Sonic indexed relation.

Definition 14 (Sonic indexed relation). *The indexed relation $\mathcal{R}_{\mathsf{Sonic}}$ is the set of all triples*

$$((\mathbb{F},n,Q,(\mathbf{u}_q)_{q\in[Q]},(\mathbf{v}_q)_{q\in[Q]},(\mathbf{w}_q)_{q\in[Q]}),(k_q)_{q\in[Q]},(\mathbf{a},\mathbf{b},\mathbf{c}))$$

such that

$$\forall q\in[Q]:\quad \mathbf{a}\cdot\mathbf{u}_q+\mathbf{b}\cdot\mathbf{v}_q+\mathbf{c}\cdot\mathbf{w}_q=k_q$$
$$\mathbf{a}\odot\mathbf{b}=\mathbf{c}$$

As mentioned above checking Eq. (8) is equivalent to checking whether an instance is in $\mathcal{R}_{\mathsf{Sonic}}$. To verify Eq. (8) Sonic implicitly relies on Lemma 3. In Fig. 8 we also present the underlying AHP of Sonic for $\mathcal{R}_{\mathsf{Sonic}}$, where the verifier V essentially checks the second point of the following lemma.

Lemma 3. *The following two properties hold.*

1. *Let $r(X,Y),s(X,Y)$ and $t(X,Y)$ be given as above. If Eq. (8) holds, then the constant term of $t(X,Y)$ w.r.t. X is zero.*
2. *Let $s(X,Y)$ be given as above and suppose that $r(X,Y)$ is a Laurent polynomial of the form $r(X,Y)=\sum_{-D}^{n}r_iX^iY^i$. If the constant term w.r.t X of*

$$r(X,1)(r(X,Y)+s(X,Y))-k(Y)$$

is zero, then Eq. (8) holds for $\mathbf{a}=(r_i)_{i=1}^n$, $\mathbf{b}=(r_{-i})_{i=1}^n$ and $\mathbf{c}=(r_{-i-n})_{i=1}^n$.

Proof. For the first statement, we notice that it follows directly from the fact the constant term w.r.t. X of $t(X,Y)$ is exactly the left hand side of Eq. (8). For the second statement, let $r(X,Y)=\sum_{-D}^{n}r_iX^iY^i$

Protocol $\mathsf{AHP}_{\mathsf{Sonic}}$

Offline phase. The indexer I receives as input $\mathbb{F} \in \mathcal{F}$ and $\mathsf{i} = (\mathbb{F}, n, Q, (\mathbf{u}_q)_{q\in[Q]}, (\mathbf{v}_q)_{q\in[Q]}, (\mathbf{w}_q)_{q\in[Q]})$, and computes the polynomial oracle $s(X, Y)$ as described in the text.

Input. P receives $(\mathbb{F}, \mathsf{i}, (k_q)_{q\in[Q]}, (\mathbf{a}, \mathbf{b}, \mathbf{c}))$ and V receives $(\mathbb{F}, (k_q)_{q\in[Q]})$ and oracle access to the polynomials output by $\mathsf{I}(\mathbb{F}, \mathsf{i})$.

Online phase: first round. P computes $r(X, Y)$ and $t(X, Y)$ as described in Eq. (11). Blind $r(X, Y)$ as $r(X, Y) := r(X, Y) + \sum_{i=1}^{4} c_{n+i} X^{-2n-i} Y^{-2n-i}$ with random $c_{n+i} \in \mathbb{F}$ and send an oracle polynomial $r(X, 1)$ to V.

Online phase: second round. Upon receiving challenges $y \in \mathbb{F}$ from the V, P sends an oracle polynomial $t(X, y)$ to V.

Query phase. V queries online oracles $r(X, 1)$ and $t(X, y)$ with a random query point $z \in \mathbb{F}$. Moreover, it makes additional queries to $r(X, 1)$ with yz and to $s(X, Y)$ with (z, y).

Decision phase. V first computes an instance polynomial $k(Y)$ as described in the text. Then V checks that

$$t(z, y) \stackrel{?}{=} r(z, 1)(r(yz, 1) + s(z, y)) - k(y).$$

Fig. 8. AHP for $\mathcal{R}_{\mathsf{Sonic}}$

with $r_i \in \mathbb{F}$. Now, we notice that

$$\begin{aligned} r(X,1)(r(X,Y)+s(X,Y)) - k(Y) = &\left(r_0 + \sum_{i=1}^{n} (r_i X^i + r_{-i} X^{-i} + r_{-i-n} X^{-i-n}) + \sum_{i=2n+1}^{D} r_{-i} X^{-i} \right) \\ &\cdot \left(r_0 + \sum_{i=1}^{n} \left(r_i (XY)^i + r_{-i} (XY)^{-i} + r_{-i-n} (XY)^{-i-n} \right) \right. \\ &+ \sum_{i=2n+1}^{D} r_{-i} (XY)^{-i} \\ &\left. + \sum_{i=1}^{n} \left(u_i(Y) X^{-i} + v_i(Y) X^i + w_i(Y) X^{i+n} \right) \right) - k(Y). \end{aligned}$$

From the above we see that the constant term w.r.t. X is

$$r_0^2 + \sum_{i=1}^{n} r_i r_{-i} (Y^i + Y^{-i}) + \sum_{i=1}^{n} r_i u_i(Y) + \sum_{i=1}^{n} r_{-i} v_i(Y) + \sum_{i=1}^{n} r_{-i-n} w_i(Y) - k(Y),$$

which can only be zero if $r_0 = 0$. It therefore follows as wanted that if the constant term w.r.t. X of $r(X, 1)(r(X, Y) + s(X, Y)) - k(Y)$ is zero, then Eq. (8) holds for $\mathbf{a} = (r_i)_{i=1}^n$, $\mathbf{b} = (r_{-i})_{i=1}^n$ and $\mathbf{c} = (r_{-i-n})_{i=1}^n$.

7.2 CP-Sonic

Our goal is to turn $\mathsf{AHP}_{\mathsf{Sonic}}$ into CP-Sonic with our compiler. We first describe a commit-and-prove variant of relation $\mathcal{R}_{\mathsf{Sonic}}$. We assume without loss of generality that every committed witness is left input to gate i, i.e., $(a_i)_{i\in I_{\mathsf{com}}}$ is the committed witness whereas $((a_i)_{i\notin I_{\mathsf{com}}}, \mathbf{b}, \mathbf{c})$ is the non-committed part. Then we use the following disjoint witness index sets: $I_{\mathsf{com}} = [n - \ell d + 1, n]$, $I_{\mathsf{mid}} = [1, n - \ell d]$, assuming that $a_{n-\ell d+1}, \ldots, a_n$ are ℓd witness values committed in advance. Moreover, suppose every vector compound of d values $(a_i)_{i\in I_k}$, where $I_k = [n - dk + 1, n - d(k - 1)]$, is committed into kth auxiliary commitment $\hat{C}_k$ for $k \in [\ell]$. Then we have $I_{\mathsf{com}} = \bigcup_{k\in[\ell]} I_k$.

Definition 15 (CP-Sonic indexed relation). *The indexed relation $\mathcal{R}_{\mathsf{CP\text{-}Sonic}}$ is the set of all triples*

$$((\mathbb{F}, n, Q, (\mathbf{u}_q)_{q\in[Q]}, (\mathbf{v}_q)_{q\in[Q]}, (\mathbf{w}_q)_{q\in[Q]}, I_{\mathsf{com}}, (I_k)_{k\in[\ell]}, \mathsf{ack}), ((k_q)_{q\in[Q]}, (C_k)_{k\in[\ell]}), (\mathbf{a}, \mathbf{b}, \mathbf{c}, (r_k)_{k\in[\ell]}))$$

such that

$$\begin{aligned} \forall q \in [Q]: \quad & \mathbf{a} \cdot \mathbf{u}_q + \mathbf{b} \cdot \mathbf{v}_q + \mathbf{c} \cdot \mathbf{w}_q = k_q \\ & \mathbf{a} \odot \mathbf{b} = \mathbf{c} \\ \forall k \in [\ell]: \quad & \hat{C}_k = \mathsf{AC.Com}_{\mathsf{ack}}((a_i)_{i\in I_k}; r_k) \end{aligned}$$

7.2.1 Applying our compiler We show that $\mathsf{AHP}_{\mathsf{Sonic}}$ as well as the polynomial commitment scheme meets the requirements of Theorem 1.

– Decomp takes $\mathsf{n_w} = 1$ (blinded) witness-carrying polynomial $r(X) := r(X, 1)$ and $I_{\mathsf{com}} \subset [n]$, parses $r(X)$ as $\sum_{i=1}^{n}(a_i X^i + b_i X^{-i} + c_i X^{-n-i}) + \sum_{i=1}^{4} c_{n+i} X^{-2n-i}$, and decompose them as follows.

$$r_{\mathsf{com}}(X) := \sum_{i \in I_{\mathsf{com}}} a_i X^i + \sum_{i=1}^{4} \rho_{n+i} X^{-2n-i}$$
$$r_{\mathsf{mid}}(X) := \sum_{i \in I_{\mathsf{mid}}} a_i X^i + \sum_{i=1}^{n}(b_i X^{-i} + c_i X^{-n-i}) + \sum_{i=1}^{4} \lambda_{n+i} X^{-2n-i}$$

where ρ_{n+i} was randomly chosen and $\lambda_{n+i} := c_{n+i} - \rho_{n+i}$ for $i = 1, 2, 3, 4$. Clearly, the decomposition is additive, degree-preserving (in the sense that separate Laurent polynomials do not exceed the prescribed degree range), and non-overlapping.

– WitExt takes a witness-carrying polynomial $r(X) = \sum_{-D}^{n} r_i X^i$ and uniquely extracts witness vectors $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ such that $a_i := r_i$, $b_i := r_{-i}$ and $c_i := r_{-n-i}$ for every $i \in [n]$.

– Sonic uses a variant of the KZG scheme optimized for Laurent polynomials with a 0 constant term. Concretely, $\mathsf{PC}_{\mathsf{Sonic}}.\mathsf{Com}$ takes as input

$$\mathsf{ck} = ([\chi^{-D}]_1, \ldots, [\chi^{D}]_1, [\alpha\chi^{-D}]_1, \ldots, [\alpha\chi^{-1}]_1, [\alpha\chi]_1, \ldots, [\alpha\chi^{D}]_1, [\chi^{-D}]_2, \ldots, [\chi^{D}]_2, [\alpha\chi^{-D}]_2, \ldots, [\alpha\chi^{D}]_2),$$

a polynomial $f(X) \in \mathbb{F}[X, X^{-1}]$, and the degree bound $\mathsf{d} \leq D$, and then outputs $[\alpha\chi^{D-\mathsf{d}} f(\chi)]_1$. Clearly, this is an additively homomorphic commitment scheme. In the AGM its evaluation binding and extractability were formally proved under the $2D$-DLOG assumption (see Theorem 6.3 of [MBKM19]). The plain binding for a fixed degree bound can be also shown just as in the KZG scheme. Unlike PLONK, Sonic must enforce a precise degree bound n on the witness-carrying polynomial $r(X)$ to achieve knowledge soundness. Our commit-and-prove variant should thus enforce the same bound on both $r_{\mathsf{com}}(X)$ and $r_{\mathsf{mid}}(X)$. Finally, Sonic retains zero-knowledge by blinding witness-carrying polynomial, instead of hiding commitment. Hence commitment randomness is empty for all commitments. To sum up, the compiled protocol involves the following commitments to decomposed witness-carrying polynomials.

$$C_{\mathsf{com}} = [\alpha\chi^{D-n} r_{\mathsf{com}}(\chi)]_1$$
$$C_{\mathsf{mid}} = [\alpha\chi^{D-n} r_{\mathsf{mid}}(\chi)]_1$$

We now present a suitable commitment-linking protocol $\mathsf{CP}_{\mathtt{lnk}}$ in Fig. 9. The high-level idea is to (1) prove consistency between $r_{\mathsf{com}}(X)$ and auxiliary commitments $\hat{C}_k$ with the $\mathsf{AmComEq}$ protocol, and (2) force the prover to show $r_{\mathsf{mid}}(X)$ has degree bounded by $n - \ell d$. The latter is in particular crucial for WitExt to successfully output a witness vector consistent with auxiliary commitments, even after taking the sum of $r_{\mathsf{com}}(X)$ and $r_{\mathsf{mid}}(X)$.

Lemma 4. *Assuming hardness of the $2D$-DLOG problem, extractability of $\mathsf{PC}_{\mathsf{Sonic}}$ and argument of knowledge of $\mathsf{CompAmComEq}$, the protocol $\mathsf{CP}_{\mathtt{lnk}}$ (Fig. 9) is an argument of knowledge in the algebraic group model [FKL18].*

Proof. First, the extractor $\mathcal{E}_{\mathtt{lnk}}$ obtains $r(X) \in \mathbb{F}[X, X^{-1}]$ of degree at most n such that $[\alpha\chi^{D-n} r(\chi)]_1 = C_{\mathsf{com}} \cdot C_{\mathsf{mid}}$ and $r(z) = v$, by internally invoking an extractor for $\mathsf{PC}_{\mathsf{Sonic}}$, which succeeds with overwhelming probability as long as a malicous prover $\mathcal{P}^*_{\mathtt{lnk}}$ convinces the verifier.

Second, $\mathcal{E}_{\mathtt{lnk}}$ invokes an extractor $\mathcal{E}_{\mathsf{ComEq}}$ for the $\mathsf{CompAmComEq}$ protocol, which outputs $(a_i)_{i \in I_{\mathsf{com}}}$ and $(r_k)_{k \in [\ell]}$ such that $\hat{C}_k = \mathsf{AC}.\mathsf{Com}_{\mathsf{ack}}((a_i)_{i \in I_k}; r_k)$ for $k \in [\ell]$, and $C_{\mathsf{com}} = [\alpha(\sum_{i \in I_{\mathsf{com}}} a_i \chi^{D-n+i} + \sum_{i \in [1,4]} \rho_{n+i} \chi^{D-3n-i})]_1$. So we have extracted $r_{\mathsf{com}}(X) = \sum_{i \in I_{\mathsf{com}}} a_i X^i + \sum_{i \in [1,4]} \rho_{n+i} X^{-2n-i}$ such that $C_{\mathsf{com}} = \mathsf{PC}_{\mathsf{Sonic}}.\mathsf{Com}_{\mathsf{ck}}(r_{\mathsf{com}}(X), n)$.

Let $r_{\mathsf{mid}}(X) := r(X) - r_{\mathsf{com}}(X)$. Due to the homomorphism of committing function it holds that $C_{\mathsf{mid}} = C \cdot C_{\mathsf{com}}^{-1} = \mathsf{PC}_{\mathsf{Sonic}}.\mathsf{Com}_{\mathsf{ck}}(r_{\mathsf{mid}}(X), n) = [\alpha\chi^{D-n} r_{\mathsf{mid}}(\chi)]_1$. Due to the second pairing check we also have that $C'_{\mathsf{mid}} = (C_{\mathsf{mid}})^{\chi^{\ell d}} = [\alpha\chi^{D-n+\ell d} r_{\mathsf{mid}}(\chi)]_1$.

On the other hand, when an algebraic adversary $\mathcal{P}^*_{\mathtt{lnk}}$ outputs C'_{mid} it is accompanied by the representation $f_\chi(X) + X_\alpha f_\alpha(X)$ such that $C'_{\mathsf{mid}} = [f_\chi(\chi) + \alpha f_\alpha(\chi)]_1$, $f_\chi(X)$ has non-zero terms between degree $-D$ and D, and $f_\alpha(X)$ has non-zero terms between degree $-D$ and D except for the constant term. If

$f_\chi(X) + X_\alpha f_\alpha(X) \neq X_\alpha X^{D-n+\ell d} r_{\mathsf{mid}}(X)$ we have two distinct representations of C'_{mid}, from which one can find χ solving the $2D$-DLOG problem, as in a proof of Theorem 6.3 of [MBKM19]. Hence we may assume that $f_\chi(X) = 0$ and $f_\alpha(X) = X^{D-n+\ell d} r_{\mathsf{mid}}(X)$, implying that $r_{\mathsf{mid}}(X)$ has degree bounded by $n - \ell d$.

Now the committed part of coefficients of $r(X)$ corresponds to extracted $r_{\mathsf{com}}(X)$. Hence if WitExt is invoked on $r(X)$ it does extract witness $(a_i)_{i \in I_{\mathsf{com}}}$ consistent with $(\hat{C}_k)_{k \in [\ell]}$, which is guaranteed by $\mathcal{E}_{\mathsf{ComEq}}$.

Lemma 5. *Assuming zero knowledge of Fiat–Shamir-transformed* $\mathsf{CompAmComEq}$*, the protocol* $\mathsf{CP}_{\mathtt{lnk}}$ *is zero-knowledge in the SRS model.*

Proof. To simulate π_{ComEq} we simply invoke the zero-knowledge simulator for $\mathsf{CompAmComEq}$ made non-interactive with Fiat–Shamir [FS87]. To simulate the evaluation proof $\mathit{\Pi}$ the simulator uses the trapdoor α and χ used for generating the commitment key to compute $\mathit{\Pi} := \left((C_{\mathsf{com}} \cdot C_{\mathsf{mid}})^{\frac{1}{\alpha\chi^{-D+n}}} \cdot [-v]_1\right)^{\frac{1}{\chi - z}}$. To simulate C'_{mid} we compute $C'_{\mathsf{mid}} := C_{\mathsf{mid}}^{\chi^{\ell d}}$.

Protocol $\mathsf{CP}_{\mathtt{lnk}}$ for Sonic

Input. Both $\mathcal{P}_{\mathtt{lnk}}$ and $\mathcal{V}_{\mathtt{lnk}}$ receives $(\mathsf{ck}, \mathsf{ack}, (\hat{C}_k)_{k \in [\ell]}, (C_{\mathsf{com}}, C_{\mathsf{mid}}), z, v)$ as statements. The $\mathcal{P}_{\mathtt{lnk}}$ has as input witness $(r_{\mathsf{com}}(X), r_{\mathsf{mid}}(X), (r_k)_{k \in [\ell]})$ such that

$$r_{\mathsf{com}}(X) = \sum_{i \in I_{\mathsf{com}}} a_i X^i + \sum_{i=1}^{4} \rho_{n+i} X^{-2n-i} \qquad r_{\mathsf{mid}}(X) = \sum_{i \in I_{\mathsf{com}}} a_i X^i + \sum_{i=1}^{n} (b_i X^{-i} + c_i X^{-n-i}) + \sum_{i=1}^{4} \lambda_{n+i} X^{-2n-i}$$

$$\hat{C}_k = \mathbf{G}^{(a_i)_{i \in I_k}} H^{r_k} \qquad C_{\mathsf{com}} = [\alpha \chi^{D-n} r_{\mathsf{com}}(\chi)]_1 \qquad C_{\mathsf{mid}} = [\alpha \chi^{D-n} r_{\mathsf{mid}}(\chi)]_1 \qquad v = r_{\mathsf{com}}(z) + r_{\mathsf{mid}}(z)$$

Prove.

- Compute a proof $\pi_{\mathsf{CompAmComEq}}$ of the following statement where $g_i := [\alpha \chi^{D-n+i}]$ for $i \in I_{\mathsf{com}}$, $\boldsymbol{g} := (g_i)_{i \in I_{\mathsf{com}}}$, $h_i := [\alpha \chi^{D-3n-i}]$ for $i \in [1,4]$, and $\mathbf{h} := (h_i)_{i \in [1,4]}$.

$$\mathsf{CompAmComEq}\,.\mathsf{PK}\left\{((a_i)_{i \in I_{\mathsf{com}}}, (r_k)_{k \in [\ell]}, (\rho_{n+i})_{i \in [1,4]}) \;:\; \hat{C}_k = \mathbf{G}^{(a_i)_{i \in I_k}} H^{r_k} \wedge C_{\mathsf{com}} = \boldsymbol{g}^{(a_i)_{i \in I_{\mathsf{com}}}} \mathbf{h}^{(\rho_{n+i})_{i \in [1,4]}}\right\}$$

- Let $r(X) := r_{\mathsf{com}}(X) + r_{\mathsf{mid}}(X)$. Compute evaluation proof as follows.

$$W(X) = \frac{r(X) - v}{X - z} \qquad \mathit{\Pi} = [W(\chi)]_1$$

- Compute a shifted commitment as follows.

$$C'_{\mathsf{mid}} := [\alpha \chi^{D-n+\ell d} r_{\mathsf{mid}}(\chi)]_1 = \mathsf{PC}_{\mathsf{Sonic}}.\mathsf{Com}_{\mathsf{ck}}(r_{\mathsf{mid}}(X), n - \ell d)$$

- Output $\pi_{\mathtt{lnk}} := (\pi_{\mathsf{CompAmComEq}}, \mathit{\Pi}, C'_{\mathsf{mid}})$.

Verify. Given $\pi_{\mathtt{lnk}}$, verify $\pi_{\mathsf{CompAmComEq}}$, check evaluation proof:

$$e(\mathit{\Pi}, [\alpha\chi]_2) \cdot e([v]_1 \cdot \mathit{\Pi}^{-z}, [\alpha]_2) \stackrel{?}{=} e(C_{\mathsf{com}} \cdot C_{\mathsf{mid}}, [\chi^{-D+n-\ell d}]_2)$$

and check $r_{\mathsf{mid}}(X)$ has degree at most $n - \ell d$:

$$e(C_{\mathsf{mid}}, [\chi^{\ell d}]_2) \stackrel{?}{=} e(C'_{\mathsf{mid}}, h)$$

Fig. 9. Commitment-linking protocol for Sonic

References

AC20. T. Attema and R. Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In *CRYPTO 2020, Part III*, vol. 12172 of *LNCS*, pp. 513–543. Springer, Heidelberg, 2020.

ACF20. T. Attema, R. Cramer, and S. Fehr. Compressing proofs of k-out-of-n partial knowledge. Cryptology ePrint Archive, Report 2020/753, 2020. https://eprint.iacr.org/2020/753.

ACK21. T. Attema, R. Cramer, and L. Kohl. A compressed σ-protocol theory for lattices. Cryptology ePrint Archive, Report 2021/307, 2021. https://eprint.iacr.org/2021/307.

ACR20. T. Attema, R. Cramer, and M. Rambaud. Compressed σ-protocols for bilinear group arithmetic circuits and applications. Cryptology ePrint Archive, Report 2020/1447, 2020. https://eprint.iacr.org/2020/1447.

AGM18. S. Agrawal, C. Ganesh, and P. Mohassel. Non-interactive zero-knowledge proofs for composite statements. In *CRYPTO 2018, Part III*, vol. 10993 of *LNCS*, pp. 643–673. Springer, Heidelberg, 2018.

BBB+18. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pp. 315–334. IEEE Computer Society Press, 2018.

BCC+16. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT 2016, Part II*, vol. 9666 of *LNCS*, pp. 327–357. Springer, Heidelberg, 2016.

BCG+13. E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In *CRYPTO 2013, Part II*, vol. 8043 of *LNCS*, pp. 90–108. Springer, Heidelberg, 2013.

BCG+14. E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pp. 459–474. IEEE Computer Society Press, 2014.

BCI+13. N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC 2013*, vol. 7785 of *LNCS*, pp. 315–333. Springer, Heidelberg, 2013.

BCR+19. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In *EUROCRYPT 2019, Part I*, vol. 11476 of *LNCS*, pp. 103–128. Springer, Heidelberg, 2019.

BCTV14. E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *USENIX Security 2014*, pp. 781–796. USENIX Association, 2014.

BDFG20. D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Report 2020/081, 2020. https://eprint.iacr.org/2020/081.

BFS20. B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. In *EUROCRYPT 2020, Part I*, vol. 12105 of *LNCS*, pp. 677–706. Springer, Heidelberg, 2020.

BGG+90. M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway. Everything provable is provable in zero-knowledge. In *CRYPTO'88*, vol. 403 of *LNCS*, pp. 37–56. Springer, Heidelberg, 1990.

BGM17. S. Bowe, A. Gabizon, and I. Miers. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017. https://eprint.iacr.org/2017/1050.

BHH+19. M. Backes, L. Hanzlik, A. Herzberg, A. Kate, and I. Pryvalov. Efficient non-interactive zero-knowledge proofs in cross-domains without trusted setup. In *PKC 2019, Part I*, vol. 11442 of *LNCS*, pp. 286–313. Springer, Heidelberg, 2019.

CDG+17. M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *ACM CCS 2017*, pp. 1825–1842. ACM Press, 2017.

CDS94. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO'94*, vol. 839 of *LNCS*, pp. 174–187. Springer, Heidelberg, 1994.

CFF+20. M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodríguez. Lunar: a toolbox for more efficient universal and updatable zksnarks and commit-and-prove extensions. Cryptology ePrint Archive, Report 2020/1069, 2020. https://eprint.iacr.org/2020/1069.

CFQ19. M. Campanelli, D. Fiore, and A. Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In *ACM CCS 2019*, pp. 2075–2092. ACM Press, 2019.

CGM16. M. Chase, C. Ganesh, and P. Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In *CRYPTO 2016, Part III*, vol. 9816 of *LNCS*, pp. 499–530. Springer, Heidelberg, 2016.

CHM+20. A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *EUROCRYPT 2020, Part I*, vol. 12105 of *LNCS*, pp. 738–768. Springer, Heidelberg, 2020.

CL01. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT 2001*, vol. 2045 of *LNCS*, pp. 93–118. Springer, Heidelberg, 2001.

CS97. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO'97*, vol. 1294 of *LNCS*, pp. 410–424. Springer, Heidelberg, 1997.

DGK+21. I. Damgård, C. Ganesh, H. Khoshakhlagh, C. Orlandi, and L. Siniscalchi. Balancing privacy and accountability in blockchain identity management. In *CT-RSA 2021*, vol. 12704, pp. 552–576. Springer, 2021.

FFS87. U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. In *19th ACM STOC*, pp. 210–217. ACM Press, 1987.

FKL18. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *CRYPTO 2018, Part II*, vol. 10992 of *LNCS*, pp. 33–62. Springer, Heidelberg, 2018.

For87. L. Fortnow. The complexity of perfect zero-knowledge (extended abstract). In *19th ACM STOC*, pp. 204–209. ACM Press, 1987.

FS87. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, vol. 263 of *LNCS*, pp. 186–194. Springer, Heidelberg, 1987.

GGPR13. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *EUROCRYPT 2013*, vol. 7881 of *LNCS*, pp. 626–645. Springer, Heidelberg, 2013.

GKM+18. J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In *CRYPTO 2018, Part III*, vol. 10993 of *LNCS*, pp. 698–728. Springer, Heidelberg, 2018.

GMO16. I. Giacomelli, J. Madsen, and C. Orlandi. ZKBoo: Faster zero-knowledge for Boolean circuits. In *USENIX Security 2016*, pp. 1069–1083. USENIX Association, 2016.

GMR85. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pp. 291–304. ACM Press, 1985.

GMW86. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th FOCS*, pp. 174–187. IEEE Computer Society Press, 1986.

GMW87. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th ACM STOC*, pp. 218–229. ACM Press, 1987.

GQ88. L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In *EUROCRYPT'88*, vol. 330 of *LNCS*, pp. 123–128. Springer, Heidelberg, 1988.

Gro10. J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT 2010*, vol. 6477 of *LNCS*, pp. 321–340. Springer, Heidelberg, 2010.

Gro16. J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016, Part II*, vol. 9666 of *LNCS*, pp. 305–326. Springer, Heidelberg, 2016.

GWC19. A. Gabizon, Z. J. Williamson, and O. Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

IKO07. Y. Ishai, E. Kushilevitz, and R. Ostrovsky. Efficient arguments without short pcps. In *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*, pp. 278–291. IEEE, 2007.

JKO13. M. Jawurek, F. Kerschbaum, and C. Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *ACM CCS 2013*, pp. 955–966. ACM Press, 2013.

KPV19. A. Kattis, K. Panarin, and A. Vlasov. Redshift: Transparent snarks from list polynomial commitment iops. Cryptology ePrint Archive, Report 2019/1400, 2019. https://eprint.iacr.org/2019/1400.

KZG10. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT 2010*, vol. 6477 of *LNCS*, pp. 177–194. Springer, Heidelberg, 2010.

Lip12. H. Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC 2012*, vol. 7194 of *LNCS*, pp. 169–189. Springer, Heidelberg, 2012.

Lip13. H. Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In *ASIACRYPT 2013, Part I*, vol. 8269 of *LNCS*, pp. 41–60. Springer, Heidelberg, 2013.

Max15. G. Maxwell. Confidential transactions. *URL: https://people. xiph. org/greg/confidential values. txt*, 2015.

MBKM19. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In *ACM CCS 2019*, pp. 2111–2128. ACM Press, 2019.

NY90. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pp. 427–437. ACM Press, 1990.

Ped92. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*, vol. 576 of *LNCS*, pp. 129–140. Springer, Heidelberg, 1992.

PHGR13. B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pp. 238–252. IEEE Computer Society Press, 2013.

Sch90. C.-P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO'89*, vol. 435 of *LNCS*, pp. 239–252. Springer, Heidelberg, 1990.

A Additional Materials on Compressed Σ-protocol Theory

A.1 ComEq: Proving equality of two Pedersen vector commitments

In this section, we first define a naïve ComEq protocol proving equality of vectors committed in two Pedersen commitments, with proof size of $O(d)$. Our goal is to give a protocol for the relation

$$\mathcal{R}_{\mathsf{ComEq}} = \left\{ ((\boldsymbol{g}, \mathbf{h}, \mathbf{G}, \mathbf{H}, d, d', d''), (C, \hat{C}), (\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})) \ : \ \begin{array}{c} C = \boldsymbol{g}^{\mathbf{w}}\mathbf{h}^{\boldsymbol{\alpha}}, \hat{C} = \mathbf{G}^{\mathbf{w}}\mathbf{H}^{\boldsymbol{\beta}}, \boldsymbol{g}, \mathbf{G} \in \mathbb{G}^d, \mathbf{w} \in \mathbb{Z}_q^d \\ \mathbf{h} \in \mathbb{G}^{d'}, \mathbf{H} \in \mathbb{G}^{d''}, \boldsymbol{\alpha} \in \mathbb{Z}_q^{d'}, \boldsymbol{\beta} \in \mathbb{Z}_q^{d''} \end{array} \right\} \tag{12}$$

where we assume that d' and d'' are small constants. Fig. 10 shows a bare-bone protocol for $\mathcal{R}_{\mathsf{ComEq}}$.

Protocol ComEq

1. $\mathcal{P}$ samples random $\mathbf{r} \in \mathbb{Z}_q^d$, $\boldsymbol{\delta} \in \mathbb{Z}_q^{d'}$, $\boldsymbol{\gamma} \in \mathbb{Z}_q^{d''}$ and sends
$$A = \boldsymbol{g}^{\mathbf{r}}\mathbf{h}^{\boldsymbol{\delta}}, \quad \hat{A} = \mathbf{G}^{\mathbf{r}}\mathbf{H}^{\boldsymbol{\gamma}}$$
2. $\mathcal{V}$ sends random challenge $e \in \mathbb{Z}_q$.
3. $\mathcal{P}$ sends
$$\mathbf{z} = \mathbf{r} + e\mathbf{w}, \quad \boldsymbol{\omega} = \boldsymbol{\delta} + e\boldsymbol{\alpha}, \quad \boldsymbol{\Omega} = \boldsymbol{\gamma} + e\boldsymbol{\beta}$$
4. $\mathcal{V}$ checks
$$\boldsymbol{g}^{\mathbf{z}}\mathbf{h}^{\boldsymbol{\omega}} \stackrel{?}{=} AC^e, \quad \mathbf{G}^{\mathbf{z}}\mathbf{H}^{\boldsymbol{\Omega}} \stackrel{?}{=} \hat{A}\hat{C}^e$$

Fig. 10. Σ-protocol for equality of vector Pedersen commitments.

Theorem 4. ComEq *is a Σ-protocol for the relation $\mathcal{R}_{\mathsf{ComEq}}$. Moreover, the communication costs are:*
- *$\mathcal{P} \to \mathcal{V}$: 2 elements of $\mathbb{G}$ and $d + d' + d'' + 2$ elements of $\mathbb{Z}_q$.*
- *$\mathcal{V} \to \mathcal{P}$: 1 elements of $\mathbb{Z}_q$.*

Proof. **Special soundness.** Given two accepting transcripts $(A, \hat{A}, e, \mathbf{z}, \boldsymbol{\Omega}, \boldsymbol{\omega})$ and $(A, \hat{A}, e', \mathbf{z}', \boldsymbol{\Omega}', \boldsymbol{\omega}')$ we extract valid witness as follows.

$$\mathbf{w} = (\mathbf{z} - \mathbf{z}')/(e - e'), \quad \boldsymbol{\alpha} = (\boldsymbol{\omega} - \boldsymbol{\omega}')/(e - e'), \quad \boldsymbol{\beta} = (\boldsymbol{\Omega} - \boldsymbol{\Omega}')/(e - e') \tag{13}$$

Special HVZK. Given e, the simulator samples random $\mathbf{z} \in \mathbb{Z}_q^d$ as well as $\boldsymbol{\omega} \in \mathbb{Z}_q^{d'}$ and $\boldsymbol{\Omega} \in \mathbb{Z}_q^{d''}$. Then the first messages are determined such that $A = \boldsymbol{g}^{\mathbf{z}}\mathbf{h}^{\boldsymbol{\omega}}C^{-e}$ and $\hat{A} = \mathbf{G}^{\mathbf{z}}\mathbf{H}^{\boldsymbol{\Omega}}\hat{C}^{-e}$.

Let CompComEq be a protocol identical to ComEq, except that its last move is replaced by the compression mechanisim CompDLEq (Fig. 3). Then we obtain the following.

Corollary 2. CompComEq *is a $(2\mu+3)$-move protocol for the relation $\mathcal{R}_{\mathsf{ComEq}}$, where $\mu = \lceil \log_2(d) \rceil - 1$. It is perfectly complete and unconditionally $(2, k_1, \ldots, k_\mu)$-special sound, where $k_i = 3$ for all $i \in [1, \mu]$. Moreover, the communication costs are:*
- *$\mathcal{P} \to \mathcal{V}$: $4\lceil \log_2(d) \rceil - 2$ elements of $\mathbb{G}$ and $2 + d' + d''$ elements of $\mathbb{Z}_q$.*
- *$\mathcal{V} \to \mathcal{P}$: $\lceil \log_2(d) \rceil$ elements of $\mathbb{Z}_q$.*

A.2 AmComEq′: as a result of [ACF20]

Protocol AmComEq′

1. $\mathcal{V}$ sends random challenge $x \in \mathbb{Z}_q$. Both parties compute

$$\tilde{\mathbf{G}} = [\mathbf{G}, \mathbf{G}^x, \ldots, \mathbf{G}^{x^{\ell-1}}] \in \mathbb{G}^{\ell d}$$
$$\tilde{\mathbf{H}} = [\mathbf{H}, \mathbf{H}^x, \ldots, \mathbf{H}^{x^{\ell-1}}] \in \mathbb{G}^{\ell d''}.$$

2. $\mathcal{P}$ samples random $\mathbf{r} \in \mathbb{Z}_q^{\ell d}$, $\boldsymbol{\delta} \in \mathbb{Z}_q^{d'}$, $\boldsymbol{\gamma} \in \mathbb{Z}_q^{\ell d''}$, and sends

$$A = \boldsymbol{g}^{\mathbf{r}} \mathbf{h}^{\boldsymbol{\delta}}$$
$$\hat{A} = \tilde{\mathbf{G}}^{\mathbf{r}} \tilde{\mathbf{H}}^{\boldsymbol{\gamma}}$$

3. $\mathcal{V}$ sends random challenge $e \in \mathbb{Z}_q$.
4. $\mathcal{P}$ sends

$$\mathbf{z} = \mathbf{r} + e\mathbf{w}, \quad \boldsymbol{\omega} = \boldsymbol{\delta} + e\boldsymbol{\alpha}, \quad \boldsymbol{\Omega} = \boldsymbol{\gamma} + e\boldsymbol{\beta}$$

5. $\mathcal{V}$ checks

$$\boldsymbol{g}^{\mathbf{z}} \mathbf{h}^{\boldsymbol{\omega}} \stackrel{?}{=} AC^e, \quad \tilde{\mathbf{G}}^{\mathbf{z}} \tilde{\mathbf{H}}^{\boldsymbol{\Omega}} \stackrel{?}{=} \hat{A} \prod_{i=1}^{\ell} (\hat{C}_i^{x^{i-1}})^e$$

Fig. 11. Four-move protocol for amortized equality of many vector Pedersen commitments

B PLONK Preliminaries

Conventions. We use i as an index for *gate* and j for *wire*.

B.1 PLONK constraint systems.

We consider a fan-in two arithmetic circuit over $\mathbb{F}$, consisting of n gates and m wires. The vector $\mathbf{w} \in \mathbb{F}^m$ consists of assigned *wire values*. The *index vector* $\mathbf{v} = \mathbf{L}||\mathbf{R}||\mathbf{O} \in [m]^{3n}$ represents the indices of wires for each gate: concretely, for each $i \in [n]$, $\mathbf{L}_i$ represents left, $\mathbf{R}_i$ represents right, and $\mathbf{O}_i$ represents output wire of gate j, respectively. For example, the left input wire value of i-th gate is obtained by $w_{\mathbf{L}_i}$. The per-gate constraints are specified by *selector vectors* $\mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O, \mathbf{q}_M, \mathbf{q}_C \in \mathbb{F}^n$. We call $\mathcal{C} = (n, m, \mathbf{L}, \mathbf{R}, \mathbf{O}, \mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O, \mathbf{q}_M, \mathbf{q}_C)$ *constraint systems*. We say that $\mathbf{w} \in \mathbb{F}^m$ *satisfies the constraint systems* $\mathcal{C}$ if for each gate $i \in [n]$

$$(\mathbf{q}_L)_i \cdot w_{\mathbf{L}_i} + (\mathbf{q}_R)_i \cdot w_{\mathbf{R}_i} + (\mathbf{q}_O)_i \cdot w_{\mathbf{O}_i} + (\mathbf{q}_M)_i \cdot w_{\mathbf{L}_i} w_{\mathbf{R}_i} + (\mathbf{q}_C)_i = 0. \tag{14}$$

For the wire values $\mathbf{w} \in \mathbb{F}^m$, we call $(w_j)_{j \in [l]}$ *public input* and $(w_j)_{j \in [l+1,m]}$ *private input*, respectively. We say $\mathcal{C}$ *is prepared for* l *public inputs* if for each $i \in [l]$ we define $\mathbf{L}_i = i$, $(\mathbf{q}_L)_i = 1, (\mathbf{q}_R)_i = (\mathbf{q}_M)_i = (\mathbf{q}_R)_i = (\mathbf{q}_C)_i = 0$, i.e., each gate $i \in [l]$ is dedicated for the input wire $j = i \in [l]$ of $\mathbf{w}$. Then the constraint for an input gate $i \in [l]$ can be satisfied by subtracting w_j from the above equation. Accordingly, we can define relation wrt $\mathcal{C}$.

Definition 16 (PLONK indexed relation). *The indexed relation* $\mathcal{R}_{\mathsf{PLONK}}$ *is the set of all triples*

$$((\mathbb{F}, n, m, l, \mathbf{L}, \mathbf{R}, \mathbf{O}, \mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O, \mathbf{q}_M, \mathbf{q}_C), (w_j)_{j \in [l]}, (w_j)_{j \in [l+1,m]})$$

such that

$$\forall i \in [l], (\mathbf{q}_L)_i \cdot w_{\mathbf{L}_i} + (\mathbf{q}_R)_i \cdot w_{\mathbf{R}_i} + (\mathbf{q}_O)_i \cdot w_{\mathbf{O}_i} + (\mathbf{q}_M)_i \cdot w_{\mathbf{L}_i} w_{\mathbf{R}_i} + (\mathbf{q}_C)_i - w_i = 0$$
$$\forall i \in [l+1, n], (\mathbf{q}_L)_i \cdot w_{\mathbf{L}_i} + (\mathbf{q}_R)_i \cdot w_{\mathbf{R}_i} + (\mathbf{q}_O)_i \cdot w_{\mathbf{O}_i} + (\mathbf{q}_M)_i \cdot w_{\mathbf{L}_i} w_{\mathbf{R}_i} + (\mathbf{q}_C)_i = 0$$

B.2 Lagrange basis.

Let q be a characteristic of $\mathbb{F}$ and n be such that $q = 1 \mod n$. Then $\mathbb{F}^*$ contains a multiplicative subgroup $\mathbb{H} = \{\zeta, \zeta^2, \ldots, \zeta^n\}$ generated by an nth primitive root of unity $\zeta \in \mathbb{F}^*$. It follows that an associated vanishing polynomial $v_{\mathbb{H}}(X) = X^n - 1$ splits completely in $\mathbb{F}[X]$, i.e., $X^n - 1 = \prod_{i=1}^{n}(X - \zeta^i)$. In PLONK the Lagrange basis $L_x(X)$ for $x \in \mathbb{H}$ is defined as follows.

$$L_x(X) \coloneqq \frac{c_x(X^n - 1)}{X - x}$$

By definition it is easy to check that $L_x(y) = 0$ for all $y \in \mathbb{H} \setminus \{x\}$. We show $L_x(x) = 1$ so that $L_x(X)$ is indeed a Lagrange basis. First, due to the Euclidean division of polynomials $X^n - 1$ can be rewritten as

$$X^n - 1 = (X - x) \cdot \left(\sum_{i=0}^{n-1} x^i X^{n-1-i} \right) + (x^n - 1).$$

As x has order n the remainder $x^n - 1$ vanishes. Therefore, we get

$$L_x(X) = c_x \cdot \left(\sum_{i=0}^{n-1} x^i X^{n-1-i} \right).$$

Defining $c_x = (nx^{n-1})^{-1}$ we have $L_x(x) = 1$. In what follows we write $L_i(X) \coloneqq L_x(X)$ for $x = \zeta^i$.

B.3 Checking gate-by-gate constraints.

When working over a multiplicative subgroup $\mathbb{H} \subset \mathbb{F}^*$, the selector vectors define polynomials in $\mathbb{F}_{<n}[X]$ via interpolation:

$$q_L(X) = \sum_{i \in [n]} (\mathbf{q}_L)_i \cdot L_i(X) \qquad q_R(X) = \sum_{i \in [n]} (\mathbf{q}_R)_i \cdot L_i(X) \qquad q_O(X) = \sum_{i \in [n]} (\mathbf{q}_O)_i \cdot L_i(X) \tag{15}$$

$$q_M(X) = \sum_{i \in [n]} (\mathbf{q}_M)_i \cdot L_i(X) \qquad q_C(X) = \sum_{i \in [n]} (\mathbf{q}_C)_i \cdot L_i(X) \tag{16}$$

So $q_L(\zeta^i) = (\mathbf{q}_L)_i, q_R(\zeta^i) = (\mathbf{q}_R)_i$ and so on. Let us define the following polynomials.

$$f_{\mathsf{pub}}(X) = \sum_{i \in [l]} -w_i L_i(X) \quad f_L(X) = \sum_{i \in [n]} w_{\mathbf{L}_i} L_i(X) \quad f_R(X) = \sum_{i \in [n]} w_{\mathbf{R}_i} L_i(X) \quad f_O(X) = \sum_{i \in [n]} w_{\mathbf{O}_i} L_i(X) \tag{17}$$

Then the gate-by-gate constraint of Eq. (14) can be checked if the polynomial

$$F_{\mathcal{C}}(X) \coloneqq q_L(X) f_L(X) + q_R f_R(X) + q_O(X) f_O(X) + q_M(X) f_L(X) f_R(X) + q_C(X) + f_{\mathsf{pub}}(X) \tag{18}$$

vanishes at ζ^i for all $i \in [n]$.

B.4 Checking copy constraints.

Notice that the above ranged polynomial evaluations are only individually checking constraint for each gate, but do not care about how different gates are associated with each other. To define a relation equivalent to $\mathcal{R}_{\mathsf{PLONK}}$, we need to enforce the *copy constraints* on evaluations of witness polynomials f_L, f_R, f_O. Let us first define two useful notions.

Definition 17 (Extended permutation across multiple polynomials). *Let $f_1, \ldots, f_c, h_1, \ldots, h_c \in \mathbb{F}[X]$ and $\sigma : [cn] \to [cn]$ be a permutation. Define the sequences of polynomial evaluations $f_{(1)}, \ldots, f_{(cn)}, h_{(1)}, \ldots h_{(cn)}$ over $\mathbb{H} = \{\zeta, \ldots, \zeta^n\}$ as follows:*

$$f_{((j-1)n+i)} \coloneqq f_j(\zeta^i) \text{ and } h_{((j-1)n+i)} \coloneqq h_j(\zeta^i)$$

for each $i \in [n]$ and $j \in [c]$. Then we write $(h_1, \ldots, h_c) = \sigma(f_1, \ldots, f_c)$ if $h_{(i)} = f_{(\sigma(i))}$ for all $i \in [cn]$.

Definition 18 (Copy-satisfy). *Let $\mathcal{T} = \{T_1, \ldots, T_m\}$ be a partition of $[cn]$. We say that $f_1, \ldots, f_c \in \mathbb{F}[X]$ copy-satisfy $\mathcal{T}$ if $f_{(i)} = f_{(i')}$ for all distinct pairs $i, i' \in T_j$ and for all $j \in [m]$.*

Lemma 6 ([KPV19][GWC19]). *Let $\mathcal{T} = \{T_1, \ldots, T_m\}$ be a partition of $[cn]$. Suppose a permutation $\sigma : [cn] \to [cn]$ is defined such that its restriction $\sigma|_{T_j}$ contains a cycle going over all elements in T_j for all $j \in [m]$. Then $f_1, \ldots, f_c \in \mathbb{F}[X]$ copy-satisfy $\mathcal{T}$ if and only if $(f_1, \ldots, f_c) = \sigma(f_1, \ldots, f_c)$*

In a concrete instantiation of PLONK, we set $c = 3$ and consider an extended permutation across $f_1 = f_L, f_2 = f_R$, and $f_3 = f_O$. Let $\mathcal{T}_{\mathcal{C}} = \{T_1, \ldots, T_m\}$ be a partition of $[3n]$ such that $T_j = \{i \in [3n] : \mathbf{v}_i = j\}$, i.e., a set T_j contains positions in $\mathbf{v} := \mathbf{L}||\mathbf{R}||\mathbf{O} \in [m]^{3n}$ that point to w_j. Then, by defining a permutation $\sigma : [3n] \to [3n]$ such that it satisfies a condition for Lemma 6, it suffices to provide some *permutation argument* that proves $(f_L, f_R, f_O) = \sigma(f_L, f_R, f_O)$, in order to show (f_L, f_R, f_O) copy-satisfy $\mathcal{T}_{\mathcal{C}}$.

B.5 Putting together.

We are now set out to define an alternative formulation of the indexed relation $\mathcal{R}_{\mathsf{PLONK}}$, which is in fact the one used by the resulting protocol of [GWC19]. Let f_L, f_R, f_O be polynomials as defined above and let us define a slightly redundant form of statement and witness. Namely, we define $((\mathsf{w}_i)_{i\in[l]}, (\mathsf{w}_i)_{i\in[l+1,3n]})$ such that

$$f_{\mathsf{pub}}(X) = \sum_{i\in[l]} -\mathsf{w}_i L_i(X) \tag{19}$$

$$f_L(X) = \sum_{i\in[n]} \mathsf{w}_i L_i(X) \tag{20}$$

$$f_R(X) = \sum_{i\in[n]} \mathsf{w}_{n+i} L_i(X) \tag{21}$$

$$f_O(X) = \sum_{i\in[n]} \mathsf{w}_{2n+i} L_i(X) \tag{22}$$

so $\mathsf{w}_i = f_L(\zeta^i)$, $\mathsf{w}_{n+i} = f_R(\zeta^i)$, and $\mathsf{w}_{2n+i} = f_O(\zeta^i)$.

Definition 19 (PLONK indexed relation (alternative formulation)). *The indexed relation $\mathcal{R}'_{\mathsf{PLONK}}$ is the set of all triples*

$$((\mathbb{F}, n, m, l, \mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O, \mathbf{q}_M, \mathbf{q}_C, \sigma, \mathcal{T}_{\mathcal{C}}), (\mathsf{w}_i)_{i\in[l]}, (\mathsf{w}_i)_{i\in[l+1,3n]})$$

such that

$$\begin{aligned}
\forall i \in [n]: &\quad \mathsf{w}_i = \mathsf{w}_{\sigma(i)} \\
\forall i \in [l]: &\quad (\mathbf{q}_L)_i \cdot \mathsf{w}_i + (\mathbf{q}_R)_i \cdot \mathsf{w}_{n+i} + (\mathbf{q}_O)_i \cdot \mathsf{w}_{2n+i} + (\mathbf{q}_M)_i \mathsf{w}_i \mathsf{w}_{n+i} + (\mathbf{q}_C)_i - \mathsf{w}_i = 0 \\
\forall i \in [l+1, n]: &\quad (\mathbf{q}_L)_i \cdot \mathsf{w}_i + (\mathbf{q}_R)_i \cdot \mathsf{w}_{n+i} + (\mathbf{q}_O)_i \cdot \mathsf{w}_{2n+i} + (\mathbf{q}_M)_i \mathsf{w}_i \mathsf{w}_{n+i} + (\mathbf{q}_C)_i = 0
\end{aligned}$$

By construction, given an instance of $\mathcal{R}_{\mathsf{PLONK}}$ one can clearly define constraint systems $\mathcal{C}$ as well as $(\mathsf{i}', \mathsf{x}', \mathsf{w}') \in \mathcal{R}'_{\mathsf{PLONK}}$. It turns out that the converse is also true. We sketch how to efficiently construct a tuple $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{PLONK}}$, given $(\mathsf{i}', \mathsf{x}', \mathsf{w}') \in \mathcal{R}'_{\mathsf{PLONK}}$.

From i' we can clearly determine the constraint systems $\mathcal{C} = (n, m, \mathbf{L}, \mathbf{R}, \mathbf{O}, \mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O, \mathbf{q}_M, \mathbf{q}_C)$ as well as $\mathsf{i} = (\mathbb{F}, l, \mathcal{C})$. Since $\mathsf{w}_i = \mathsf{w}_{\sigma(i)}$ holds and due to the way $\sigma : [3n] \to [3n]$ is defined (i.e., such that its restriction $\sigma|_{T_j}$ contains a cycle going over all elements in T_j for all $j \in [m]$), we have that $\mathsf{w}_i = \mathsf{w}_{i'}$ for all distinct pairs $i, i' \in T_j$ and for each $j \in [m]$. Now for each $j \in [m]$ we define $w_j = \mathsf{w}_i$ for some $i \in T_j$. Recall that, by construction of $\mathcal{T}_{\mathcal{C}}$, for each $j \in [m]$ we also have $j = \mathbf{v}_i = \mathbf{v}_{i'}$ for each $i, i' \in T_j$, where $\mathbf{v} = \mathbf{L}||\mathbf{R}||\mathbf{O} \in [m]^{3n}$. So overall $w_j = w_{\mathbf{v}_i} = \mathsf{w}_i = w_{\mathbf{v}_{i'}} = \mathsf{w}_{i'}$. This indicates that

$$\begin{aligned}
&(\mathbf{q}_L)_i \cdot w_{\mathbf{v}_i} + (\mathbf{q}_R)_i \cdot w_{\mathbf{v}_{n+i}} + (\mathbf{q}_O)_i \cdot w_{\mathbf{v}_{2n+i}} + (\mathbf{q}_M)_i w_{\mathbf{v}_i} w_{\mathbf{v}_{n+i}} + (\mathbf{q}_C)_i - w_i = 0 \text{ for } i \in [l] \\
&(\mathbf{q}_L)_i \cdot w_{\mathbf{v}_i} + (\mathbf{q}_R)_i \cdot w_{\mathbf{v}_{n+i}} + (\mathbf{q}_O)_i \cdot w_{\mathbf{v}_{2n+i}} + (\mathbf{q}_M)_i w_{\mathbf{v}_i} w_{\mathbf{v}_{n+i}} + (\mathbf{q}_C)_i = 0 \text{ for } i \in [l+1, n]
\end{aligned}$$

or in other words,

$$\begin{aligned}
&(\mathbf{q}_L)_i \cdot w_{\mathbf{L}_i} + (\mathbf{q}_R)_i \cdot w_{\mathbf{R}_i} + (\mathbf{q}_O)_i \cdot w_{\mathbf{O}_i} + (\mathbf{q}_M)_i w_{\mathbf{L}_i} w_{\mathbf{R}_i} + (\mathbf{q}_C)_i - w_i = 0 \text{ for } i \in [l] \\
&(\mathbf{q}_L)_i \cdot w_{\mathbf{L}_i} + (\mathbf{q}_R)_i \cdot w_{\mathbf{R}_i} + (\mathbf{q}_O)_i \cdot w_{\mathbf{O}_i} + (\mathbf{q}_M)_i w_{\mathbf{L}_i} w_{\mathbf{R}_i} + (\mathbf{q}_C)_i = 0 \text{ for } i \in [l+1, n]
\end{aligned}$$

implying $(\mathsf{i}, (w_j)_{j\in[l]}, (w_j)_{j\in[l+1,m]}) \in \mathcal{R}_{\mathsf{PLONK}}$.

B.6 Extended Permutation Argument

To prove $(f_L, f_R, f_O) = \sigma(f_L, f_R, f_O)$, PLONK invokes an *extended permutation argument* subprotocol, which we recall in Fig. 12 in the form of AHP. Due to Lemma 5.3 of [GWC19], for any $f_L, f_R, f_O \in \mathbb{F}_{<D}$ and any permutation $\sigma : [3n] \to [3n]$ such that $D \geq n$, if $(f_L, f_R, f_O) \neq \sigma(f_L, f_R, f_O)$ then for any unbounded prover P, the probability that V accepts in the above protocol is negligible in the security parameter.

Protocol $\mathsf{AHP}_{\mathsf{PermArg}}$

Offline phase. The indexer I receives as input $\mathbb{F} \in \mathcal{F}$ and $\mathsf{i} = (\mathbb{F}, n, m, l, \mathbf{L}, \mathbf{R}, \mathbf{O}, \mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O, \mathbf{q}_M, \mathbf{q}_C)$, and computes the permutation $\sigma : [3n] \to [3n]$. Then I generates the preprocessed polynomial oracles.

$$S_{L,\mathsf{ID}} = \sum_{i\in[n]} i \cdot L_i(X) \qquad S_{L,\sigma} = \sum_{i\in[n]} \sigma(i) \cdot L_i(X)$$
$$S_{R,\mathsf{ID}} = \sum_{i\in[n]} (n+i) \cdot L_i(X) \qquad S_{R,\sigma} = \sum_{i\in[n]} \sigma(n+i) \cdot L_i(X)$$
$$S_{O,\mathsf{ID}} = \sum_{i\in[n]} (2n+i) \cdot L_i(X) \qquad S_{O,\sigma} = \sum_{i\in[n]} \sigma(2n+i) \cdot L_i(X)$$

Input. Polynomial oracles $f_L, f_R, f_O \in \mathbb{F}_{<n}[X]$.

Online phase. Upon receiving random challenges $\beta, \gamma \in \mathbb{F}$ from V, P computes

$$h_{L,\mathsf{ID}} = f_L + \beta \cdot S_{L,\mathsf{ID}} + \gamma \qquad h_{L,\sigma} = f_L + \beta \cdot S_{L,\sigma} + \gamma \tag{23}$$
$$h_{R,\mathsf{ID}} = f_R + \beta \cdot S_{R,\mathsf{ID}} + \gamma \qquad h_{R,\sigma} = f_R + \beta \cdot S_{R,\sigma} + \gamma \tag{24}$$
$$h_{O,\mathsf{ID}} = f_O + \beta \cdot S_{O,\mathsf{ID}} + \gamma \qquad h_{O,\sigma} = f_O + \beta \cdot S_{O,\sigma} + \gamma \tag{25}$$
$$h_{\mathsf{ID}} = h_{L,\mathsf{ID}} \cdot h_{R,\mathsf{ID}} \cdot h_{O,\mathsf{ID}} \qquad h_\sigma = h_{L,\sigma} \cdot h_{R,\sigma} \cdot h_{O,\sigma} \tag{26}$$

Then P sends a permutation polynomial oracle:

$$s(X) = L_1(X) + \sum_{i\in[2,n]} \left(L_i(X) \cdot \prod_{1\leq j<i} \frac{h_{\mathsf{ID}}(\zeta^j)}{h_\sigma(\zeta^j)} \right). \tag{27}$$

Query phase. V queries all offline and online oracles with all points in $a \in \mathbb{H}$.

Online phase. V checks that the following polynomials vanish on $\mathbb{H}$.

$$F_1(X) = h_{\mathsf{ID}}(X)s(X) - h_\sigma(X)s(\zeta X)$$
$$F_2(X) = L_1(X)(s(X) - 1)$$

Fig. 12. Permutation argument subprotocol for $(f_L, f_R, f_O) = \sigma(f_L, f_R, f_O)$

B.7 PLONK AHP

Fig. 4 describes the underlying $\mathsf{AHP}_{\mathsf{PLONK}}$ implicit in the final AoK protocol of PLONK. Recall that the goal of PLONK is to verify (1) gate-by-gate constraints by checking $F_{\mathcal{C}}(X)$ vanishes on $\mathbb{H}$, and (2) copy constraints by checking $(f_L, f_R, f_O) = \sigma(f_L, f_R, f_O)$, as described in $\mathcal{R}'_{\mathsf{PLONK}}$ of Appendix B.5. Due to the permutation argument from Fig. 12 the second part amounts to checking that polynomials $F_1(X)$ and $F_2(X)$ vanish on $\mathbb{H}$. A naïve way to achieve these would be to let the verifier query polynomial oracles with every point in $\mathbb{H}$, which of course incurs $O(n)$ query complexity on verfier's side. This can be circumvented by replacing queries with divisibility check by *vanishing polynomial* $v_{\mathbb{H}}(X) = X^n - 1 = \prod_{i\in[n]}(X - \zeta^i)$, and by taking random challenge α to batch polynomials $F_{\mathcal{C}}, F_1$, and F_2 to be divided. From Lemma 4.5 and 4.7 of [GWC19] the $\mathsf{AHP}_{\mathsf{PLONK}}$ has knowledge soundness.[7]

[7] We remark that [GWC19] presents their protocol in a slightly different form called a *polynomial protocol*. The main difference with AHP is that it performs identity checks of polynomials, instead of evaluations of

B.8 Adding zero-knowledge

To achieve ZK the polynomials f_L, f_R, f_O, s carrying witness in Fig. 4 have to be slightly adjusted; since these are evaluated at a single point the prover adds random extra terms that lie outside of the degree bounds of the original polynomials. Concretely, now P commits to

$$\begin{aligned} f'_L(X) &= f_L(X) + (b_1 X + b_2) v_{\mathbb{H}}(X) \\ f'_R(X) &= f_R(X) + (b_3 X + b_4) v_{\mathbb{H}}(X) \\ f'_O(X) &= f_O(X) + (b_5 X + b_6) v_{\mathbb{H}}(X) \\ s'(X) &= s(X) + (b_7 X^2 + b_8 X + b_9) v_{\mathbb{H}}(X) \end{aligned}$$

where the so-called *blinding terms* b_i are randomly chosen from $\mathbb{F}$. The reason why three blinding terms are required for $s(X)$ is that it gets evaluated at two points z and ζz. Note that this change doesn't affect the correctness, since the additional terms are guaranteed to be divisible by $v_{\mathbb{H}}(X)$. In a similar fashion, the Marlin AHP in the next section can be also made zero-knowledge.

polynomials at random query points as in AHP. Deriving knowledge soundness of the latter formulation is straightforward due to the Schwartz–Zippel lemma.