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
emil@bennedsen.eu
[3] Indian Institute of Science, India
chaya@iisc.ac.in

Abstract. We advance the state-of-the art for zero-knowledge commit-and-prove SNARKs (CP-SNARKs). CP-SNARKs are an important class of SNARKs which, using commitments as "glue", allow to efficiently combine proof systems—e.g., general-purpose SNARKs (an efficient way to prove statements about circuits) and Σ-protocols (an efficient way to prove statements about group operations). Thus, CP-SNARKs allow to efficiently provide zero-knowledge proofs for composite statements such as $h = H(g^x)$ for some hash-function H.

Our main contribution is providing the first construction of CP-SNARKs where the proof size is succinct in the number of commitments.

We achieve our result by providing a general technique to compile Algebraic Holographic Proofs (AHP) (an underlying abstraction used in many modern SNARKs) with special "decomposition" properties into an efficient CP-SNARK. We then show that some of the most efficient AHP constructions—Marlin, PLONK, and Sonic—satisfy our compilation requirements.

Our resulting SNARKs achieve universal and updatable reference strings, which are highly desirable features as they greatly reduce the trust needed in the SNARK setup phase.

1 Introduction

Zero-knowledge (ZK) proofs and argument systems (ZK) [35] are one of the most fascinating concepts in modern cryptography, as they allow proving that a statement is valid without revealing any additional information as to why said statement is true. Even further, *Succinct Non-interactive ARguments of Knowledge* (zk-SNARKs), allow to do so in such a way that the size of the proof and the work the verifier needs to perform in order to check the proof is sublinear in the size of the statement. Today, zk-SNARKs are a fundamental building block in complex cryptographic systems such as e.g., Zcash [9], where succinct zero-knowledge proofs are used to provide integrity while maintaining privacy. In such applications, it is crucial that the verification time is minimal

* Full version available at [3].

(as every user in the system has to perform the verification) and that the proofs are short and non-interactive (as they need to be posted on the Blockchain).

In this work we focus on *commit-and-prove* SNARKs (CP-SNARKs) (introduced in [21]).

This is an important family of SNARKs in which the witness is committed using Pedersen commitments (the de-facto *lingua franca* of commitments). The presence of these commitments allow to "glue" together different proof systems. An important application of CP-SNARKs is proving composite statements using the most efficient tool for each part of the statement. Such modularity of the CP proof system enhances *interoperability* with other protocols specialized for efficiently proving certain *algebraic relations*: consider a composite computation that naturally presents different components, like an arithmetic 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, e.g., SNARKs for an arithmetic circuit and Σ-protocol for an algebraic relation. One of the simplest examples of such a statement is proving knowledge of the secret key corresponding to a Bitcoin address e.g., proving knowledge of some x such that $y = H(g^x)$ (without revealing g^x).

There are many other practical scenarios where the CP extension is useful, including, but not limited to, anonymous credentials [24, 29, 2], verifiable encryption [44], proof stitching [26, 47, 53, 52], and e-voting [44]. Given these various potential applications, a working group focused on CP-ZK has recently been launched as part of the ZKProof Standards [12].

Unfortunately, existing CP-SNARKs are not truly "succinct" since their proof size scales linearly with the number of commitments containing subvectors of the witness. In this work, we fill this gap in the literature and provide the first truly succinct CP-SNARK.

1.1 Applications

To further motivate the need for succinct CP-SNARKs, we now provide some example applications. In all these applications, the commitments to (subset of) the witness are part of the public statement and, in practice, often exist prior to the time we prove properties on them. Motivated by this, we do not count the commitments as part of the proof size in this work.

We denote by ℓ the number of individual commitments containing the partial witness vector.

1. *Anonymous and Delegated Credentials.* In 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. The main challenge is that the statement being considered is a composite statement containing both Boolean (hash function)

and algebraic (group operations) components, since either the message is hashed before being signed or one needs to prove properties on the signed message. Efficient NIZK for composite statements that use a zk-SNARK for the circuit part and Σ-protocols for the algebraic would yield a proof system that is more efficient for the prover.

Consider now the 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 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. If the resulting proof is succinct in ℓ, then this batching technique results in important savings. Note here that in this application it would not be feasible to commit to the credentials of all users in a single (vector) commitment, because the ℓ commitments to the credentials already exist and each single user should be able to verify that on their own.[4]

2. *Blockchains.* CP-SNARKs are useful in many Blockchain applications like *confidential* transactions [49] where range proofs are required on committed values, and in systems balancing privacy and accountability [28] where credentials are proven on committed values.

   An example Blockchain application where $\ell > 1$ and succinct CP-SNARKs are desirable is *proof of solvency*. In privacy-preserving proof of solvency [2], the number of commitments ℓ is typically large. This is because in proof of liabilities, each customer has to check that their own balance has been included in the total liabilities published by the exchange. This is done by having the exchange send the decommitment information to each customer privately. Thus, in this application too, using a single (vector) commitment is not a feasible solution. Since each customer's balance is private, there must be as many commitments as the number of customers instead of one vector commitment to all balances.

3. *Machine Learning.* Another example of an application that benefits from succinct CP-SNARKs is verifying integrity of Machine Learning (ML) models. A hospital owns sensitive patient data, and one wishes to construct a model by running a training algorithm on this sensitive data. The hospital does not wish to and/or legally cannot release the data; making it a challenge to check the integrity of the model. Such settings have been considered, for example, in [54]. One way to do this is to have the hospital use a zkSNARK

[4] The service can afford to wait for ℓ orders depending on the application, and the expected throughput and time-to-service of the application. As an example, the ID-Layer in Concordium [28] orders may be even serviced each epoch.

to prove that the model is the output obtained by training it on the sensitive data and that public commitments indeed open to the same sensitive data.

In practice, ML algorithms are run on data held by different entities (hospitals in the example above). Each of the ℓ entities publishes a commitment to their sensitive data. Now a single trusted party can perform training on the combined data, but has to prove integrity with respect to commitment of each individual entity. Succinct CP-SNARKs provide efficiency benefits also in this case.

1.2 Our Contributions

In this work we present the first CP-SNARKs whose proof size is succinct in the number of commitments to the partial witness vectors. To do so, we combine state-of-the-art SNARKs with state-of-the-art Σ-protocols, inheriting several important properties of the underlying tools which we use.

An important property of our resulting proof system is that it has *universal, updatable and linear-size reference string*: Since we are interested in practically efficient and succinct proof systems, our starting points are *preprocessing* SNARKs, in which some form of trusted setup (in the form of a *structured reference string* or SRS) is required. If the trusted setup is compromised, it becomes possible to break the soundness property of the proof system. However, using SNARKs with *universal and updatable setup* (as introduced in [38]) the trust in the setup phase can be reduced to a minimum, as this allows participants to dynamically update the SRS was proposed. Even though this does not completely remove the problem of trusted setup, the security now depends on at least one honest party deleting the contributed randomness. Moreover, the SRS is *universal* in the sense that it allows to prove statements about all circuits of some bounded size (as opposed to earlier systems in which a different SRS was needed for each circuit, thus increasing the need for trusted setups). Furthermore, the size of the setup will be linear in the size (or upper bound of) the circuit to be proven.

From a technical point of view, our contributions can be summarized as follows:

– *Compiler from AHP to CP-SNARK.* In Sec. 3 we present a compiler that takes an AHP (Algebraic Holographic Proof, the information-theoretic protocol underlying many existing zkSNARKs) and compiles it into a CP-SNARK. Our compiler is similar in spirit to compilers of [18, 25, 20] that convert information-theoretic protocols to succinct arguments, but it naturally allows efficient CP extensions because of our "decompose–and–link" paradigm outlined in Sec. 1.3. The main technical challenge in building this compiler is that existing SNARK constructions employ different ways to *encode* the witness into a polynomial, even though the underlying information-theoretic objects can be described in the language of AHP. This makes it hard to identify how to generically & succinctly link committed values to only a small fraction of the large witness vector used in SNARK. Yet, we are

able to abstract out a set of basic properties that AHPs and commitment schemes should satisfy, in order to apply the same paradigm. Thanks to our abstract approach, one does not need to examine the entire machinery of the AHP protocol; instead, it is sufficient to look at a few *witness-carrying polynomials* present in the AHP, check if they satisfy the properties required by our compiler theorem, and then focus on designing a sub-protocol performing a minimum set of tasks for "linking". We believe that our techniques are general enough to extend to future AHPs and commitment schemes.

– *Concrete instantiations.* We then apply our compiler to the AHPs of Marlin, PLONK and Sonic to obtain concrete CP-SNARKs.[5] 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 (Pedersen) 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 (zk-SNARK) worlds. 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 techniques of [15, 4] that achieve 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.

1.3 Technical Overview

Most recent constructions of updatable SRS zkSNARKs [18, 25, 32] 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. While several abstractions for this information-theoretic parts exist, it is folklore among researchers in this community that these formalizations are to some extent equivalent. In this paper, we rely on the formalization of (public-coin) *Algebraic Holographic Proofs (AHP)* of [25] and we cast the other SNARKs (PLONK [32] and Sonic [48]) in the same language.

[5] 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 (the first can outperform the second, which can outperform the third, which can in turn outperform the first). This is because they use different models of computations, and therefore it may be possible to prove some statements more efficiently with one system rather than the others.

Plain AHP-to-SNARK framework. In an AHP the prover P takes a statement x and a witness vector $\mathbf{w} = (w_1, \ldots, w_n)$ as inputs and sends some *oracle polynomials* to the verifier V in each round, who responds with a random challenge. In the query phase, V can query an oracle polynomial p with an *evaluation point* z to obtain $v = p(z)$. V can iterate this process for several different polynomials and evaluation points. Finally, V outputs a decision bit indicating "accept" or "reject", based on the result of the evaluation queries.

An AHP can be turned into an argument system by replacing the oracles and the query phase with a *polynomial commitment scheme (PCS)*. As proposed by [42], PCS can be succinctly instantiated by using the discrete log-based encoding of polynomial: $\mathsf{PC.Com}_{\mathsf{ck}}(p(X)) := g^{p_0 + p_1\chi + \ldots p_{n-1}\chi^{n-1}}$ with a commitment key $\mathsf{ck} = (g, g^{\chi}, \ldots, g^{\chi^{n-1}})$. Then upon receiving an evaluation point z, the prover responds with an *evaluation proof* to convince the verifier that evaluation $v = p(z)$ is done correctly.

Witness-carrying polynomials and CP extension. Typically, one or few oracles sent by an AHP prover are *witness-carrying polynomials (WCP)* [20], meaning that they encode the entire witness vector $\mathbf{w}$. For ease of exposition, we assume the AHP has a single WCP $w(X)$ here, but our abstract compiler works for AHP with multiple WCP as well. The encoding/decoding method differs depending on the protocol. For example, Sonic employs a simple *coefficient* encoding, therefore, decoding works by mapping the coefficients to a witness vector, i.e., $w(X) := \sum_i w_i X^i$; PLONK and Marlin use *interpolation*, and decoding works by evaluating WCP on some prescribed set, i.e., $w(X) := \sum_i w_i \cdot L_i(X)$, where $(L_i(X))_{i\in[n]}$ are the Lagrange polynomials associated with some set $\mathbb{H}$ of size n.

In our CP scenario, we additionally consider 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, and in some applications, these commitments already exist. For example, if a subvector of witness $(w_i)_{i\in I_{\mathsf{com}}}$ with $I_{\mathsf{com}} \subset [n]$ is committed in advance via vector Pedersen commitment, an argument system additionally takes $\hat{c} = \mathsf{AC.Com}_{\mathsf{ack}}((w_i)_{i\in I_{\mathsf{com}}}; r) := H^r \prod_{i\in I_{\mathsf{com}}} G_i^{w_i}$ as part of the statement, where $\mathsf{ack} := ((G_i)_{i\in I_{\mathsf{com}}}, H)$. The goal of CP extension is to guarantee consistency between what is committed to via PC and AC. To this end, it should suffice to provide a sub-protocol for relation

$$\mathcal{R} := \left\{ ((c, \hat{c}), (\mathbf{w}, r)) \ : \ c = \textstyle\prod_{i=1}^{n} g_i^{w_i} \ \wedge \ \hat{c} = H^r \prod_{i\in I_{\mathsf{com}}} G_i^{w_i} \right\}.$$

where $g_i = g^{\chi^{i-1}}$ or $g_i = g^{L_i(\chi)}$, depending on how the AHP under consideration encodes the witness into WCP.

A naïve approach would be to describe an arithmetic circuit for $\mathcal{R}$ and invoke another instance of SNARK. However, if the committing function of AC involves certain algebraic operations, e.g., group exponentiation or elliptic curve scalar multiplications as required in the Pedersen commitment, it would be very costly

for the prover to express them in a circuit[6]. This is where a Σ-protocol comes into play.

Decomposing WCP and linking with Σ-protocol. A simple Σ-protocol can be used for proving equality of Pedersen-committed messages. However, because naïve instantiation of such a protocol for $\mathcal{R}$ inevitably proves knowledge of the *entire vector* $\mathbf{w}$, it would incur $O(n)$ proof size and verification time, losing succinctness. Although it is possible to apply the compressed Σ-protocol theory [15, 4] to achieve $O(\log(n))$ proof size, if logarithmic proof size is acceptable, one could instead use Bulletproofs, which supports CP extensions with the Pedersen commitment by construction and already achieves $O(\log(n))$ proof size.

In fact, proving $\mathcal{R}$ turns out to be quite wasteful, since at the end of the day we only care about a small fraction of $\mathbf{w}$ that are committed beforehand. We circumvent the issue by additively *decomposing* the WCP $w(X)$ into two parts $w_{\mathsf{com}}(X)$ and $w_{\mathsf{mid}}(X)$, such that $w(X) = w_{\mathsf{com}}(X) + w_{\mathsf{mid}}(X)$, $w_{\mathsf{com}}(X)$ encodes the committed part of the witness $(w_i)_{i \in I_{\mathsf{com}}}$, and $w_{\mathsf{mid}}(X)$ contains the rest. In Sec. 3.2 we formally define this intuition. Accordingly, assuming *additively homomorphic* PCS (satisfied by KZG), one can also decompose a polynomial commitment c into c_{com} and c_{mid} such that $c = c_{\mathsf{com}} + c_{\mathsf{mid}} = \mathsf{PC.Com}_{\mathsf{ck}}(w_{\mathsf{mid}}) + \mathsf{PC.Com}_{\mathsf{ck}}(w_{\mathsf{com}})$. Now we only need to *link* c_{com} and $\hat{c}$; it suffices to cast c_{com} to the Σ-protocol for relation

$$\mathcal{R}' := \left\{ ((c_{\mathsf{com}}, \hat{c}), (\mathbf{w}, r)) \ : \ c = \prod_{i \in I_{\mathsf{com}}} g_i^{w_i} \ \wedge \ \hat{c} = H^r \prod_{i \in I_{\mathsf{com}}} G_i^{w_i} \right\}$$

which only incurs $O(\log(|I_{\mathsf{com}}|))$ proof size and verification time.

Proving "non-overlapping" decomposition. The above idea needs additional care in order to preserve knowledge soundness since it is not guaranteed that a cheating prover honestly decomposes WCP. For example, what if a prover crafted $\tilde{w}_{\mathsf{mid}}(X)$ such that it decodes to $\tilde{w}_{\mathsf{mid},i}$ for some $i \in I_{\mathsf{com}}$? In that case, the knowledge extractor for SNARK outputs $\tilde{w}_i = \tilde{w}_{\mathsf{com},i} + \tilde{w}_{\mathsf{mid},i}$ as one of the witness vector elements, whereas the Σ-protocol only proves that $\hat{c}$ contains $\tilde{w}_{\mathsf{com},i}$. This breaks consistency between the value in $\hat{c}$ and the actual witness used in SNARK. To fix this issue, we require a prover to show the decomposed WCPs are "non-overlapping", meaning that $w_{\mathsf{mid}}(X)$ *only* maps to $(w_i)_{i \notin I_{\mathsf{com}}}$.[7] In Sec. 5, 6, we present different ways to instantiate this additional check: for Sonic it amounts to perform a degree bound check for $w_{\mathsf{mid}}(X)$, while for PLONK and Marlin it suffices to verify $w_{\mathsf{mid}}(X)$ vanishes on certain evaluation points.

Compressing and aggregating many equality proofs. So far we have only considered a single auxiliary commitment $\hat{c}$. But clearly, as described earlier, we are interested in the case where the number ℓ of commitments is large and

[6] While there are approaches that mitigate this problem [43, 1, 22], they are curve-dependent—hindering generality and interoperability—and still relatively expensive (at 4-6 constraints per curve operation).

[7] While it is also necessary to prove $w_{\mathsf{com}}(X)$ only maps to $(w_i)_{i \in I_{\mathsf{com}}}$, this is trivially achieved by knowledge soundness of the Σ-protocol.

	$\lvert\pi\rvert$	Prove (time)	Verify (time)
This work	$O(\log(\ell \cdot d))$	$O(n + \ell \cdot d)$	$O(\ell \cdot d)$
Lunar [20]	$O(\ell)$	$O(n + \ell \cdot d)$	$O(\ell)$
LegoUAC [21]	$O\left(\ell \log^2(n)\right)$	$O(n) + \ell \cdot \tilde{O}(d)$	$O\left(\ell \log^2(n)\right)$

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 Sec. 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).

we want our proof to be succinct in ℓ. Naïvely, the above ideas can easily be generalized by invoking ℓ instances of the equality proof for $\mathcal{R}'$ with statement $(c_{\mathsf{com}}, \hat{c}_k)$ for $k \in [\ell]$. This in turn would incur in a multiplicative factor of $O(\ell)$ overhead in the proof size. In Sec. 4 we show how to amortize ℓ different protocol instances to achieve $O(\log(\ell d))$ proof size by adapting the amortization technique from [5], where d is a dimension of the vector committed to in each $\hat{c}_k$.

1.4 Related Work

Σ-protocols are proof systems that are efficient for proving algebraic statements about discrete logarithms, roots, or polynomial relationships among values [51, 39, 27, 19]. 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 [30]. A recursive argument for an inner product relation of committed values was presented in [15] and was subsequently improved in Bulletproofs [17]. These can be used to prove statements on algebraically committed inputs, and the proof can be made non-interactive using Fiat-Shamir. Even though proof sizes scale logarithmically, unfortunately, the verification time scales linearly with the size of the circuit. Recent work on compressed Σ-protocol theory [4] is a strengthening of Σ-protocols that compress the communication complexity from linear to logarithmic. The underlying pivot of the compressed protocol 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.

The seminal paper of [33] 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 [40, 36, 45] and led to several follow ups [13, 50, 10, 46, 11, 37] which have proofs that are very short and have fast verification time.

The first zk-SNARK with an updatable SRS was introduced by [38]. However, here the size of this universal updatable SRS is quadratic in the number of multiplication gates of the circuit representing the statement. In [48], the authors construct Sonic, the first zkSNARK that is universal and updatable with a linear-sized SRS. A different solution to SNARKs with universal and updatable SRS is to use a secure multi-party computation protocol (MPC) to conduct the setup [16], and as long as at least one party is honest, the setup remains secure.

Although several works on general-purpose CP-ZK exist in the literature, such as Geppetto [26], Cinderella [29], and [47], there are few examples 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 [24] 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 [41] for circuit-based statements with Σ-protocols for algebraic parts. However, this is inherently interactive which is inherited from the interactivity of [41] where the verifier uses private coins. In [8], the authors show how to extend the MPC-in-the-head techniques of ZKBoo [34] and ZKB++ [23] 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 [2], 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. Not only does their approach lead to more efficient anonymous credentials than Cinderella, but it also found new applications to the blockchain, such as proof-of-solvency. Our approach achieves better asymptotic efficiency as well as further generality compared to [2], which relies on naïve Σ-protocols and a specific QAP-based SNARK construction with non-updatable SRS.

The works most closely related to ours are LegoSNARK and Lunar. LegoSNARK [21] is a framework for CP-SNARKs that gives general composition tools to build new CP-SNARKs from proof gadgets in a modular way. The construction LegoUAC in [21] is a CP-SNARK with a universal and updatable SRS. Lunar [20] obtains CP-SNARKs with a universal and updatable SRS and presents proof systems for "linking" committed inputs to the polynomial commitments used in AHP-based arguments. Table 1 shows the efficiency comparison between our work, Lunar and LegoUAC. Note that Lunar constructions and ECLIPSE outperform each other in different settings. See also §1.2 of [20] for a technical comparison of Lunar and ECLIPSE.

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. Throughout, $\mathbb{G}$ denotes an Abelian group of prime order q. For vectors of generators $\boldsymbol{g} = (g_1, \ldots, g_d) \in \mathbb{G}^d$ and exponents $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{Z}_q^d$ we often write $\boldsymbol{g}^{\mathbf{x}} := \prod_{i=1}^{d} g_i^{x_i}$.

Definition 1 (Indexed relation [25]). *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* $|\mathsf{i}| \leq \mathsf{N}$*.*

A zero-knowledge proof (or argument)[8] 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 [25]). *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* $\mathcal{A}$*,*

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

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

In [3] we recall the standard definitions of knowledge soundness and zero knowledge. 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.

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

2.1 Algebraic Holographic Proofs

Below we recall the definition of AHP from Marlin.

Definition 3 (AHP [25]). *An* Algebraic Holographic Proof (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 outputs 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 [25] 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.

Generalization to multivariate polynomials. Even though the above formalization is tailored to univariate polynomial oracles, it is straightforward to generalize it to support multivariate, Laurent polynomials $p_{i,j} \in \mathbb{F}[X_1, X_1^{-1}, \ldots, X_m, X_m^{-1}]$. In that case, a query set Q consists of $((i,j),(z_1,\ldots,z_m))$ and is to be interpreted as "query $p_{i,j}$ at $(z_1,\ldots,z_m) \in \mathbb{F}^m$". Likewise, the polynomial commitment scheme definition can also be adapted to support multivariate polynomials as inputs. Our Theorem 1 in the next section holds under this generalization because the proof does not rely on whether polynomials are univariate or not. This is analogous to the compiler theorem of [25]. However, the generalization is only required for Sonic and PLONK, and Marlin only deals with univariate polynomials. Therefore, we focus on the univariate version in the main body for ease of exposition.

2.2 Polynomial Commitment

Polynomial commitment schemes were introduced by Kate–Zaverucha–Goldberg [42]. Below we recall the definition of standard polynomial commitment scheme. The definition is taken verbatim from Section 6.1 of [25].

Definition 4 (Polynomial Commitment Scheme). *A polynomial commitment scheme (PCS) 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 [42] and its variants described in Marlin and Sonic already satisfy.

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

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

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 $\mathbf{d}$,

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

In [3] we recall formal security requirements for PCS: *extractability*, *binding*, and *hiding*. On a high-level, the extractability property assures that the prover actually *knows* the polynomial p committed to c whenever the verifier accepts an evaluation proof π.

2.2.1 The KZG scheme. Below we recall the polynomial commitment scheme due to Kate–Zaverucha–Goldberg [42], denoted by $\mathsf{PC}_{\mathsf{KZG}}$. The scheme is proven extractable under the strong Diffie–Hellman (SDH) assumption in the *algebraic group model (AGM)* [31], polynomial binding under the discrete-log assumption, and perfectly hiding [25, 42]. 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 [25] 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 $\omega \in \mathbb{F}_{\leq D}[X]$ 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 that turns AHPs to commit-and-prove zkSNARKs.

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 $d \in \mathbb{N}$, where $d \in \mathsf{poly}(\lambda)$, 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 some randomness r. In our concrete instantiations, we use the Pedersen vector commitment scheme 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

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

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 [20]: straight-line extractability for AHP. We note that our definition is in the AHP model, while that in [20] is for Polynomially Holographic Proofs. The reason why we explicitly define witness-carrying polynomials (WCPs) 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 admissible* P^*, *every field* $\mathbb{F} \in \mathcal{F}$, *every index* i *and instance* x,

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

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 (WCPs) *of* AHP. *If all WCPs 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 WCPs 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 1. 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 WCPs. 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). *Consider an* AHP *for relation* $\mathcal{R}$ *with* S*-straight-line extractor* WitExt*. We say that* WitExt *performs* unique extraction*, if for any honest prover* P *and 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* $\langle \mathsf{P}(\mathsf{i}, \mathsf{w}), \mathsf{V}^{\mathsf{I}(\mathbb{F},\mathsf{i})}\rangle(\mathbb{F}, \mathsf{x})$*.*

Definition 9 (Decomposable witness-carrying polynomials). *Consider an* AHP *for relation* $\mathcal{R}$ *with* W*-straight-line extractor* WitExt*. Let* $(p_{i,j}(X))_{(i,j)\in W}$ *be WCPs of* AHP*. We say that polynomials* $(p_{i,j}(X))_{(i,j)\in W}$ *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*

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

Remark 2. If the above decomposition function is invoked on WCPs, one can observe that witness extraction/decoding is also additively homomorphic on such honest inputs, i.e.,

$$\begin{aligned}&\mathsf{WitExt}((p_{i,j}(X))_{(i,j)\in W}) = \mathsf{WitExt}((p^{(1)}_{i,j}(X))_{(i,j)\in W} + (p^{(2)}_{i,j}(X))_{(i,j)\in W})\\ =&\mathsf{WitExt}((p^{(1)}_{i,j}(X))_{(i,j)\in W}) + \mathsf{WitExt}((p^{(2)}_{i,j}(X))_{(i,j)\in W}).\end{aligned}$$

3.3 Our compiler

In order to prove the relation $\mathcal{R}_{\mathsf{com}}$ above, our compiler will use a commit-and-prove NIZKAoK subprotocol for following relation. Although the abstract relation $\mathcal{R}_{\mathtt{lnk}}$ looks cumbersome for the sake of generality, the actual instantiation of $\mathsf{CP}_{\mathtt{lnk}}$ will be rather simple: it can be achieved by "linking" committed witness sub-vectors and proving "non-overlapping" decomposition as outlined in 1.3. See Figs. 3 and 4 for concrete examples.

Definition 10 (Commitment-linking relation). *Fix an* AHP *for relation* $\mathcal{R}$ *with* W*-straight-line extractor* WitExt *and with witness carrying polynomials* $(p_{i,j}(X))_{(i,j)\in W}$*, a polynomial commitment scheme* PC*, and an auxiliary com-*

mitment scheme AC. *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. On a high-level the relation guarantees "the prover knows polynomials committed via PC, such that their sum correctly decodes to the partial witnesses committed via AC". 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 also 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 [25], Lunar [20] and DARK [18]. One important difference is the use of polynomial decomposition where the prover will commit separately to each of the "parts" of the WCPs. 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* WitExt*, 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 an additively homomorphic polynomial commitment over $\mathcal{F}$ with binding and extractability;*
- $\mathsf{CP}_{\mathtt{lnk}} = (\mathcal{I}_{\mathtt{lnk}},\mathcal{P}_{\mathtt{lnk}},\mathcal{V}_{\mathtt{lnk}})$ *is (preprocessing) 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* PC *is hiding,* $\mathsf{CP}_{\mathtt{lnk}}$ *is zero-knowledge, and* AHP *is zero-knowledge as defined in Definition 3, then* ARG *is also zero-knowledge.*

Moreover, if we additionally assume that the 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 commit to the witness carrying polynomials using a special version of KZG (see for example the input format of commitments in Figure 3) 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., [25],[32],[48], and we are still able to satisfy the security requirements of the general compiler this way.

Proof sketch Full proofs are deferred to [3]. Completeness follows from inspection. In particular, we benefit from a combination of homomorphism of PC and additive, non-overlapping decomposition of WCP. For zero-knowledge, we construct a simulator $\mathcal{S}$ 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. For knowledge soundness, we construct the extractor $\mathcal{E}_{\mathsf{ARG}}$ that works as follows: (1) Extract the polynomials from the polynomial commitments sent at each round through the extractor $\mathcal{E}_{\mathsf{PC}}$ for the polynomial commitments; (2) From these, for each $(i,j) \in W$ reconstruct the WCP 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 WCP $(p^{\mathsf{com}}_{i,j}(X), p^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}$ such that $p_{i,j}(X) = p^{\mathsf{com}}_{i,j}(X) + p^{\mathsf{mid}}_{i,j}(X)$, by invoking the extractor $\mathcal{E}_{\mathtt{lnk}}$ for $\mathsf{CP}_{\mathtt{lnk}}$; (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 $((\tilde{\mathsf{w}}_i)_{i\in[n]}, (\tilde{r}_k)_{k\in[\ell]})$.

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 and Cramer [4] and Attema, Cramer and Fehr [5]. 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

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

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 $(c_{0,j})_{j=1}^{\mathsf{s}(0)} = \mathsf{PC.Com}_{\mathsf{ck}}((p_{0,j})_{j=1}^{\mathsf{s}(0)})$. It also invokes the indexer of $\mathsf{CP}_{\mathtt{lnk}}$: $(\mathsf{ipk}_{\mathtt{lnk}}, \mathsf{ivk}_{\mathtt{lnk}}) \leftarrow \mathcal{I}^{\mathsf{srs}}_{\mathtt{lnk}}(I_{\mathsf{com}}, (I_k)_{k\in[\ell]})$. The indexer outputs $\mathsf{ipk} := (\mathsf{ck}, \mathsf{i}, (p_{0,j})_{j=1}^{\mathsf{s}(0)}, (c_{0,j})_{j=1}^{\mathsf{s}(0)}, \mathsf{ipk}_{\mathtt{lnk}})$ and $\mathsf{ivk} := (\mathsf{rk}, (c_{0,j})_{j=1}^{\mathsf{s}(0)}, \mathsf{ivk}_{\mathtt{lnk}})$.

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 $(\mathsf{ivk}, (\mathsf{x}, (\hat{c}_k)_{k\in[\ell]}))$.

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
$$\mathcal{P}_{\mathtt{lnk}}(\mathsf{ipk}_{\mathtt{lnk}}, ((\hat{c}_k)_{k\in[\ell]}, \mathbf{v}, Q, (c^{\mathsf{com}}_{i,j}(X), c^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}), ((p^{\mathsf{com}}_{i,j}(X), p^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}, (\omega^{\mathsf{com}}_{i,j}(X), \omega^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}, (r_k)_{k\in[\ell]}))$$
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., $\mathcal{V}_{\mathtt{lnk}}(\mathsf{ivk}_{\mathtt{lnk}}, ((\hat{c}_k)_{k\in[\ell]}, \mathbf{v}, Q, (c^{\mathsf{com}}_{i,j}(X), c^{\mathsf{mid}}_{i,j}(X))_{(i,j)\in W}), \pi_{\mathtt{lnk}}) = 1$;

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

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

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 $O(d)$. 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 Bulletproofs [17] and batch verification of EdDSA signatures. Note that the protocol below can be seen as a verifier-optimized version of the technique described by Attema–Cramer–Fehr [5, §3.4]. For completeness, in [3] we include a version derived by invoking their amortization of multiple group homomorphisms in a black-box way. The advantage of our AmComEq over AmComEq′ 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 [30]. 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. Security proof is deferred to [3].

Theorem 2. 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* $\mathbb{Z}_q$*.*

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* [4, 5, 7, 6] comes into play. That is, the last move of $\mathsf{AmComEq}$ can invoke another protocol $\mathsf{CompDLEq}$ of proof size $O(\log(\ell d))$, for the relation

$$\mathcal{R}_{\mathsf{DLEq}} = \left\{ ((\boldsymbol{g}, \tilde{\mathbf{G}}, \ell d), (Y, \hat{Y}), \mathbf{z}) \ : \ Y = \boldsymbol{g}^{\mathbf{z}}, \hat{Y} = \tilde{\mathbf{G}}^{\mathbf{z}} \right\}. \tag{2}$$

The protocol $\mathsf{CompDLEq}$ for $\mathcal{R}_{\mathsf{DLEq}}$ is described in [3]. From [4, Theorem 2] we immediately get the following result.

Corollary 1. *Let* $\mathsf{CompAmComEq}$ *be a protocol identical to* $\mathsf{AmComEq}$*, except that its last move is replaced by* $\mathsf{CompDLEq}$*.* $\mathsf{CompAmComEq}$ *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* $\mathbb{Z}_q$*.*

5 Instantiation with PLONK

In this section we apply our $\mathsf{ECLIPSE}$ 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 [3].

5.1 PLONK AHP

We consider an 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 $\mathbb{H} = \left\{\zeta, \zeta^2, \ldots, \zeta^n\right\} \subset \mathbb{F}^*$ 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 $L_i(\zeta^j) = 0$ for $j \neq i$.

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

$$f_L(X) = \sum_{i\in[n]} \mathsf{w}_i L_i(X) \quad f_R(X) = \sum_{i\in[n]} \mathsf{w}_{n+i} L_i(X) \quad f_O(X) = \sum_{i\in[n]} \mathsf{w}_{2n+i} L_i(X) \tag{3}$$

To achieve zero-knowledge these polynomials are masked by polynomials $(\rho_{L,1}X + \rho_{L,2})v_{\mathbb{H}}(X)$, $(\rho_{R,1}X + \rho_{R,2})v_{\mathbb{H}}(X)$ and $(\rho_{O,1}X + \rho_{O,2})v_{\mathbb{H}}(X)$ 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 $\mathcal{R}'_{\mathsf{PLONK}}$. The auxiliary commitment scheme AC is instantiated with vector Pedersen commitment and its key ack consists of randomly chosen generators of $\mathbb{G}$ with unknown relative discrete logarithms: $\mathbf{G} = (G_1, \ldots, G_d)$ and H.

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, every d values are batched into a single commitment, that is, every vector compound of d wires w_i, for $i \in I_k = [l+1+d(k-1), l+dk]$, is committed to in the kth auxiliary commitment $\hat{C}_k = \mathbf{G}^{(\mathsf{w}_i)_{i \in I_k}} H^{r_k}$ 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

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

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 $\mathsf{n}_{\mathsf{w}} = 3$ masked WCPs (f_L, f_R, f_O) and $I_{\mathsf{com}} \subset [n]$, parses f_L as $\sum_{i \in [n]} \mathsf{w}_i L_i(X) + (\rho_1 X + \rho_2) v_{\mathbb{H}}(X)$, and decompose them as follows.

$$f_{L,\mathsf{com}}(X) := \sum_{i \in I_{\mathsf{com}}} \mathsf{w}_i L_i(X) + (\lambda_{\mathsf{com},1} X + \lambda_{\mathsf{com},2}) v_{\mathbb{H}}(X) \qquad f_{R,\mathsf{com}}(X) := 0 \qquad f_{O,\mathsf{com}}(X) := 0$$

$$f_{L,\mathsf{mid}}(X) := \sum_{i \in [n] \setminus I_{\mathsf{com}}} \mathsf{w}_i L_i(X) + (\lambda_{\mathsf{mid},1} X + \lambda_{\mathsf{mid},2}) v_{\mathbb{H}}(X) \quad f_{R,\mathsf{mid}}(X) := f_R(X) \quad f_{O,\mathsf{mid}}(X) := f_O(X)$$

where $\lambda_{\mathsf{com},i}$'s are randomly chosen and $\lambda_{\mathsf{mid},i} := \rho_i - \lambda_{\mathsf{com},i}$. Clearly, the decomposition is additive, degree-preserving, and non-overlapping.

– WitExt takes WCPs (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 masking WCPs, but without hiding commitment[9], we use de-randomized version of $\mathsf{PC}_{\mathsf{KZG}}.\mathsf{Com}_{\mathsf{ck}}$ (see Sect. 2.2.1) that takes polynomial $f \in \mathbb{F}_{\leq D}[X]$ and outputs $[f(\chi)]_1$. Hence the polynomial commitment key is $\mathsf{ck} = \mathsf{pp} = (g, g^{\chi}, \ldots, g^{\chi^D})$. Clearly, this is an additively homomorphic commitment scheme. Its binding and extractability were formally shown in Appendix B-D of [25]. As mentioned in [32] 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. 3. Since WCPs 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 [14]. 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. 3) is an argument of knowledge. Assuming zero knowledge of Fiat–Shamir-transformed* $\mathsf{CompAmComEq}$*, the protocol* $\mathsf{CP}_{\mathtt{lnk}}$ *is zero-knowledge in the SRS model.*

Proof is deferred to [3].

[9] More formally, if the underlying AHP is $(\mathsf{b}+1, \mathsf{C})$-zero knowledge, where b is the maximum number of queries made by the verifier to polynomials, one can retain ZK of the resulting SNARK by compiling AHP via PCS with *somewhat hiding* security, a weaker notion of hiding [20]. Because the deterministic KZG is already somewhat hiding and every WCP in $\mathsf{AHP}_{\mathsf{PLONK}}$ is queried once, it suffices to add $v_{\mathbb{H}}$ multiplied by a masking polynomial of degree 1 to tolerate 2 openings (i.e., one evaluation and one commitment).

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

Indexing $\mathcal{I}^{\mathsf{srs}}_{\mathsf{lnk}}(I_{\mathsf{com}}, (I_k)_{k\in[\ell]})$ precomputes $[v_{\mathbb{H}_{\mathsf{com}}}(\chi)]_2$ such that $v_{\mathbb{H}_{\mathsf{com}}}(X) = \prod_{a\in\mathbb{H}_{\mathsf{com}}}(X-a)$ and $\mathbb{H}_{\mathsf{com}} = \{\zeta^i : i \in I_{\mathsf{com}}\} \subset \mathbb{H}$, obtains generators $g_i := [L_i(\chi)]_1$ for $i \in I_{\mathsf{com}}$, $\boldsymbol{g} := (g_i)_{i\in I_{\mathsf{com}}}$, $h_1 = [\chi v_{\mathbb{H}}(\chi)]_1$, $h_2 = [v_{\mathbb{H}}(\chi)]_1$, $\mathbf{G}$ and H by accessing srs. It outputs $(\mathsf{ipk}_{\mathsf{lnk}}, \mathsf{ivk}_{\mathsf{lnk}})$ such that

$$\mathsf{ipk}_{\mathsf{lnk}} = (\mathsf{pp}, v_{\mathbb{H}_{\mathsf{com}}}(X), \boldsymbol{g}, h_1, h_2, \mathbf{G}, H) \quad \text{and} \quad \mathsf{ivk}_{\mathsf{lnk}} = ([v_{\mathbb{H}_{\mathsf{com}}}(\chi)]_2, \boldsymbol{g}, h_1, h_2, \mathbf{G}, H).$$

Input. $\mathcal{P}_{\mathsf{lnk}}$ (resp. $\mathcal{V}_{\mathsf{lnk}}$) receives $\mathsf{ipk}_{\mathsf{lnk}}$ (resp. $\mathsf{ivk}_{\mathsf{lnk}}$). The statement $((\hat{C}_k)_{k\in[\ell]}, (C_{L,\mathsf{com}}, C_{L,\mathsf{mid}}))$ is a common input. The $\mathcal{P}_{\mathsf{lnk}}$ has as input witness $(f_{L,\mathsf{com}}(X), f_{L,\mathsf{mid}}(X), (r_k)_{k\in[\ell]})$ such that $\hat{C}_k = \mathbf{G}^{(\mathsf{w}_i)_{i\in I_k}} H^{r_k}$, $C_{L,\mathsf{com}} = [f_{L,\mathsf{com}}(\chi)]_1$, $C_{L,\mathsf{mid}} = [f_{L,\mathsf{mid}}(\chi)]_1$, $f_{L,\mathsf{com}}(X) = \sum_{i\in I_{\mathsf{com}}} \mathsf{w}_i L_i(X) + (\lambda_{\mathsf{com},1}X + \lambda_{\mathsf{com},2})v_{\mathbb{H}}(X)$, and $f_{L,\mathsf{mid}}(X) = \sum_{i\in[n]\setminus I_{\mathsf{com}}} \mathsf{w}_i L_i(X) + (\lambda_{\mathsf{mid},1}X + \lambda_{\mathsf{mid},2})v_{\mathbb{H}}(X)$.

Prove.

- Compute a proof π_{ComEq} of the following statement.

$$\mathsf{CompAmComEq}:\mathsf{PK}\left\{((\mathsf{w}_i)_{i\in I_{\mathsf{com}}}, (r_k)_{k\in[\ell]}, \lambda_{\mathsf{com},1}, \lambda_{\mathsf{com},2}) : \begin{array}{c}\hat{C}_k = \mathbf{G}^{(\mathsf{w}_i)_{i\in I_k}} H^{r_k} \wedge \\ C_{L,\mathsf{com}} = \boldsymbol{g}^{(\mathsf{w}_i)_{i\in I_{\mathsf{com}}}} h_1^{\lambda_{\mathsf{com},1}} h_2^{\lambda_{\mathsf{com},2}}\end{array}\right\}$$

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

Verify. Given π_{lnk}, verify π_{ComEq} and check that $f_{L,\mathsf{mid}}$ vanishes on $\mathbb{H}_{\mathsf{com}}$: $e(C_{L,\mathsf{mid}}, h) \stackrel{?}{=} e(\Pi, [v_{\mathbb{H}_{\mathsf{com}}}(\chi)]_2)$.

Fig. 3. Commitment-linking protocol for PLONK

6 Instantiation with Marlin

In this section we apply our compiler to Marlin. As in the previous section, we first identify WCPs and how it encodes the witness vector in AHP. More detailed preliminaries are provided in [3].

6.1 Marlin AHP

Notations. 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 $\mathbb{S}$. We denote by $\mathbb{F}^{\mathbb{S}}$ the set of vectors indexed by elements in a finite set $\mathbb{S}$. 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, $\hat{f}(a) = f(a)$ for all $a \in \mathbb{S}$. In particular, the polynomial $\hat{f}$ can be expressed as a linear combination

$$\hat{f}(X) = \sum_{a\in\mathbb{S}} f(a) \cdot L_{a,\mathbb{S}}(X)$$

where $\{L_{a,\mathbb{S}}(X)\}_{a\in\mathbb{S}}$ are the *Lagrange basis polynomials* of degree less than $|\mathbb{S}|$ such that $L_{a,\mathbb{S}}(a) = 1$ and $L_{a,\mathbb{S}}(a') = 1$ for $a' \in \mathbb{S} \setminus \{a\}$.

Constraint systems. Unlike PLONK, Marlin's AHP is for R1CS (Rank-1 constraint satisfiability) indexed relation defined by the set of tuples $(\mathsf{i}, \mathsf{x}, \mathsf{w}) = \big((\mathbb{F}, \mathbb{H}, \mathbb{K}, A, B, C), x, w\big)$, where $\mathbb{F}$ is a finite field, $\mathbb{H}$ and $\mathbb{K}$ are subsets of $\mathbb{F}$, such that $n = |\mathbb{H}|$ and $m = |\mathbb{K}|$, A, B, C are $\mathbb{H} \times \mathbb{H}$ matrices over $\mathbb{F}$ with $|\mathbb{K}| \geq \max\{\|A\|, \|B\|, \|C\|\}$, and $z := (x, w)$ is a vector in $\mathbb{F}^{\mathbb{H}}$ such that $Az \circ Bz = Cz$.

Following [25], we assume efficiently computable bijections $\phi_{\mathbb{H}} : \mathbb{H} \to [n]$ and $\phi_{\mathbb{K}} : \mathbb{K} \to [m]$, and denote the first l elements in $\mathbb{H}$ and the remaining elements, via sets $\mathbb{H}[\leq l] \coloneqq \{a \in \mathbb{H} : 1 \leq \phi_{\mathbb{H}}(a) \leq l\}$ and $\mathbb{H}[> l] \coloneqq \{a \in \mathbb{H} : l < \phi_{\mathbb{H}}(a) \leq n\}$ respectively. We then denote the first part of the vector z as the public component $x \in \mathbb{F}^{\mathbb{H}[\leq l]}$ and the second part as witness component $w \in \mathbb{F}^{\mathbb{H}[>l]}$.

WCP. In $\mathsf{AHP}_{\mathsf{Marlin}}$, the prover P receives as input the instance $x \in \mathbb{F}^{\mathbb{H}[\leq l]}$, a witness $w \in \mathbb{F}^{\mathbb{H}[>l]}$. The verifier V receives as input x, and obtains oracle access to the nine polynomials output at the end of the preprocessing phase.

Let $\hat{x}(X) \in \mathbb{F}_{<l}[X]$ and $\hat{w}(X) \in \mathbb{F}_{\leq n-l}[X]$ be polynomials that agree with the instance x on $\mathbb{H}[\leq l]$, and with the shifted witness on $\mathbb{H}[> l]$ respectively. Concretely, these polynomials are defined as follows:

$$\hat{x}(X) := \sum_{a \in \mathbb{H}[\leq l]} x(a) \cdot L_{a,\mathbb{H}[\leq l]}(X)$$

$$\hat{w}(X) := \sum_{a \in \mathbb{H}[>l]} \left(\frac{w(a) - \hat{x}(a)}{v_{\mathbb{H}[\leq l]}(a)} \right) \cdot L_{a,\mathbb{H}[>l]}(X) + \rho \cdot v_{\mathbb{H}[>l]}(X)$$

where the second term of $\hat{w}$ is added to retain zero-knowledge when the number of evaluation queries to $\hat{w}$ is 1 (which is the case in Marlin AHP) and ρ is sampled uniformly at random from $\mathbb{F}$. Let $z := (x, w)$ denote the full assignment. Then the polynomial $\hat{z}(X) := \hat{w}(X) \cdot v_{\mathbb{H}[\leq l]}(X) + \hat{x}(X)$ agrees with z on $\mathbb{H}$.

6.2 CP-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. For convenience we define the following subsets: $\mathbb{H}_{\mathsf{pub}} := \mathbb{H}[\leq l], \mathbb{H}_{\mathsf{com}} := \mathbb{H}[> l, \leq l + d\ell], \mathbb{H}_{\mathsf{mid}} := \mathbb{H}[> l + d\ell]$, assuming that $w(a)$ for $a \in \mathbb{H}_{\mathsf{com}}$ 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 $w(a)$, for $a \in \mathbb{H}_{\mathsf{com},k} = \mathbb{H}[> l + d(k-1), \leq l + dk]$, is committed to in the kth auxiliary commitment $\hat{C}_k = \mathbf{G}^{(w(a))_{a \in \mathbb{H}_{\mathsf{com},k}}} H^{r_k}$ for $k \in [\ell]$. Then we have $\mathbb{H}_{\mathsf{com}} = \bigcup_{k \in [\ell]} \mathbb{H}_{\mathsf{com},k}$.

Definition 12 (CP-Marlin indexed relation). *The indexed relation $\mathcal{R}_{\mathsf{CP\text{-}Marlin}}$ is the set of all triples*

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

where $\mathbb{F}$ is a finite field, $\mathbb{H}$ and $\mathbb{K}$ are subsets of $\mathbb{F}$, such that $n = |\mathbb{H}|$ and $m = |\mathbb{K}|$, A, B, C are $\mathbb{H} \times \mathbb{H}$ matrices over $\mathbb{F}$ with $|\mathbb{K}| \geq \max\{\|A\|, \|B\|, \|C\|\}$, and $z := (x, w)$ is a vector in $\mathbb{F}^{\mathbb{H}}$ such that

$$Az \circ Bz = Cz \text{ and } \forall k \in [\ell], \hat{C}_k = \mathsf{AC.Commit}_{\mathsf{ack}}((w(a))_{a \in \mathbb{H}_{\mathsf{com},k}}; r_k)$$

Applying our compiler. We now show that $\mathsf{AHP}_{\mathsf{Marlin}}$ and the polynomial commitment scheme $\mathsf{PC}_{\mathsf{KZG}}$ [42] meet the requirements of Theorem 1.

- Unique witness extraction: WitExt takes $\hat{w}(X)$, evaluates $\hat{w}(X)$ on every $a \in \mathbb{H}[> l]$, multiplies the results by $v_{\mathbb{H}[\leq l]}(a)$, and add $\hat{x}(a)$ to constructs a vector of values $w \in \mathbb{F}^{\mathbb{H}[>l]}$. It is easy to see that WitExt satisfies unique extraction (Definition 8).
- Decomposable WCP: Decomp takes $\hat{w}(X)$ and $\mathbb{H}_{\mathsf{com}}$, and outputs $\hat{w}_{\mathsf{com}}$ and $\hat{w}_{\mathsf{mid}}$ of degree at most $n - l$ as follows:

$$\hat{w}_{\mathsf{com}}(X) := \sum_{a \in \mathbb{H}_{\mathsf{com}}} \left(\frac{w(a) - \hat{x}(a)}{v_{\mathbb{H}[\leq l]}(a)} \right) \cdot L_{a,\mathbb{H}[>l]}(X) + \lambda_{\mathsf{com}} \cdot v_{\mathbb{H}[>l]}(X)$$

$$\hat{w}_{\mathsf{mid}}(X) := \sum_{a \in \mathbb{H}_{\mathsf{mid}}} \left(\frac{w(a) - \hat{x}(a)}{v_{\mathbb{H}[\leq l]}(a)} \right) \cdot L_{a,\mathbb{H}[>l]}(X) + \lambda_{\mathsf{mid}} \cdot v_{\mathbb{H}[>l]}(X)$$

  where λ_{com} was sampled from $\mathbb{F}$ uniformly at random and $\lambda_{\mathsf{mid}} := \rho - \lambda_{\mathsf{com}}$. Clearly, the decomposition is additive, degree-preserving and non-overlapping.
- Marlin compiles $\mathsf{AHP}_{\mathsf{Marlin}}$ using the plain KZG polynomial commitment except that degrees of hiding polynomials are minimized. That is, to commit to the WCP $\mathsf{PC}_{\mathsf{KZG}}.\mathsf{Com}_{\mathsf{ck}}$ takes $\hat{w}(X)$ and $\omega(X) := \omega_0 + \omega_1 X$ as input and outputs $[\hat{w}(\chi) + \gamma\omega(\chi)]_1$, where $\omega_0, \omega_1 \in \mathbb{F}$ are randomly sampled masking coefficients. As mentioned in §9.2 of [25] 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. In order to construct our commitment-linking protocol for Marlin, we modify how hiding is achieved. Specifically, we now mask the two decomposed WCPs independently as follows: commitment to $\hat{w}_{\mathsf{com}}(X)$ is masked by a random polynomial $\omega_{\mathsf{com}}(X) := \omega_{\mathsf{com},0} + \omega_{\mathsf{com},1} X$ and $\hat{w}_{\mathsf{mid}}(X)$ is masked by a random polynomial $\omega_{\mathsf{mid}}(X)$ that vanishes on $\mathbb{H}_{\mathsf{com}}$; $\omega_{\mathsf{mid}}(X) := (\omega_{\mathsf{mid},0} + \omega_{\mathsf{mid},1} X) v_{\mathbb{H}_{\mathsf{com}}}(X)$. Note that, for $\hat{w}_{\mathsf{mid}}$, we do not apply Marlin's optimization of minimising the degree.

  Following PLONK and Lunar, one may alternatively compile $\mathsf{AHP}_{\mathsf{Marlin}}$ with the deterministic KZG by increasing the degree of masking factor to 1 (i.e., $\rho_1 X + \rho_2$) to hide one evaluation and the commitment. In this way, decomposition of WCPs as well as $\mathsf{CP}_{\mathtt{lnk}}$ can be done as in $\mathsf{CP\text{-}PLONK}$ and the number of SRS elements does not grow due to the CP extension.

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

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

Indexing $\mathcal{I}^{\mathsf{srs}}_{\mathtt{lnk}}(I_{\mathsf{com}}, (I_k)_{k\in[\ell]})$ precomputes $[v_{\mathbb{H}_{\mathsf{com}}}(\chi)]_2$ such that $v_{\mathbb{H}_{\mathsf{com}}}(X) = \prod_{a\in\mathbb{H}_{\mathsf{com}}}(X-a)$, obtains generators $g_a := [L_{a,\mathbb{H}_{[>l]}}(\chi)/v_{\mathbb{H}_{[\le l]}}(a)]_1$ for $a \in \mathbb{H}_{\mathsf{com}}$, $\boldsymbol{g} := (g_a)_{a\in\mathbb{H}_{\mathsf{com}}}$, $h_1 := [v_{\mathbb{H}_{[>l]}}(\chi)]_1$, $h_2 := [\gamma]_1$, $h_3 := [\gamma\chi]_1$, $\mathbf{G}$ and H by accessing srs. It outputs $(\mathsf{ipk}_{\mathtt{lnk}}, \mathsf{ivk}_{\mathtt{lnk}})$ such that

$$\mathsf{ipk}_{\mathtt{lnk}} = (\mathsf{pp}, v_{\mathbb{H}_{\mathsf{com}}}(X), \boldsymbol{g}, h_1, h_2, h_3, \mathbf{G}, H) \quad \text{and} \quad \mathsf{ivk}_{\mathtt{lnk}} = ([v_{\mathbb{H}_{\mathsf{com}}}(\chi)]_2, \boldsymbol{g}, h_1, h_2, h_3, \mathbf{G}, H).$$

Input. $\mathcal{P}_{\mathtt{lnk}}$ (resp. $\mathcal{V}_{\mathtt{lnk}}$) receives $\mathsf{ipk}_{\mathtt{lnk}}$ (resp. $\mathsf{ivk}_{\mathtt{lnk}}$). The statement $((\hat{C}_k)_{k\in[\ell]}, (C_{\mathsf{com}}, C_{\mathsf{mid}}))$ is a common input. The $\mathcal{P}_{\mathtt{lnk}}$ has as input witness $(\hat{w}_{\mathsf{com}}(X), \hat{w}_{\mathsf{mid}}(X), (r_k)_{k\in[\ell]})$ such that $\hat{C}_k = \mathbf{G}^{(w(a))_{a\in\mathbb{H}_{\mathsf{com}},k}} H^{r_k}$, $C_{\mathsf{com}} = [\hat{w}_{\mathsf{com}}(\chi) + \gamma\omega_{\mathsf{com}}(\chi)]_1$, $C_{\mathsf{mid}} = [\hat{w}_{\mathsf{mid}}(\chi) + \gamma\omega_{\mathsf{mid}}(\chi)]_1$, and

$$\hat{w}_{\mathsf{com}}(X) = \sum_{a\in\mathbb{H}_{\mathsf{com}}} \left(\frac{w(a)-\hat{x}(a)}{v_{\mathbb{H}_{[\le l]}}(a)}\right) \cdot L_{a,\mathbb{H}_{[>l]}}(X) + \lambda_{\mathsf{com}} \cdot v_{\mathbb{H}_{[>l]}}(X)$$

$$\hat{w}_{\mathsf{mid}}(X) = \sum_{a\in\mathbb{H}_{\mathsf{mid}}} \left(\frac{w(a)-\hat{x}(a)}{v_{\mathbb{H}_{[\le l]}}(a)}\right) \cdot L_{a,\mathbb{H}_{[>l]}}(X) + \lambda_{\mathsf{mid}} \cdot v_{\mathbb{H}_{[>l]}}(X)$$

Prove.

- Compute a proof π_{ComEq} of the following statement where $\bar{C}_{\mathsf{com}} := C_{\mathsf{com}} \cdot \boldsymbol{g}^{(\hat{x}(a))_{a\in\mathbb{H}_{\mathsf{com}}}}$.

$$\mathsf{PK}\left\{ \begin{matrix}((w(a))_{a\in\mathbb{H}_{\mathsf{com}}}, (r_k)_{k\in[\ell]},\\ \lambda_{\mathsf{com}}, \omega_{\mathsf{com},0}, \omega_{\mathsf{com},1})\end{matrix} : \begin{matrix}\hat{C}_k = \mathbf{G}^{(w(a))_{a\in\mathbb{H}_{\mathsf{com}},k}} H^{r_k}\\ \wedge\ \bar{C}_{\mathsf{com}} = \boldsymbol{g}^{(w(a))_{a\in\mathbb{H}_{\mathsf{com}}}} h_1^{\lambda_{\mathsf{com}}} h_2^{\omega_{\mathsf{com},0}} h_3^{\omega_{\mathsf{com},1}}\end{matrix} \right\}$$

- Compute evaluation proof $\Pi = [W_1 + \gamma W_2(\chi)]_1$, where $W_1(X) = \frac{\hat{w}_{\mathsf{mid}}(X)}{v_{\mathbb{H}_{\mathsf{com}}}(X)}, W_2(X) = \frac{\omega_{\mathsf{mid}}(X)}{v_{\mathbb{H}_{\mathsf{com}}}(X)}$. Set $\pi_{\mathtt{lnk}} = (\Pi, \pi_{\mathsf{ComEq}})$. Note that since $\omega_{\mathsf{mid}}(X)$ vanishes on $\mathbb{H}_{\mathsf{com}}$, is divisible by $v_{\mathbb{H}_{\mathsf{com}}}$, and therefore W_2 is a polynomial.

Verify. Given $\pi_{\mathtt{lnk}}$, verify π_{ComEq}, and check that $\hat{w}_{\mathsf{mid}}$ vanishes on $\mathbb{H}_{\mathsf{com}}$: $e(C_{\mathsf{mid}}, h) \stackrel{?}{=} e(\Pi, [v_{\mathbb{H}_{\mathsf{com}}}(\chi)]_2)$.

Fig. 4. Commitment-linking protocol for Marlin

Lemma 2. *Assuming extractability of* PC_{KZG} *and argument of knowledge of* CompAmComEq*, the protocol* $\mathsf{CP}_{\mathtt{lnk}}$ *(Fig. 4) is an argument of knowledge. Assuming zero knowledge of Fiat–Shamir-transformed* CompAmComEq*, the protocol* $\mathsf{CP}_{\mathtt{lnk}}$ *is zero-knowledge in the SRS model.*

Proof is deferred to [3].

Acknowledgments

The authors are grateful for Sean Bowe, Ben Fisch, Ariel Gabizon, and Mary Maller for clarifying the details of their works. We thank the anonymous reviewers of PKC 2022 for helpful comments. This work has been 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).

References

1. What is Jubjub? https://z.cash/technology/jubjub

2. Agrawal, S., Ganesh, C., Mohassel, P.: Non-interactive zero-knowledge proofs for composite statements. In: CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 643–673. Springer, Heidelberg
3. Aranha, D.F., Bennedsen, E.M., Campanelli, M., Ganesh, C., Orlandi, C., Takahashi, A.: Eclipse: Enhanced compiling method for pedersen-committed zksnark engines. Cryptology ePrint Archive, Report 2021/934
4. Attema, T., Cramer, R.: Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In: CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 513–543. Springer, Heidelberg
5. Attema, T., Cramer, R., Fehr, S.: Compressing proofs of k-out-of-n partial knowledge. Cryptology ePrint Archive, Report 2020/753
6. Attema, T., Cramer, R., Kohl, L.: A compressed σ-protocol theory for lattices. Cryptology ePrint Archive, Report 2021/307
7. Attema, T., Cramer, R., Rambaud, M.: Compressed σ-protocols for bilinear group arithmetic circuits and applications. Cryptology ePrint Archive, Report 2020/1447
8. Backes, M., Hanzlik, L., Herzberg, A., Kate, A., Pryvalov, I.: Efficient non-interactive zero-knowledge proofs in cross-domains without trusted setup. In: PKC 2019, Part I. LNCS, vol. 11442, pp. 286–313. Springer, Heidelberg
9. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE Computer Society Press
10. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: Verifying program executions succinctly and in zero knowledge. In: CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 90–108. Springer, Heidelberg
11. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct non-interactive zero knowledge for a von neumann architecture. In: USENIX Security 2014. pp. 781–796. USENIX Association
12. Benarroch, D., Campanelli, M., Fiore, D., Kim, J., Lee, J., Oh, H., Querol, A.: Proposal: Commit-and-prove zero-knowledge proof systems and extensions. 4th ZKProof Workshop
13. Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct non-interactive arguments via linear interactive proofs. In: TCC 2013. LNCS, vol. 7785, pp. 315–333. Springer, Heidelberg
14. Boneh, D., Drake, J., Fisch, B., Gabizon, A.: Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Report 2020/081
15. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg
16. Bowe, S., Gabizon, A., Miers, I.: Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050
17. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society Press
18. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 677–706. Springer, Heidelberg
19. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups (extended abstract). In: CRYPTO'97. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg

20. Campanelli, M., Faonio, A., Fiore, D., Querol, A., Rodríguez, H.: Lunar: a toolbox for more efficient universal and updatable zksnarks and commit-and-prove extensions. Cryptology ePrint Archive, Report 2020/1069
21. Campanelli, M., Fiore, D., Querol, A.: LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In: ACM CCS 2019. pp. 2075–2092. ACM Press
22. Campanelli, M., Hall-Andersen, M.: Veksel: Simple, efficient, anonymous payments with large anonymity sets from well-studied assumptions. Cryptology ePrint Archive, Report 2020/1069
23. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: ACM CCS 2017. pp. 1825–1842. ACM Press
24. Chase, M., Ganesh, C., Mohassel, P.: Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In: CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 499–530. Springer, Heidelberg
25. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.P.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 738–768. Springer, Heidelberg
26. Costello, C., Fournet, C., Howell, J., Kohlweiss, M., Kreuter, B., Naehrig, M., Parno, B., Zahur, S.: Geppetto: Versatile verifiable computation. In: 2015 IEEE Symposium on Security and Privacy. pp. 253–270. IEEE Computer Society Press
27. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: CRYPTO'94. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg
28. Damgård, I., Ganesh, C., Khoshakhlagh, H., Orlandi, C., Siniscalchi, L.: Balancing privacy and accountability in blockchain identity management. In: CT-RSA 2021. LNCS, vol. 12704, pp. 552–576. Springer, Heidelberg
29. Delignat-Lavaud, A., Fournet, C., Kohlweiss, M., Parno, B.: Cinderella: Turning shabby X.509 certificates into elegant anonymous credentials with the magic of verifiable computation. In: 2016 IEEE Symposium on Security and Privacy. pp. 235–254. IEEE Computer Society Press
30. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg
31. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Heidelberg
32. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953
33. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg
34. Giacomelli, I., Madsen, J., Orlandi, C.: ZKBoo: Faster zero-knowledge for Boolean circuits. In: USENIX Security 2016. pp. 1069–1083. USENIX Association
35. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th ACM STOC. pp. 291–304. ACM Press
36. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg
37. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg

38. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. In: CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 698–728. Springer, Heidelberg
39. Guillou, L.C., Quisquater, J.J.: A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In: EUROCRYPT'88. LNCS, vol. 330, pp. 123–128. Springer, Heidelberg
40. Ishai, Y., Kushilevitz, E., Ostrovsky, R.: Efficient arguments without short pcps. In: Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07). pp. 278–291. IEEE
41. Jawurek, M., Kerschbaum, F., Orlandi, C.: Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In: ACM CCS 2013. pp. 955–966. ACM Press
42. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg
43. Kosba, A., Zhao, Z., Miller, A., Qian, Y., Chan, H., Papamanthou, C., Pass, R., shelat, a., Shi, E.: How to use SNARKs in universally composable protocols. Cryptology ePrint Archive, Report 2015/1093
44. Lee, J., Choi, J., Kim, J., Oh, H.: Saver: Snark-friendly, additively-homomorphic, and verifiable encryption and decryption with rerandomization. Cryptology ePrint Archive, Report 2019/1270
45. Lipmaa, H.: Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In: TCC 2012. LNCS, vol. 7194, pp. 169–189. Springer, Heidelberg
46. Lipmaa, H.: Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In: ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 41–60. Springer, Heidelberg
47. Lipmaa, H.: Prover-efficient commit-and-prove zero-knowledge SNARKs. In: AFRICACRYPT 16. LNCS, vol. 9646, pp. 185–206. Springer, Heidelberg
48. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: ACM CCS 2019. pp. 2111–2128. ACM Press
49. Maxwell, G.: Confidential transactions. URL: https://people. xiph. org/greg/confidential values. txt
50. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy. pp. 238–252. IEEE Computer Society Press
51. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: CRYPTO'89. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg
52. Setty, S.: Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In: CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 704–737. Springer, Heidelberg
53. Wahby, R.S., Tzialla, I., shelat, a., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: 2018 IEEE Symposium on Security and Privacy. pp. 926–943. IEEE Computer Society Press
54. Wu, H., Zheng, W., Chiesa, A., Popa, R.A., Stoica, I.: DIZK: A distributed zero knowledge proof system. In: USENIX Security 2018. pp. 675–692. USENIX Association