Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS

Alessandro Chiesa[1], Yuncong Hu[1], Mary Maller[2], Pratyush Mishra[1], Noah Vesely[1] and Nicholas Ward[1]

[1] UC Berkeley
{alexch,yuncong_hu,pratyush,psi,npward}@berkeley.edu
[2] Ethereum Foundation
mary.maller@ethereum.org

Abstract. We present a methodology to construct preprocessing zk-SNARKs where the structured reference string (SRS) is universal and updatable. This exploits a novel use of *holography* [Babai et al., STOC 1991], where fast verification is achieved provided the statement being checked is given in encoded form.

We use our methodology to obtain a preprocessing zkSNARK where the SRS has linear size and arguments have constant size. Our construction improves on Sonic [Maller et al., CCS 2019], the prior state of the art in this setting, in all efficiency parameters: proving is an order of magnitude faster and verification is thrice as fast, even with smaller SRS size and argument size. Our construction is most efficient when instantiated in the algebraic group model (also used by Sonic), but we also demonstrate how to realize it under concrete knowledge assumptions. We implement and evaluate our construction.

The core of our preprocessing zkSNARK is an efficient *algebraic holographic proof* (AHP) for rank-1 constraint satisfiability (R1CS) that achieves linear proof length and constant query complexity.

1 Introduction

Succinct non-interactive arguments (SNARGs) are efficient certificates of membership in non-deterministic languages. Recent years have seen a surge of interest in zero-knowledge SNARGs of knowledge (zkSNARKs), with researchers studying constructions under different cryptographic assumptions, improvements in asymptotic efficiency, concrete performance of implementations, and numerous applications. The focus of this paper is *SNARGs in the preprocessing setting*, a notion that we motivate next.

When is fast verification possible? The size of a SNARG must be, as a minimum condition, sublinear in the size of the non-deterministic witness, and often is required to be even smaller (e.g., logarithmic in the size of the non-deterministic computation). The time to verify a SNARG would be, ideally, as fast as reading the SNARG. *This is in general too much to hope for, however.* The verification procedure must also read the *description* of the computation, in order know what statement is being verified. While there are natural computations that have succinct descriptions (e.g., machine computations), in general the description of a computation could be as large as the computation itself, which means that the time to verify the SNARG could be asymptotically comparable to the size of the computation. This is unfortunate because there is a very useful class of computations for which we cannot expect fast verification: general circuit computations.

The preprocessing setting. An approach to avoid the above limitation is to design a verification procedure that has two phases: an offline phase that produces a short summary for a given circuit; and an online phase that uses this short summary to verify SNARGs that attest to the satisfiability of the circuit with different partial assignments to its input wires. Crucially, now the online phase could in principle be as fast as reading the SNARG (and the partial assignment), and thus sublinear in the circuit size. This goal was captured by *preprocessing SNARGs* [Gro10; Lip12; GGPR13; BCI+13], which have been studied in an influential line of works that has led to highly-efficient constructions that fulfill this goal (e.g., [Gro16]) and large-scale deployments in the real world that benefit from the online fast verification (e.g., [Zcash]).

The problem: circuit-specific SRS. The offline phase in efficient constructions of preprocessing SNARGS consists of sampling a structured reference string (SRS) that depends on the circuit that is being preprocessed. This implies that producing/validating proofs with respect to different circuits requires different SRSs. In many applications of interest, there is no single party that can be entrusted with sampling the SRS, and so real-world deployments have had to rely on cryptographic "ceremonies" [ZcashMPC] that use secure multi-party sampling protocols [BCG+15; BGG17; BGM17]. However, any modification in the circuit used in an application requires another cryptographic ceremony, which is unsustainable for many applications.

A solution: universal SRS. The above motivates preprocessing SNARGs where the SRS is *universal*, which means that the SRS supports any circuit up to a given size bound by enabling anyone, in an offline phase *after* the SRS is sampled, to publicly derive a circuit-specific SRS.[3] Known techniques to obtain a universal SRS from circuit-specific SRS introduce expensive overheads due to universal simulation [BCTV14a; BCTV14b]. Also, these techniques lead to universal SRSs that are not *updatable*, a property introduced in [GKM+18] that significantly simplifies cryptographic ceremonies. The recent work of Maller et al. [MBKM19] overcomes these shortcomings, obtaining the first efficient construction of a preprocessing SNARG with universal (and updatable) SRS. Even so, the construction in [MBKM19] is considerably more expensive than the state of the art for circuit-specific SRS [Gro16]. In this paper we ask: *can the efficiency gap between universal SRS and circuit-specific SRS be closed, or at least significantly reduced?*

[3] Even better than a universal SRS would be a URS (uniform reference string). However, achieving preprocessing SNARGs in the URS model with small argument size remains an open problem; see Section 1.2.

Concurrent work. A concurrent work [GWC19] studies the same question as this paper. See Section 1.2 for a brief discussion that compares the two works.

1.1 Our results

In this paper we present MARLIN, a new preprocessing zkSNARK with universal (and updatable) SRS that improves on the prior state of the art [MBKM19, Sonic] in essentially all relevant efficiency parameters.[4] In addition to reducing argument size by several group and field elements and reducing time complexity of the verifier by over $3\times$, our construction overcomes the main efficiency drawback of [MBKM19, Sonic]: the cost of producing proofs. Indeed, our construction improves time complexity of the prover by over $10\times$, achieving prover efficiency comparable to the case of preprocessing zkSNARKs with *circuit-specific* SRS. In Fig. 1 we provide a comparison of our construction and [MBKM19, Sonic], including argument sizes for two popular elliptic curves; the table also includes the state of the art for circuit-specific SRS. We have implemented MARLIN in a Rust library,[5] and report evaluation results in Fig. 2.

Our zkSNARK is the result of several contributions that we deem of independent interest, summarized below.

(1) A new methodology. We present a general methodology to construct preprocessing SNARGs (and also zkSNARKs) where the SRS is universal (and updatable). Our methodology produces succinct *interactive* arguments that can be made non-interactive via the Fiat–Shamir transformation [FS86], and so below we focus on *preprocessing arguments with universal and updatable SRS*.

Our key observation is that the ability to preprocess a circuit in an offline phase is closely related to constructing "holographic proofs" [BFLS91], which means that the verifier does not receive the circuit description as an input but, rather, makes a small number of queries to an encoding of it. These queries are in addition to queries that the verifier makes to proofs sent by the prover. Moreover, in this paper we focus on the setting where the encoding of the circuit description consists of low-degree polynomials and also where proofs are themselves low-degree polynomials — this can be viewed as a requirement that honest and malicious provers are "algebraic". We call these *algebraic holographic proofs* (AHPs); see Section 4 for definitions.

We present a transformation that "compiles" any public-coin AHP into a corresponding preprocessing argument with universal (and updatable) SRS by using suitable polynomial commitments.

Theorem 1. *There is an efficient transformation that combines any public-coin AHP for a relation $\mathcal{R}$ and an extractable polynomial commitment scheme to obtain*

[4] Maller et al. [MBKM19] discuss two variants of their protocol, a cheaper one for the "helped setting" and a costlier one for the "unhelped setting". The variant that is relevant to this paper is the latter one, because it is a preprocessing zkSNARK. (The former variant does not achieve succinct verification, and instead achieves a weaker guarantee that applies to proof batches.)

[5] `https://github.com/scipr-lab/marlin`

construction	argument size over BN-256 (bytes)	argument size over BLS12-381 (bytes)
Sonic [MBKM19]	1152	1472
MARLIN [this work]	1088	1296
Groth16 [Gro16]	128	192

zkSNARK construction		sizes			time complexity			
		$\|\mathsf{ipk}\|$	$\|\mathsf{ivk}\|$	$\|\pi\|$	generator	indexer	prover	verifier
Sonic [MBKM19]	$\mathbb{G}_1$	$8M$	—	20	8 f-MSM(M)	4 v-MSM($3m$)	273 v-MSM(m)	7 pairings
	$\mathbb{G}_2$	$8M$	3	—	8 f-MSM(M)	—	—	
	$\mathbb{F}_q$	—	—	16	—	$O(m \log m)$	$O(m \log m)$	$O(\|\mathbb{x}\| + \log m)$
MARLIN [this work]	$\mathbb{G}_1$	$6M$	2	13	1 f-MSM($6M$)	9 v-MSM(m)	21 v-MSM(m)	2 pairings
	$\mathbb{G}_2$	—	2	—	—	—	—	
	$\mathbb{F}_q$	—	—	21	—	$O(m \log m)$	$O(m \log m)$	$O(\|\mathbb{x}\| + \log m)$
Groth16 [Gro16]	$\mathbb{G}_1$	$4n$	$O(\|\mathbb{x}\|)$	2	4 f-MSM(n)	N/A	4 v-MSM(n)	1 v-MSM($\|\mathbb{x}\|$)
	$\mathbb{G}_2$	n	$O(1)$	1	1 f-MSM(n)		1 v-MSM(n)	3 pairings
	$\mathbb{F}_q$	—	—	—	$O(m + n \log n)$		$O(m + n \log n)$	—

n: number of multiplication gates in the circuit
m: total number of (addition or multiplication) gates in the circuit
M: maximum supported circuit size (= number of addition and multiplication gates)

Fig. 1: Comparison of two preprocessing zkSNARKs with universal (and updatable) SRS: the prior state of the art and our construction. We include the current state of the art for circuit-specific SRS (in gray), for reference. Here $\mathbb{G}_1/\mathbb{G}_2/\mathbb{F}_q$ denote the number of elements or operations over the respective group/field; also, f-MSM(m) and v-MSM(m) denote fixed-base and variable-base multi-scalar multiplications (MSM) each of size m, respectively. The number of pairings that we report for Sonic's verifier is lower than that reported in [MBKM19] because we account for standard batching techniques for pairing equations.

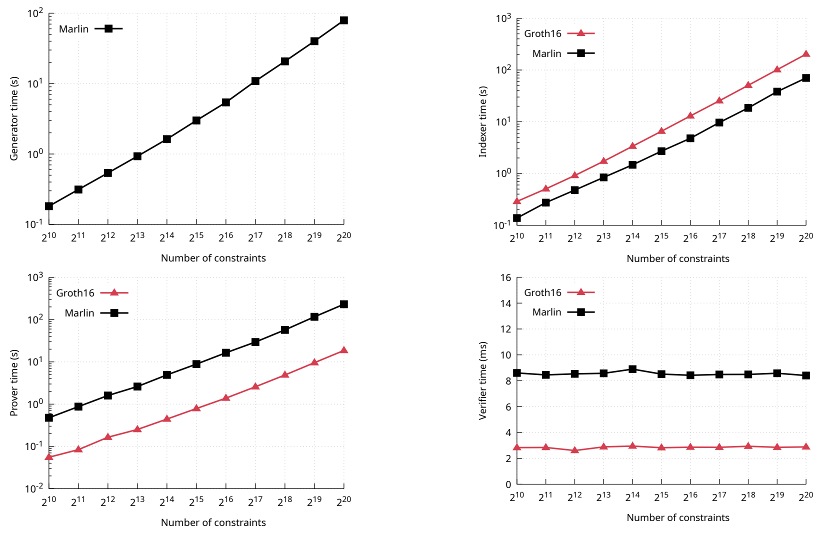

Fig. 2: Measured performance of MARLIN and [Gro16] over the BLS12-381 curve. We could not include measurements for [MBKM19, Sonic] because at the time of writing there is no working implementation of its unhelped variant.

a public-coin preprocessing argument with universal SRS for the relation $\mathcal{R}$. The transformation preserves zero knowledge and proof of knowledge of the underlying AHP. The SRS is updatable provided the SRS of the polynomial commitment scheme is.

The above transformation provides us with a *methodology* to construct preprocessing zkSNARKs with universal SRS (see Fig. 3). Namely, to improve the efficiency of preprocessing zkSNARKs with universal SRS it suffices to improve the efficiency of *simpler building blocks*: AHPs (an information-theoretic primitive) and polynomial commitments (a cryptographic primitive).[6]

The improvements achieved by our preprocessing zkSNARK (see Fig. 1) were obtained by following this methodology: we designed efficient constructions for each of these two building blocks (which we discuss shortly), combined them via Theorem 1, and then applied the Fiat–Shamir transformation [FS86].

Methodologies that combine information-theoretic probabilistic proofs and cryptographic tools have played a fundamental role in the construction of efficient argument systems. In the particular setting of preprocessing SNARGs, for example, the compiler introduced in [BCI+13] for *circuit-specific* SRS has paved the way towards current state-of-the-art constructions [Gro16], and also led to constructions that are plausibly post-quantum [BISW17; BISW18]. We believe that our methodology for universal SRS will also be useful in future work, and may lead to further efficiency improvements.

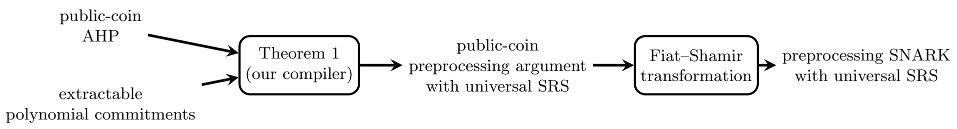
public-coin AHP
extractable polynomial commitments
Theorem 1 (our compiler)
public-coin preprocessing argument with universal SRS
Fiat–Shamir transformation
preprocessing SNARK with universal SRS

Fig. 3: Our methodology for constructing preprocessing SNARGs with universal SRS.

(2) An efficient AHP for R1CS. We design an algebraic holographic proof (AHP) that achieves linear proof length and constant query complexity, among other useful efficiency features. The protocol is for rank-1 constraint satisfiability (R1CS), a well-known generalization of arithmetic circuits where the "circuit description" is given by coefficient matrices (see definition below). Note that the relations that we consider consist of *triples* rather than *pairs*, because we need to split the verifier's input into a part for the offline phase and a part for the online phase. The offline input is called the *index*, and it consists of the coefficient matrices; the online input is called the *instance*, and it consists of a partial assignment to the variables. The algorithm that encodes the index (coefficient matrices) in the offline phase is called the *indexer*.

[6] The methodology also captures as a special case various folklore approaches used in prior works to construct *non*-preprocessing zkSNARKs via polynomial commitment schemes (see Section 1.2), thereby providing the first formal statement that clarifies what properties of algebraic proofs and polynomial commitment schemes are essential for these folklore approaches.

Definition 1 (informal)**.** *The* **indexed relation** $\mathcal{R}_{\mathrm{R1CS}}$ *is the set of triples* $(\mathbb{i}, \mathbb{x}, \mathbb{w}) = \big((\mathbb{F}, n, m, A, B, C), x, w\big)$ *where* $\mathbb{F}$ *is a finite field,* A, B, C *are* $n \times n$ *matrices over* $\mathbb{F}$*, each containing at most* m *non-zero entries, and* $z := (x, w)$ *is a vector in* $\mathbb{F}^n$ *such that* $Az \circ Bz = Cz$*. (Here "*$\circ$*" denotes the entry-wise product.)*

Theorem 2 (informal)**.** *There exists a constant-round AHP for the indexed relation* $\mathcal{R}_{\mathrm{R1CS}}$ *with linear proof length and constant query complexity. The soundness error is* $O(m/|\mathbb{F}|)$*, and the construction is a zero knowledge proof of knowledge. The arithmetic complexity of the indexer is* $O(m \log m)$*, of the prover is* $O(m \log m)$*, and of the verifier is* $O(|x| + \log m)$*.*

The literature on probabilistic proofs contains algebraic protocols that are holographic (e.g., [BFLS91] and [GKR15]) but *none achieve constant query complexity*, and so applying our methodology (Theorem 1) to these would lead to large argument sizes (many tens of kilobytes). These prior algebraic protocols rely on the multivariate sumcheck protocol applied to certain multivariate polynomials, which means that they incur sizable communication costs due to (a) the many rounds of the sumcheck protocol, and (b) the fact that applying the methodology would involve using multivariate polynomial commitment schemes that (for known constructions) lead to communication costs that are linear in the number of variables.

In contrast, our algebraic protocol relies on univariate polynomials and achieves constant query complexity, incurring small communication costs. Our algebraic protocol can be viewed as a "holographic variant" of the algebraic protocol for R1CS used in Aurora [BCR+19], because it achieves an *exponential* improvement in verification time when the verifier is given a suitable encoding of the coefficient matrices.

(3) Extractable polynomial commitments. Polynomial commitment schemes, introduced in [KZG10], are commitment schemes specialized to work with univariate polynomials. The security properties in [KZG10], while sufficient for the applications therein, do not appear sufficient for standalone use, or even just for the transformation in Theorem 1. We propose a definition for polynomial commitment schemes that incorporates the functionality and security that we believe to suffice for standalone use (and in particular suffices for Theorem 1). Moreover, we show how to extend the construction of [KZG10] to fulfill this definition in the plain model under non-falsifiable knowledge assumptions, or via a more efficient construction in the algebraic group model [FKL18] under falsifiable assumptions. These constructions are of independent interest, and when combined with our transformation, lead to the first efficient preprocessing arguments with universal SRS under concrete knowledge assumptions, and also to the efficiency reported in Fig. 1.

We have implemented in a Rust library[7] the polynomial commitment schemes, and our implementation of Marlin relies on this library. We deem this library of independent interest for other projects.

[7] `https://github.com/scipr-lab/poly-commit`

1.2 Related work

In this paper we study the goal of constructing preprocessing SNARGs with universal SRS, which achieve succinct verification regardless of the structure of the non-deterministic computation being checked. The most relevant prior work is Sonic [MBKM19], on which we improve as already discussed (see Fig. 1). The notion of updatable SRS was defined and achieved in [GKM+18], but with a less efficient construction.

Concurrent work. A concurrent work [GWC19] studies the same question as this paper, and also obtains efficiency improvements over Sonic [MBKM19]. Below is a brief comparison.

- We provide an implementation and evaluation of our construction, while [GWC19] do not. The estimated costs reported in [GWC19] suggest that an implementation may perform similarly to ours.
- Similarly to our work, [GWC19] extends the polynomial commitment in [KZG10] to support batching, and proves the extension secure in the algebraic group model. We additionally show how to prove security in the plain model under non-falsifiable knowledge assumptions, and consider the problem of enforcing different degrees for different polynomials (a feature that is not needed in [GWC19]).
- We show how to compile any algebraic holographic proof into a preprocessing argument with universal SRS, while [GWC19] focus on compiling a more restricted notion that they call "polynomial protocols".
- Our protocol natively supports R1CS, and can be viewed as a holographic variant of the algebraic protocol in [BCR+19]. The protocol in [GWC19] natively supports a different constraint system, and involves a protocol that, similar to [Gro10], uses a permutation argument to attest that all variables in the same cycle of a permutation are equal (e.g., $(1)(2,3)(4)$ would require that the second and third entries are equal).

Preprocessing SNARGs with a URS. Setty [Set19] studies preprocessing SNARGs with a URS (uniform reference string), and describes a protocol that for n-gate arithmetic circuits and a chosen constant $c \geq 2$ achieves proving time $O_\lambda(n)$, argument size $O_\lambda(n^{1/c})$, and verification time $O_\lambda(n^{1-1/c})$. The protocol in [Set19] offers a tradeoff compared to our work: preprocessing with a URS instead of a SRS, at the cost of asymptotically larger argument size and verification time. The question of achieving processing with a URS while also achieving asymptotically small argument size and verification time remains open.

The protocol in [Set19] is obtained by combining the multivariate polynomial commitments of [WTS+18] and a modern rendition of the PCP in [BFLS91] (which itself can be viewed as the "bare bones" protocol of [GKR15] for circuits of depth 1). [Set19] lacks an analysis of concrete costs, and also does not discuss how to achieve zero knowledge beyond stating that techniques in other papers [ZGK+17a; WTS+18; XZZ+19] can be applied. Nevertheless, argument sizes would at best be similar to these other papers (tens of kilobytes), which is much larger than our argument sizes (in the SRS model).

We conclude by noting that the informal security proof in [Set19] appears insufficient to show soundness of the argument system, because the polynomial commitment scheme is only assumed to be binding but not also extractable (there is no explanation of where the witness encoded in the committed polynomial comes from). Our definitions and security proofs, if ported over to the multivariate setting, would fill this gap.

Non-preprocessing SNARGs for arbitrary computations. Checking arbitrary circuits without preprocessing them requires the verifier to read the circuit, so the main goal is to obtain small argument size. In this setting of non-preprocessing SNARGs for arbitrary circuits, constructions with a URS (uniform reference string) are based on discrete logarithms [BCC+16; BBB+18] or hash functions [AHIV17; BCR+19], while constructions with a universal SRS (structured reference string) combine polynomial commitments and non-holographic algebraic proofs [Gab19]; all use random oracles to obtain non-interactive arguments.[8]

We find it interesting to remark that our methodology from Theorem 1 generalizes protocols such as [Gab19] in two ways. First, it formalizes the folklore approach of combining polynomial commitments and algebraic proofs to obtain arguments, identifying the security properties required to make this approach work. Second, it demonstrates how for algebraic *holographic* proofs the resulting argument enables preprocessing.

Non-preprocessing SNARGs for structured computations. Several works study SNARGs for structured computations. This structure enables fast verification *without* preprocessing. A line of works [BBC+17; BBHR19; BCG+19] combines hash functions and various interactive oracle proofs. Another line of works [ZGK+17b; ZGK+18; ZGK+17a; WTS+18; XZZ+19] combines multivariate polynomial commitments [PST13] and doubly-efficient interactive proofs [GKR15].

While in this paper we study a different setting (*preprocessing* SNARGs for *arbitrary* computations), there are similarities, and notable differences, in the polynomial commitments used in our work and prior works. We begin by noting that the notion of "multivariate polynomial commitments" varies considerably across prior works, despite the fact that most of those commitments are based on the protocol introduced in [PST13].

– The commitments used in [ZGK+17b; ZGK+18] are required to satisfy extractability (a stronger notion than binding) because the security proof of the argument system involves extracting a polynomial encoding a witness. The commitment is a modification of [PST13] that uses knowledge commitments, a standard ingredient to achieve extractability under non-falsifiable assumptions

[8] The linear verification time in most of the cited constructions can typically be partially mitigated via techniques that enable an untrusted party to help the verifier to check a batch of proofs for the same circuit faster than checking each proof individually (the linear cost in the circuit is paid only once per batch rather than once for each proof in the batch).

in the plain model. Neither of these works consider hiding commitments as zero knowledge is not a goal for them.

– The commitments used in [ZGK+17a; WTS+18] must be compatible with the Cramer–Damgård transform [CD98] used in constructing the argument system. They consider a *modified setting* where the sender does not reveal the value of the commitment polynomial at a desired point but, instead, reveals a commitment to this value, along with a proof attesting that the committed value is correct. For this modified setting, they consider commitments that satisfy natural notions of extractability *and hiding* (achieving zero knowledge arguments is a goal in both papers). The commitments constructed in the two papers offer different tradeoffs. The commitment in [ZGK+17a] is based on [PST13]: it relies on a SRS (structured reference string); it uses pairings; and for ℓ-variate polynomials achieves $O_\lambda(\ell)$-size arguments that can be checked in $O_\lambda(\ell)$ time. The commitment in [WTS+18] is inspired from [BG12] and [BBB+18]: it relies on a URS (uniform reference string); it does not use pairings; and for ℓ-variate *multilinear* polynomials and a given constant $c \geq 2$ achieves $O_\lambda(2^{\ell/c})$-size arguments that can be checked in $O_\lambda(2^{\ell-\ell/c})$ time.

– The commitments used in [XZZ+19] are intended for the regular (unmodified) setting of commitment schemes where the sender reveals the value of the polynomial, because zero knowledge is later achieved by building on the algebraic techniques described in [CFS17]. The commitment definition in [XZZ+19] considers binding and hiding, but not extractability. However, the given security analysis for the argument system does not seem to go through for this definition (there is no explanation of where the witness encoded in the committed polynomial comes from). Also, no commitment construction is provided in [XZZ+19], and instead the reader is referred to [ZGK+17a], which considers the modified setting described above.

In sum there are multiple notions of commitment and one must be precise about the functionality and security needed to construct an argument system. We now compare prior notions of commitments to the one that we use.

First, since in this paper we do not use the Cramer–Damgård transform for zero knowledge, commitments in the modified setting are not relevant. Instead, we achieve zero knowledge via *bounded independence* [BCGV16], and in particular we consider the familiar setting where the sender reveals evaluations to the committed polynomial. Second, prior works consider protocols where the sender commits to a polynomial in a single round, while we consider protocols where the sender commits to multiple polynomials of different degrees in each of several rounds. This multi-polynomial multi-round setting requires suitable extensions in terms of functionality (to enable batching techniques to save on argument size) and security (extractability and hiding need to be strengthened), which means that prior definitions do not suffice for us.

The above discrepancies have led us to formulate new definitions of functionality and security for polynomial commitments (as summarized in Section 2.2). We conclude by noting that, since in this paper we construct arguments that use *univariate* polynomials, our definitions are specialized to commitments for univariate polynomials. Corresponding definitions for multivariate polynomials can be obtained with straightforward modifications, and would strengthen definitions appearing in some prior works. Similarly, we fulfill the required definitions via natural adaptations of the univariate scheme of [KZG10], and analogous adaptations of the multivariate scheme of [PST13] would fulfill the multivariate analogues of these definitions.

2 Techniques

We discuss the main ideas behind our results. First we describe the two building blocks used in Theorem 1: AHPs and polynomial commitment schemes (described in Sections 2.1 and 2.2 respectively). We describe how to combine these to obtain preprocessing arguments with universal SRS in Section 2.3. Next, we discuss constructions for these building blocks: in Section 2.4 we describe our AHP (underlying Theorem 2), and in Section 2.5 we describe our construction of polynomial commitments.

Throughout, instead of considering the usual notion of relations that consist of instance-witness pairs, we consider *indexed relations*, which consist of triples $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ where $\mathbb{i}$ is the index, $\mathbb{x}$ is the instance, and $\mathbb{w}$ is the witness. This is because $\mathbb{i}$ represents the part of the verifier input that is preprocessed in the offline phase (e.g., the circuit description) and $\mathbb{x}$ represents the part of the verifier input that comes in the online phase (e.g., a partial assignment to the circuit's input wires). The *indexed language* corresponding to an indexed relation $\mathcal{R}$, denoted $\mathcal{L}(\mathcal{R})$, is the set of pairs $(\mathbb{i}, \mathbb{x})$ for which there exists a witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$.

2.1 Building block: algebraic holographic proofs

Interactive oracle proofs (IOPs) [BCS16; RRR16] are multi-round protocols where in each round the verifier sends a challenge and the prover sends an oracle (which the verifier can query). IOPs combine features of interactive proofs and probabilistically checkable proofs . *Algebraic holographic proofs* (AHPs) modify the notion of an IOP in two ways.

– *Holographic:* the verifier does not receive its input explicitly but, rather, has oracle access to a prescribed *encoding* of it. This potentially enables the verifier to run in time that is much faster than the time to read its input in full. (Our constructions will achieve this fast verification.)
– *Algebraic:* the honest prover must produce oracles that are low-degree polynomials (this restricts the completeness property), and all malicious provers must produce oracles that are low-degree polynomials (this relaxes the soundness property). The encoded input to the verifier must also be a low-degree polynomial.

Since in this paper we only work with *univariate* polynomials, our definitions focus on this case, but they can be modified in a straightforward way to be more general.

Informally, a (public-coin) AHP over a field $\mathbb{F}$ for an indexed relation $\mathcal{R}$ is specified by an indexer **I**, prover **P**, and verifier **V** that work as follows.

– *Offline phase.* The indexer $\mathbf{I}$ receives as input the index $\mathbb{i}$ to be preprocessed, and outputs one or more univariate polynomials over $\mathbb{F}$ encoding $\mathbb{i}$.
– *Online phase.* For some instance $\mathbb{x}$ and witness $\mathbb{w}$, the prover $\mathbf{P}$ receives $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ and the verifier $\mathbf{V}$ receives $\mathbb{x}$; $\mathbf{P}$ and $\mathbf{V}$ interact over some (in this paper, constant) number of rounds, where in each round $\mathbf{V}$ sends a challenge and $\mathbf{P}$ sends one or more polynomials; after the interaction, $\mathbf{V}(\mathbb{x})$ probabilistically queries the polynomials output by the indexer and the polynomials output by the prover, and then accepts or rejects. Crucially, $\mathbf{V}$ does *not* receive $\mathbb{i}$ as input, but instead queries the polynomials output by $\mathbf{I}$ that encode $\mathbb{i}$. This enables the construction of verifiers $\mathbf{V}$ that run in time that is sublinear in $|\mathbb{i}|$.

The completeness property states that for every $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ the probability that $\mathbf{P}(\mathbb{i}, \mathbb{x}, \mathbb{w})$ convinces $\mathbf{V}^{\mathbf{I}(\mathbb{i})}(\mathbb{x})$ to accept is 1. The soundness property states that for every $(\mathbb{i}, \mathbb{x}) \notin \mathcal{L}(\mathcal{R})$ and *admissible* prover $\tilde{\mathbf{P}}$ the probability that $\tilde{\mathbf{P}}$ convinces $\mathbf{V}^{\mathbf{I}(\mathbb{i})}(\mathbb{x})$ to accept is at most a given soundness error ϵ. A prover is "admissible" if the degrees of the polynomials it outputs fit within prescribed degree bounds of the protocol. See Section 4 for details on AHPs.

2.2 Building block: polynomial commitments

Informally, a *polynomial commitment scheme* [KZG10] allows a prover to produce a commitment c to a univariate polynomial $p \in \mathbb{F}[X]$, and later "open" $p(X)$ at any point $z \in \mathbb{F}$, producing an *evaluation proof* π showing that the opened value is consistent with the polynomial "inside" c at z. Turning this informal goal into a useful definition requires some care, however, as we explain below. In this paper we propose a set of definitions for polynomial commitment schemes that we believe are useful for standalone use, and in particular suffice as a building block for our compiler described in Section 2.3.

First, we consider constructions with strong efficiency requirements: the commitment c is much smaller than the polynomial p (e.g., c consists of a constant number of group elements), and the proof π can be validated very fast (e.g., in a constant number of cryptographic operations). These requirements not only rule out natural constructions, but also imply that the usual binding property, which states that an efficient adversary cannot open the same commitment to two different values, does not capture the desired security. Indeed, even if the adversary were to be bound to opening values of some function $f \colon \mathbb{F} \to \mathbb{F}$, it may be that the function f is consistent with a polynomial whose degree is *higher* than what was claimed. This means that a security definition needs to incorporate guarantees about the degree of the committed function.[9]

[9] This consideration motivates the *strong correctness* property in [KZG10], which states that *if* the adversary knows a polynomial that leads to the claimed commitment c then this polynomial has bounded degree. This notion, while sufficient for the application in [KZG10], does not seem to suffice for standalone use because there is no a priori guarantee that an adversary that can open values to a commitment knows a polynomial inside the commitment. In some sense, a knowledge assumption is hidden in this hypothesis.

Second, in many applications of polynomial commitments, an adversary produces multiple commitments to polynomials within a round of interaction and across rounds of interaction. After this interaction, the adversary reveals values of all of these polynomials at one or more locations. This setting motivates a number of considerations. First, it is desirable to rely on a single set of public parameters for committing to multiple polynomials, even if the polynomials differ in degree. A construction such as that of [KZG10] can be modified in a natural way to achieve this is by committing both to the polynomial and its shift to the maximum degree, similarly to techniques used to bundle multiple low-degree tests into a single one [BCR+19]. This modification needs to be addressed in any proof of security. Second, it would be desirable to batch evaluation proofs across different polynomials for the same location. Again, the construction in [KZG10] can support this, but one must argue that security still holds in this more general case.

The preceeding considerations require an extension of previous definitions and motivate our re-formulation of the primitive. Informally, a polynomial commitment scheme PC is a tuple of algorithms $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Trim}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Check})$. The setup algorithm $\mathsf{PC.Setup}$ takes as input a security parameter and maximum supported degree bound D, and outputs public parameters pp that contain the description of a finite field $\mathbb{F}$. The "trimming" algorithm $\mathsf{PC.Trim}$ then deterministically specializes these parameters for a given set of degree bounds and outputs a committer key ck and a receiver key rk. The sender can then invoke $\mathsf{PC.Commit}$ with input ck and a list of polynomials $\boldsymbol{p}$ with respective degree bounds $\boldsymbol{d}$ to generate a set of commitments $\boldsymbol{c}$. Subsequently, the sender can use $\mathsf{PC.Open}$ to produce a proof π that convinces the receiver that the polynomials "inside" $\boldsymbol{c}$ respect the degree bounds $\boldsymbol{d}$ and, moreover, evaluate to the claimed set of values $\boldsymbol{v}$ at a given query set Q that specifies any number of evaluation points for each polynomial. The receiver can invoke $\mathsf{PC.Check}$ to check this proof.

The scheme PC is required to satisfy *extractability* and *efficiency* properties, and also, optionally, a *hiding* property. We outline these properties below, and provide detailed definitions in the full version.

Extractability. Consider an efficient sender adversary $\mathcal{A}$ that can produce a commitment c and degree bound $d \leq D$ such that, when asked for an evaluation at some point $z \in \mathbb{F}$, can produce a supposed evaluation v and proof π such that $\mathsf{PC.Check}$ accepts. Then PC is *extractable* if for every maximum degree bound D and every sender adversary $\mathcal{A}$ who can produce such commitments, there exists a corresponding efficient extractor $\mathcal{E}_{\mathcal{A}}$ that outputs a polynomial p of degree at most d that "explains" c so that $p(z) = v$. While for simplicity we have described the most basic case here, our definition considers adversaries and extractors who interact over multiple rounds, wherein the adversary may produce multiple commitments in each round and the extractor is required to output corresponding polynomials on a per-round basis (before seeing the query set, proof, or supposed evaluations).

In this work we rely on extractability to prove the security of our compiler (see Section 2.3); we do not know if weaker security notions studied in prior works, such as evaluation binding, suffice. More generally, we believe that extractability is a useful property that may be required across a range of other applications.

Efficiency. We require two notions of efficiency for PC. First, the time required to commit to a polynomial p and then to create an evaluation proof must be proportional to the degree of p, and not to the maximum degree D. (This ensures that the argument prover runs in time proportional to the size of the index.)

On the receiver's side, the commitment size, proof size, and time to verify an opening must be independent of the claimed degrees for the polynomials. (This ensures that the argument produced by our compiler is succinct.)

Hiding. The hiding property of PC states that commitments and proofs of evaluation reveal no information about the committed polynomial beyond the publicly stated degree bound and the evaluation itself. Namely, PC is *hiding* if there exists an efficient simulator that outputs simulated commitments and simulated evaluation proofs that cannot be distinguished from their real counterparts by any malicious distinguisher that only knows the degree bound and the evaluation.

Analogously to the case of extractability, we actually consider a more general definition that considers commitments to multiple polynomials within and across multiple rounds; moreover, the definition considers the case where some polynomials are designated as not hidden (and thus given to the simulator) because in our application we sometimes prefer to commit to a polynomial in a non-hiding way (for efficiency reasons).

2.3 Compiler: from AHPs to preprocessing arguments with universal SRS

We describe the main ideas behind Theorem 1, which uses polynomial commitment schemes to compile any (public-coin) AHP into a corresponding (public-coin) preprocessing argument with universal SRS. In a subsequent step, the argument can be made non-interactive via the Fiat–Shamir transformation, and thereby obtain a preprocessing SNARG with universal SRS.

The basic intuition of the compiler follows the well-known framework of "commit to oracles and then open query answers" pioneered by Kilian [Kil92]. However, the commitment scheme used in our compiler leverages and enforces the algebraic structure of these oracles. While several works in the literature already take advantage of algebraic commitment schemes applied to algebraic oracles, our contribution is to observe that if we apply this framework to a holographic proof then we obtain a preprocessing argument.

Informally, first the argument indexer invokes the AHP indexer to generate polynomials, and then deterministically commits to these using the polynomial commitment scheme. Subsequently, the argument prover and argument verifier interact, each respectively simulating the AHP prover and AHP verifier. In each round, the argument prover sends succinct commitments to the polynomials output by the AHP prover in that round. After the interaction, the argument verifier declares its queries to the polynomials (of the prover and of the indexer).

The argument prover replies with the desired evaluations along with an evaluation proof attesting to their correctness relative to the commitments.

This approach, while intuitive, must be proven secure. In particular, in the proof of soundness, we need to show that if the argument prover convinces the argument verifier with a certain probability, then we can find an AHP prover that convinces the AHP verifier with similar probability. This step is non-trivial: the AHP prover outputs polynomials, while the argument prover merely outputs succinct commitments and a few evaluations, which is much less information. In order to deduce the former from the latter requires *extraction*. This motivates considering polynomial commitment schemes that are extractable, in the sense described in Section 2.2. We do not know whether weaker security properties, such as the evaluation binding property studied in some prior works, suffice for proving the compiler secure.

The compiler outlined above is compatible with the properties of argument of knowledge and zero knowledge. Specifically, we prove that if the AHP is a proof of knowledge, then the compiler produces an argument of knowledge; also, if the AHP is (bounded-query) zero knowledge and the polynomial commitment scheme is hiding, then the compiler produces a zero knowledge argument.

See the full version for more details on the compiler.

2.4 Construction: an AHP for constraint systems

In prior sections we have described how we can use polynomial commitment schemes to compile AHPs into corresponding preprocessing SNARGs. In this section we discuss the main ideas behind Theorem 2, which provides an efficient AHP for the indexed relation corresponding to R1CS (see Definition 1). The preprocessing zkSNARK that we achieve in this paper (see Fig. 1) is based on this AHP.

Our protocol can be viewed as a "holographic variant" of the *non*-holographic algebraic proof for R1CS constructed in [BCR+19]. Achieving holography involves designing a new sub-protocol that enables the verifier to evaluate low-degree extensions of the coefficient matrices at a random location. While in [BCR+19] the verifier performed this computation in time $\mathrm{poly}(|\mathbb{i}|)$ on its own, in our protocol the verifier performs it *exponentially faster*, in time $O(\log |\mathbb{i}|)$, by receiving help from the prover and having oracle access to the polynomials produced by the indexer. We introduce notation and then discuss the protocol.

Some notation. Consider an index $\mathbb{i} = (\mathbb{F}, n, m, A, B, C)$ specifying coefficient matrices, an instance $\mathbb{x} = x \in \mathbb{F}^*$ specifying a partial assignment to the variables, and a witness $\mathbb{w} = w \in \mathbb{F}^*$ specifying an assignment to the other variables such that the R1CS equation holds. The R1CS equation holds if and only if $Az \circ Bz = Cz$ for $z := (x, w) \in \mathbb{F}^n$. Below, we let H and K be prescribed subsets of $\mathbb{F}$ of sizes n and m respectively; we also let $v_H(X)$ and $v_K(X)$ be the vanishing polynomials of these two sets. (The vanishing polynomial of a set S is the monic polynomial of degree $|S|$ that vanishes on S, i.e., $\prod_{\gamma \in S}(X - \gamma)$.) We assume that both H and K are smooth multiplicative subgroups. This allows interpolation/evaluation over H in $O(n \log n)$ operations and also makes $v_H(X)$ computable in $O(\log n)$ operations (and similarly for K). Given an $n \times n$ matrix

M with rows/columns indexed by elements of H, we denote by $\hat{M}(X,Y)$ the low-degree extension of M, i.e., the polynomial of individual degree less than n such that $\hat{M}(\kappa,\iota)$ is the (κ,ι)-th entry of M for every $\kappa,\iota \in H$.

A non-holographic starting point. We sketch a *non*-holographic protocol for R1CS with linear proof length and constant query complexity, inspired from [BCR+19], that forms the starting point of our work. In this case the prover receives as input $(\mathbb{i},\mathbb{x},\mathbb{w})$ and the verifier receives as input $(\mathbb{i},\mathbb{x})$. (The verifier reads the non-encoded index $\mathbb{i}$ because we are describing a non-holographic protocol.)

In the first message the prover $\mathbf{P}$ sends the univariate polynomial $\hat{z}(X)$ of degree less than n that agrees with the variable assignment z on H, and also sends the univariate polynomials $\hat{z}_A(X),\hat{z}_B(X),\hat{z}_C(X)$ of degree less than n that agree with the linear combinations $z_A := Az$, $z_B := Bz$, and $z_C := Cz$ on H. The prover is left to convince the verifier that the following two conditions hold:

(1) Entry-wise product: $\forall\, \kappa \in H\ ,\ \hat{z}_A(\kappa)\hat{z}_B(\kappa) - \hat{z}_C(\kappa) = 0\ .$

(2) Linear relation: $\forall\, M \in \{A,B,C\}\ ,\ \forall\, \kappa \in H\ ,\ \hat{z}_M(\kappa) = \sum_{\iota\in H} M[\kappa,\iota]\hat{z}(\iota)\ .$

(The prover also needs to convince the verifier that $\hat{z}(X)$ encodes a full assignment z that is consistent with the partial assignment x, but we for simplicity we ignore this in this informal discussion.)

In order to convince the verifier of the first (entry-wise product) condition, the prover sends the polynomial $h_0(X)$ such that $\hat{z}_A(X)\hat{z}_B(X) - \hat{z}_C(X) = h_0(X)v_H(X)$. This polynomial equation is equivalent to the first condition (the left-hand side equals zero everywhere on H if and only if it is a multiple of H's vanishing polynomial). The verifier will check the equation at a random point $\beta \in \mathbb{F}$: it queries $\hat{z}_A(X),\hat{z}_B(X),\hat{z}_C(X),h_0(X)$ at β, evaluates $v_H(X)$ at β on its own, and checks that $\hat{z}_A(\beta)\hat{z}_B(\beta) - \hat{z}_C(\beta) = h_0(\beta)v_H(\beta)$. The soundness error is the maximum degree over the field size, which is at most $2n/|\mathbb{F}|$.

In order to convince the verifier of the second (linear relation) condition, the prover expects a random challenge $\alpha \in \mathbb{F}$ from the verifier, and then replies in a second message. For each $M \in \{A,B,C\}$, the prover sends polynomials $h_M(X)$ and $g_M(X)$ such that

$$\begin{array}{l} r(\alpha,X)\hat{z}_M(X) - r_M(\alpha,X)\hat{z}(X) = h_M(X)v_H(X) + Xg_M(X) \\ \quad \text{for} \quad r_M(Z,X) := \sum_{\kappa\in H} r(Z,\kappa)\hat{M}(\kappa,X) \end{array}$$

where $r(Z,X)$ is a prescribed polynomial of individual degree less than n such that $(r(Z,\kappa))_{\kappa\in H}$ are n linearly independent polynomials. Prior work [BCR+19] on checking linear relations via univariate sumchecks shows that this polynomial equation is equivalent, up to a soundness error of $n/|\mathbb{F}|$ over α, to the second condition.[10] The verifier will check this polynomial equation at the random point

[10] In particular, we are using the fact from [BCR+19] that, given a multiplicative subgroup S of $\mathbb{F}$, a polynomial $f(X)$ sums to σ over S if and only if $f(X)$ can be written as $h(X)v_S(X)+Xg(X)+\sigma/|S|$ for some $h(X)$ and $g(X)$ with $\deg(g) < |S|-1$.

$\beta \in \mathbb{F}$: it queries $\hat{z}(X), \hat{z}_A(X), \hat{z}_B(X), \hat{z}_C(X), h_M(X), g_M(X)$ at β, evaluates $v_H(X)$ at β on its own, evaluates $r(Z, X)$ and $r_M(Z, X)$ at (α, β) on its own, and checks that $r(\alpha,\beta)\hat{z}_M(\beta) - r_M(\alpha,\beta)\hat{z}(\beta) = h_M(\beta)v_H(\beta) + \beta g_M(\beta)$. The additional soundness error is $2n/|\mathbb{F}|$.

The above is a simple 3-message protocol for R1CS with soundness error $\max\{2n/|\mathbb{F}|, 3n/|\mathbb{F}|\} = 3n/|\mathbb{F}|$ in the setting where the honest prover and malicious provers send polynomials of prescribed degrees, which the verifier can query at any location. The proof length (sum of all degrees) is linear in n and the query complexity is constant.

Barrier to holography. The verifier in the above protocol runs in time that is $\Omega(|\mathbb{i}|) = \Omega(n+m)$. While this is inherent in the non-holographic setting (because the verifier must read $\mathbb{i}$), we now discuss how exactly the verifier's computation depends on $\mathbb{i}$. We shall later use this understanding to achieve an exponential improvement in the verifier's time when given a suitable encoding of $\mathbb{i}$.

The verifier's check for the entry-wise product is $\hat{z}_A(\beta)\hat{z}_B(\beta) - \hat{z}_C(\beta) = h_0(\beta)v_H(\beta)$, and can be carried out in $O(\log n)$ operations *regardless* of the coefficient matrices contained in the index $\mathbb{i}$. In other words, this check is efficient even in the non-holographic setting. However, the verifier's check for the linear relation is $r(\alpha,\beta)\hat{z}_M(\beta) - r_M(\alpha,\beta)\hat{z}(\beta) = h_M(\beta)v_H(\beta) + \beta g_M(\beta)$, which has a linear cost. Concretely, evaluating the polynomial $r_M(Z, X)$ at (α, β) requires $\Omega(n + m)$ operations.

In the holographic setting, a natural idea to reduce this cost would be to grant the verifier oracle access to the low-degree extension $\hat{M}$ for $M \in \{A, B, C\}$. This idea has two problems: the verifier *still* needs $\Omega(n)$ operations to evaluate $r_M(Z, X)$ at (α, β) and, moreover, the size of $\hat{M}$ is *quadratic* in n, which means that the encoding of the index $\mathbb{i}$ is $\Omega(n^2)$. We cannot afford such an expensive encoding in the offline preprocessing phase. We now describe how we overcome both of these problems, and obtain a holographic protocol.

Achieving holography. To overcome the above problems and obtain a holographic protocol, we rely yet again on the univariate sumcheck protocol. We introduce two additional rounds of interaction, and in each round the verifier learns that their verification equation holds provided the sumcheck from the next round holds. The last sumcheck will rely on polynomials output by the indexer, which the verifier knows are correct.

We address the first problem by letting the prover and verifier interact in an additional round, where we rely on an additional univariate sumcheck to reduce the problem of evaluating $r_M(Z, X)$ at (α, β) to the problem of evaluating $\hat{M}$ at (β_2, β) for a random $\beta_2 \in \mathbb{F}$. Namely, the verifier sends β to the prover, who computes

$$\sigma_2 := r_M(\alpha, \beta) = \sum_{\kappa \in H} r(\alpha, \kappa)\hat{M}(\kappa, \beta).$$

Then the prover replies with σ_2 and the polynomials $h_2(X)$ and $g_2(X)$ such that

$$r(\alpha, X)\hat{M}(X, \beta) = h_2(X)v_H(X) + Xg_2(X) + \sigma_2/n \ .$$

Prior techniques on univariate sumcheck [BCR+19] tell us that this equation is equivalent to the polynomial $r(\alpha, X)\hat{M}(X, \beta)$ summing to σ_2 on H. Thus the verifier needs to check this equation at a random $\beta_2 \in \mathbb{F}$: $r(\alpha, \beta_2)\hat{M}(\beta_2, \beta) = h_2(\beta_2)v_H(\beta_2) + \beta_2 g_2(\beta_2) + \sigma_2/n$. The only expensive part of this equation for the verifier is computing the value $\hat{M}(\beta_2, \beta)$, which is problematic. Indeed, we have already noted that we cannot afford to simply let the verifier have oracle access to $\hat{M}$, because this polynomial has quadratic size (it contains a quadratic number of terms).

We address this second problem as follows. Let $u_H(X, Y) := \frac{v_H(X) - v_H(Y)}{X - Y}$ be the formal derivative of the vanishing poynomial $v_H(X)$, and note that $u_H(X, Y)$ vanishes on the square $H \times H$ except for on the diagonal, where it takes on the (non-zero) values $(u_H(a, a))_{a \in H}$. Moreover, $u_H(X, Y)$ can be evaluated at any point in $\mathbb{F} \times \mathbb{F}$ in $O(\log n)$ operations. Using this polynomial, we can write $\hat{M}$ as a sum of $m = |K|$ terms instead of $n^2 = |H|^2$ terms:

$$\hat{M}(X, Y) := \sum_{\kappa \in K} u_H(X, \hat{\mathsf{row}}_M(\kappa)) \cdot u_H(Y, \hat{\mathsf{col}}_M(\kappa)) \cdot \hat{\mathsf{val}}_M(\kappa) \ ,$$

where $\hat{\mathsf{row}}_M, \hat{\mathsf{col}}_M, \hat{\mathsf{val}}_M$ are the low-degree extensions of the row, column, and value of the non-zero entries in M according to some canonical order over K.[11]

This method of representing the low-degree extension of M suggests an idea: let the verifier have oracle access to the polynomials $\hat{\mathsf{row}}_M, \hat{\mathsf{col}}_M, \hat{\mathsf{val}}_M$ and do *yet another* univariate sumcheck, but this time over the set K. The verifier sends β_2 to the prover, who computes

$$\sigma_3 := \hat{M}(\beta_2, \beta) = \sum_{\kappa \in K} u_H(\beta_2, \hat{\mathsf{row}}_M(\kappa)) \cdot u_H(\beta, \hat{\mathsf{col}}_M(\kappa)) \cdot \hat{\mathsf{val}}_M(\kappa) \ .$$

Then the prover replies with σ_3 and the polynomials $h_3(X)$ and $g_3(X)$ such that

$$u_H(\beta_2, \hat{\mathsf{row}}_M(X))u_H(\beta, \hat{\mathsf{col}}_M(X))\hat{\mathsf{val}}_M(X) = h_3(X)v_K(X) + Xg_3(X) + \sigma_3/m \ .$$

The verifier can then check this equation at a random $\beta_3 \in \mathbb{F}$, which only requires $O(\log m)$ operations.

The above idea *almost* works; the one remaining problem is that $h_3(X)$ has degree $\Omega(nm)$ (because the left-hand size of the equation has quadratic degree), which is too expensive for our target of a quasilinear-time prover. We overcome this problem by letting the prover run the univariate sumcheck protocol on the unique low-degree extension $\hat{f}(X)$ of the function $f\colon K \to \mathbb{F}$ defined as $f(\kappa) := u_H(\beta_2, \hat{\mathsf{row}}_M(\kappa))u_H(\beta, \hat{\mathsf{col}}_M(\kappa))\hat{\mathsf{val}}_M(\kappa)$. Observe that $\hat{f}(X)$ has degree less than m. The verifier checks that $\hat{f}(X)$ and $u_H(\beta_2, \hat{\mathsf{row}}_M(X))u_H(\beta, \hat{\mathsf{col}}_M(X))\hat{\mathsf{val}}_M(X)$ agree on K.

From sketch to protocol. In the above discussion we have ignored a number of technical aspects, such as proof of knowledge and zero knowledge (which

[11] Technicality: $\hat{\mathsf{val}}(\kappa)$ actually equals the value divided by $u_H(\hat{\mathsf{row}}_M(\kappa), \hat{\mathsf{row}}_M(\kappa))u_H(\hat{\mathsf{col}}_M(\kappa), \hat{\mathsf{col}}_M(\kappa))$.

are ultimately needed in the compiler if we want to construct a preprocessing zkSNARK). We have also not discussed time complexities of many algebraic steps, and we omitted discussion of how to batch multiple sumchecks into fewer ones, which brings important savings in argument size. For details, see our detailed construction in Section 5.

2.5 Construction: extractable polynomial commitments

We now sketch how to construct a polynomial commitment scheme that achieves the strong functionality and security requirements of our definition in Section 2.2. Our starting point is the $\mathsf{PolyCommit}_{\mathrm{DL}}$ construction of Kate et al. [KZG10], and then describe a sequence of natural and generic transformations that extend this construction to enable extractability, commitments to multiple polynomials, and the enforcement of per-polynomial degree bounds. In fact, once we arrive at a scheme that supports extractability for committed polynomials at a single point, our transformations build on this construction in a black box way to first support per-polynomial degree bounds, and then query sets that may request multiple evaluation points per polynomial. See the full version for details of these transformations.

Starting point: $\mathsf{PolyCommit}_{\mathbf{DL}}$. The setup phase samples a cryptographically secure bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, G, H, e)$ and then samples a committer key ck and receiver key rk for a given degree bound D. The committer key consists of group elements encoding powers of a random field element β, namely, $\mathsf{ck} := \{G, \beta G, \ldots, \beta^D G\} \in \mathbb{G}_1^{D+1}$. The receiver key consists of the group elements $\mathsf{rk} := (G, H, \beta H) \in \mathbb{G}_1 \times \mathbb{G}_2^2$. Note that the SRS, which consists of the keys ck and rk, is updatable because the coefficients of group elements in the SRS are all monomials.

To commit to a polynomial $p \in \mathbb{F}_q[X]$, the sender computes $c := p(\beta)G$. To subsequently prove that the committed polynomial evaluates to v at a point z, the sender computes a witness polynomial $w(X) := (p(X) - p(z))/(X - z)$, and provides as proof a commitment to w: $\pi := w(\beta)G$. The idea is that the witness function w is a polynomial *if and only if* $p(z) = v$; otherwise, it is a rational function, and cannot be committed to using ck.

Finally, to verify a proof of evaluation, the receiver checks that the commitment and proof of evaluation are consistent. That is, it checks that the proof commits to a polynomial of the form $(p(X) - p(z))/(X - z)$ by checking the equality $e(c - vG, H) = e(\pi, \beta H - zH)$.

Achieving extractability. While the foregoing construction guarantees correctness of evaluations, it does not by itself guarantee that a commitment actually "contains" a suitable polynomial of degree at most D. We study two methods to address this issue, and thereby achieve extractability. One method is to modify the construction to use knowledge commitments [Gro10], and rely on a concrete knowledge assumption. The main disadvantage of this approach is that each commitment doubles in size. The other method is to move away from the plain model, and instead conduct the security analysis in the algebraic group model

(AGM) [FKL18]. This latter method is more efficient because each commitment remains a single group element.

Committing to multiple polynomials at once. We enable the sender to simultaneously open multiple polynomials $[p_i]_{i=1}^n$ at the same point z as follows. Before generating a proof of evaluation for $[p_i]_{i=1}^n$, the sender requests from the receiver a random field element ξ, which he uses to take a random linear combination of the polynomials: $p := \sum_{i=1}^n \xi^i p_i$, and generates a proof of evaluation π for this polynomial p.

The receiver verifies π by using the fact that the commitments are additively homomorphic. The receiver takes a linear combination of the commitments and claimed evaluations, obtaining the combined commitment $c = \sum_{i=1}^n \xi^i c_i$ and evaluation $v = \sum_{i=1}^n \xi^i v_i$. Finally, it checks the pairing equations for c, π, and v.

Completeness of this check is straightforward, while soundness follows from the fact that if any polynomial does not match its evaluation, then the combined polynomial will not match its evaluation with high probability.

Enforcing multiple degree bounds. The construction so far enforces a single bound D on the degrees of all the polynomials p_i. To enforce a different degree bound d_i for each p_i, we require the sender to commit not only to each p_i, but also to "shifted polynomials" $p_i'(X) := X^{D-d_i} p_i(X)$. The proof of evaluation proves that, if p_i evaluates to v_i at z, then p_i' evaluates to $z^{D-d_i} v_i$.

The receiver checks that the commitment for each p_i' corresponds to an evaluation $z^{D-d_i} v_i$ so that, if z is sampled from a super-polynomial subset of $\mathbb{F}_q$, the probability that $\deg(p_i) \neq d_i$ is negligible. This trick is similar to the one used in [BS08; BCR+19] to derive low-degree tests for specific degree bounds.

However, while sound, this approach is inefficient in our setting: the witness polynomial for p_i' has $\Omega(D)$ non-zero coefficients (instead of $O(d_i)$), and so constructing an evaluation proof for it requires $\Omega(D)$ scalar multiplications (instead of $O(d_i)$). To work around this, we instead produce a proof that the related polynomial $p_i^\star(X) := p_i'(X) - p_i(z) X^{D-d_i}$ evaluates to 0 at z. As we show in the full version, the witness polynomial for this claim has $O(d_i)$ non-zero coefficients, and so constructing the evaluation proof can be done in $O(d_i)$ scalar multiplications. Completeness is preserved because the receiver can check the correct evaluation of $p_i^\star$ by subtracting $p_i(z)(\beta^{D-d_i}\mathbb{G})$ from the commitment to the shifted polynomial p_i', thereby obtaining a commitment to $p_i^\star$, while security is preserved because $p_i'(z) = z^{D-d_i} v_i \iff p_i^\star(z) = 0$.

Evaluating at a query set instead of a single point. To support the case where the polynomials $[p_i]_{i=1}^n$ are evaluated at a set of points Q, the sender proceeds as follows. Say that there are k different points $[z_i]_{i=1}^k$ in Q. The sender partitions the polynomials $[p_i]_{i=1}^n$ into different groups such that every polynomial in a group is to be evaluated at the same point z_i. The sender runs $\mathsf{PC.Open}$ on each group, and outputs the resulting list of evaluation proofs.

Achieving hiding. To additionally achieve hiding, we follow the above blueprint, replacing $\mathsf{PolyCommit}_{\mathrm{DL}}$ with the hiding scheme $\mathsf{PolyCommit}_{\mathrm{Ped}}$ described in [KZG10].

3 Preliminaries

We denote by $[n]$ the set $\{1, \ldots, n\} \subseteq \mathbb{N}$. We use $\boldsymbol{a} = [a_i]_{i=1}^n$ as a short-hand for the tuple $(a_1, \ldots, a_n)$, and $[\boldsymbol{a}_i]_{i=1}^n = [[a_{i,j}]_{j=1}^m]_{i=1}^n$ as a short-hand for the tuple $(a_{1,1}, \ldots, a_{1,m}, \ldots, a_{n,1}, \ldots, a_{n,m})$; $|\boldsymbol{a}|$ denotes the number of entries in $\boldsymbol{a}$. If x is a binary string then $|x|$ denotes its bit length. If M is a matrix then $\|M\|$ denotes the number of nonzero entries in M. If S is a finite set then $|S|$ denotes its cardinality and $x \leftarrow S$ denotes that x is an element sampled at random from S. We denote by $\mathbb{F}$ a finite field, and whenever $\mathbb{F}$ is an input to an algorithm we implicitly assume that $\mathbb{F}$ is represented in a way that allows efficient field arithmetic. Given a finite set S, we denote by $\mathbb{F}^S$ the set of vectors indexed by elements in S. We denote by $\mathbb{F}[X]$ the ring of univariate polynomials over $\mathbb{F}$ in X, and by $\mathbb{F}^{<d}[X]$ the set of polynomials in $\mathbb{F}[X]$ with degree less than d.

We denote by $\lambda \in \mathbb{N}$ a security parameter. When we state that $n \in \mathbb{N}$ for some variable n, we implicitly assume that $n = \text{poly}(\lambda)$. We denote by $\text{negl}(\lambda)$ an unspecified function that is *negligible* in λ (namely, a function that vanishes faster than the inverse of any polynomial in λ). When a function can be expressed in the form $1 - \text{negl}(\lambda)$, we say that it is *overwhelming* in λ. When we say that $\mathcal{A}$ is an *efficient adversary* we mean that $\mathcal{A}$ is a family $\{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ of non-uniform polynomial-size circuits. If the adversary consists of multiple circuit families $\mathcal{A}_1, \mathcal{A}_2, \ldots$ then we write $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \ldots)$.

Given two interactive algorithms A and B, we denote by $\langle A(x), B(y) \rangle(z)$ the output of $B(y, z)$ when interacting with $A(x, z)$. Note that this output could be a random variable. If we use this notation when A or B is a circuit, we mean that we are considering a circuit that implements a suitable next-message function to interact with the other party of the interaction.

3.1 Indexed relations

An *indexed relation* $\mathcal{R}$ is a set of triples $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ where $\mathbb{i}$ is the index, $\mathbb{x}$ is the instance, and $\mathbb{w}$ is the witness; the corresponding *indexed language* $\mathcal{L}(\mathcal{R})$ is the set of pairs $(\mathbb{i}, \mathbb{x})$ for which there exists a witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$. For example, the indexed relation of satisfiable boolean circuits consists of triples where $\mathbb{i}$ is the description of a boolean circuit, $\mathbb{x}$ is a partial assignment to its input wires, and $\mathbb{w}$ is an assignment to the remaining wires that makes the circuit to output 0. Given a size bound $\mathsf{N} \in \mathbb{N}$, we denote by $\mathcal{R}_\mathsf{N}$ the restriction of $\mathcal{R}$ to triples $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ with $|\mathbb{i}| \leq \mathsf{N}$.

4 Algebraic holographic proofs

We define *algebraic holographic proofs* (AHPs), the notion of proofs that we use. For simplicity, the formal definition below is tailored to univariate polynomials, because our AHP construction is in this setting. The definition can be modified in a straightforward way to consider the general case of multivariate polynomials.

We represent polynomials through the coefficients that define them, as opposed to through their evaluation over a sufficiently large domain (as is typically the case in probabilistic proofs). This definitional choice is due to the fact that we will consider verifiers that may query the polynomials at any location in the field of definition. Moreover, the field of definition itself can be chosen from a given field family, and so we make the field an additional input to all algorithms; this degree of freedom is necessary when combining this component with polynomial commitment schemes. Finally, we consider the setting of *indexed relations* (see Section 3.1), where the verifier's input has two parts, the index and the instance; in the definition below, the verifier receives the index encoded and the instance explicitly.

Formally, an **algebraic holographic proof** (AHP) over a field family $\mathcal{F}$ for an indexed relation $\mathcal{R}$ is specified by a tuple

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

where $\mathsf{k}, \mathsf{s}, \mathsf{d} \colon \{0,1\}^* \to \mathbb{N}$ are polynomial-time computable functions and $\mathbf{I}, \mathbf{P}, \mathbf{V}$ 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.

In the offline phase ("0-th round"), the indexer $\mathbf{I}$ receives as input a field $\mathbb{F} \in \mathcal{F}$ and an index $\mathbb{i}$ for $\mathcal{R}$, and outputs $\mathsf{s}(0)$ polynomials $p_{0,1}, \dots, p_{0,\mathsf{s}(0)} \in \mathbb{F}[X]$ of degrees at most $\mathsf{d}(|\mathbb{i}|, 0, 1), \dots, \mathsf{d}(|\mathbb{i}|, 0, \mathsf{s}(0))$ 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 $\mathbb{i}$.

In the online phase, given an instance $\mathbb{x}$ and witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$, the prover $\mathbf{P}$ receives $(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w})$ and the verifier $\mathbf{V}$ receives $(\mathbb{F}, \mathbb{x})$ and oracle access to the polynomials output by $\mathbf{I}(\mathbb{F}, \mathbb{i})$. The prover $\mathbf{P}$ and the verifier $\mathbf{V}$ interact over $\mathsf{k} = \mathsf{k}(|\mathbb{i}|)$ rounds.

For $i \in [\mathsf{k}]$, in the i-th round of interaction, the verifier $\mathbf{V}$ sends a message $\rho_i \in \mathbb{F}^*$ to the prover $\mathbf{P}$; then the prover $\mathbf{P}$ replies with $\mathsf{s}(i)$ oracle polynomials $p_{i,1}, \dots, p_{i,\mathsf{s}(i)} \in \mathbb{F}[X]$. The verifier may query any of the polynomials it has received any number of times. A query consists of a location $z \in \mathbb{F}$ for an oracle $p_{i,j}$, and its corresponding answer is $p_{i,j}(z) \in \mathbb{F}$. After the interaction, the verifier accepts or rejects.

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 $\tilde{\mathbf{P}}$ is **admissible** for AHP if, on every interaction with the verifier $\mathbf{V}$, it holds that for every round $i \in [\mathsf{k}]$ and oracle index $j \in [\mathsf{s}(i)]$ we have $\deg(p_{i,j}) \leq \mathsf{d}(|\mathbb{i}|, i, j)$. The honest prover $\mathbf{P}$ is required to be admissible under this definition.

We say that AHP has perfect completeness and soundness error ϵ if the following holds.

- **Completeness.** For every field $\mathbb{F} \in \mathcal{F}$ and index-instance-witness tuple $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$, the probability that $\mathbf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w})$ convinces $\mathbf{V}^{\mathbf{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x})$ to accept in the interactive oracle protocol is 1.
- **Soundness.** For every field $\mathbb{F} \in \mathcal{F}$, index-instance pair $(\mathbb{i}, \mathbb{x}) \notin \mathcal{L}(\mathcal{R})$, and admissible prover $\tilde{\mathbf{P}}$, the probability that $\tilde{\mathbf{P}}$ convinces $\mathbf{V}^{\mathbf{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x})$ to accept in the interactive oracle protocol is at most ϵ.

The *proof length* l is the sum of all degree bounds in the offline and online phases, $\mathsf{l}(|\mathbb{i}|) := \sum_{i=0}^{\mathsf{k}(|\mathbb{i}|)} \sum_{j=1}^{\mathsf{s}(i)} \mathsf{d}(|\mathbb{i}|, i, j)$. The intuition for this definition is that in a probabilistic proof each oracle would consist of the evaluation of a polynomial over a domain whose size (in field elements) is linearly related to its degree bound, so that the resulting proof length would be linearly related to the sum of all degree bounds.

The *query complexity* q is the total number of queries made by the verifier to the polynomials. This includes queries to the polynomials output by the indexer and those sent by the prover.

All AHPs that we construct achieve the stronger property of *knowledge soundness* (against admissible provers), and optionally also *zero knowledge*. We define both of these properties below.

Knowledge soundness. We say that AHP has knowledge error ϵ if there exists a probabilistic polynomial-time extractor $\mathbf{E}$ for which the following holds. For every field $\mathbb{F} \in \mathcal{F}$, index $\mathbb{i}$, instance $\mathbb{x}$, and admissible prover $\tilde{\mathbf{P}}$, the probability that $\mathbf{E}^{\tilde{\mathbf{P}}}(\mathbb{F}, \mathbb{i}, \mathbb{x}, 1^{\mathsf{l}(|\mathbb{i}|)})$ outputs $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ is at least the probability that $\tilde{\mathbf{P}}$ convinces $\mathbf{V}^{\mathbf{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x})$ to accept minus ϵ. Here the notation $\mathbf{E}^{\tilde{\mathbf{P}}}$ means that the extractor $\mathbf{E}$ has black-box access to each of the next-message functions that define the interactive algorithm $\tilde{\mathbf{P}}$. (In particular, the extractor $\mathbf{E}$ can "rewind" the prover $\tilde{\mathbf{P}}$.) Note that since $\mathbf{E}$ receives the proof length $\mathsf{l}(|\mathbb{i}|)$ in unary, $\mathbf{E}$ has enough time to receive, and perform efficient computations on, polynomials output by $\tilde{\mathbf{P}}$.

Zero knowledge. We say that AHP has (perfect) zero knowledge with query bound b and query checker $\mathbf{C}$ if there exists a probabilistic polynomial-time simulator $\mathbf{S}$ such that for every field $\mathbb{F} \in \mathcal{F}$, index-instance-witness tuple $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$, and $(\mathsf{b}, \mathbf{C})$-query algorithm $\tilde{\mathbf{V}}$ the random variables $\mathrm{View}(\mathbf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w}), \tilde{\mathbf{V}})$ and $\mathbf{S}^{\tilde{\mathbf{V}}}(\mathbb{F}, \mathbb{i}, \mathbb{x})$, defined below, are identical. Here, we say that an algorithm is $(\mathsf{b}, \mathbf{C})$-query if it makes at most b queries to oracles it has access to, and each query individually leads the checker $\mathbf{C}$ to output "ok".

– $\mathrm{View}(\mathbf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w}), \tilde{\mathbf{V}})$ is the *view* of $\tilde{\mathbf{V}}$, namely, is the random variable $(r, a_1, \ldots, a_q)$ where r is $\tilde{\mathbf{V}}$'s randomness and $a_1, \ldots, a_q$ are the responses to $\tilde{\mathbf{V}}$'s queries determined by the oracles sent by $\mathbf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w})$.
– $\mathbf{S}^{\tilde{\mathbf{V}}}(\mathbb{F}, \mathbb{i}, \mathbb{x})$ is the output of $\mathbf{S}(\mathbb{F}, \mathbb{i}, \mathbb{x})$ when given straightline access to $\tilde{\mathbf{V}}$ ($\mathbf{S}$ may interact with $\tilde{\mathbf{V}}$, *without rewinding*, by exchanging messages with $\tilde{\mathbf{V}}$ and answering any oracle queries along the way), *prepended* with $\tilde{\mathbf{V}}$'s randomness r. Note that r could be of super-polynomial size, so $\mathbf{S}$ cannot sample r on $\tilde{\mathbf{V}}$'s behalf and then output it; instead, as in prior work, we restrict $\mathbf{S}$ to not see r, and prepend r to $\mathbf{S}$'s output.

A special case of interest. We **only consider** AHPs that satisfy the following properties.

– *Public coins:* AHP is *public-coin* if each verifier message to the prover is a uniformly random string of some prescribed length (or an empty string). All verifier queries can be postponed, without loss of generality, to a query phase that occurs after the interactive phase with the prover.

– *Non-adaptive queries:* AHP is *non-adaptive* if all of the verifier's query locations are solely determined by the verifier's randomness and inputs (the field $\mathbb{F}$ and the instance $\mathbb{x}$).

Given these properties, we can view the verifier as two subroutines that execute in the query phase: a query algorithm $\mathbf{Q_V}$ that produces query locations based on the verifier's randomness, and a decision algorithm $\mathbf{D_V}$ that accepts or rejects based on the answers to the queries (and the verifier's randomness). In more detail, $\mathbf{Q_V}$ receives as input the field $\mathbb{F}$, the instance $\mathbb{x}$, and randomness $\rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{k}+1}$, and outputs a query set Q consisting of tuples $((i, j), z)$ to be interpreted as "query $p_{i,j}$ at $z \in \mathbb{F}$"; and $\mathbf{D_V}$ receives as input the field $\mathbb{F}$, the instance $\mathbb{x}$, answers $(v_{((i,j),z)})_{((i,j),z)\in Q}$, and randomness $\rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{k}+1}$, and outputs the decision bit.

While the above properties are not strictly necessary for the compiler that we describe in the full version, all "natural" protocols that we are aware of (including those that we construct in this paper) satisfy these properties, and so we restrict our attention to public-coin non-adaptive protocols for simplicity.

5 AHP for constraint systems

We construct an AHP for *rank-1 constraint satisfiability* (R1CS) that has linear proof length and constant query complexity. Below we define the indexed relation that represents this problem, and then state our result.

Definition 1 (R1CS indexed relation). *The indexed relation $\mathcal{R}_{\mathrm{R1CS}}$ is the set of all triples*

$$(\mathbb{i}, \mathbb{x}, \mathbb{w}) = \big((\mathbb{F}, H, K, A, B, C), x, w\big)$$

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

Theorem 1. *There exists an AHP for the indexed relation $\mathcal{R}_{\mathrm{R1CS}}$ that is a zero knowledge proof of knowledge with the following features. The indexer uses $O(|K| \log |K|)$ field operations and outputs $O(|K|)$ field elements. The prover and verifier exchange 7 messages. To achieve zero knowledge against b queries (with a query checker $\mathbf{C}$ that rejects queries in H), the prover uses $O((|K|+\mathsf{b}) \log(|K|+\mathsf{b}))$ field operations and outputs a total of $O(|H|+\mathsf{b})$ field elements. The verifier makes $O(1)$ queries to the encoded index and to the prover's messages, has soundness error $O((|K| + \mathsf{b})/|\mathbb{F}|)$, and uses $O(|x| + \log |K|)$ field operations.*

Remark 1 (restrictions on domains). Our protocol uses the univariate sumcheck of [BCR+19] as a subroutine, and in particular inherits the requirement that the domains H and K must be additive or multiplicative subgroups of the field $\mathbb{F}$. For simplicity, in our descriptions we use multiplicative subgroups because we use this case in our implementation; the case of additive subgroups involves only minor modifications. Moreover, the arithmetic complexities for the indexer and prover stated in Theorem 1 assume that the domains H and K are "FFT-friendly" (e.g., they have smooth sizes); this is not a requirement, since in general the arithmetic complexities will be that of an FFT over the domains H and K. Note that we can assume without loss of generality that $|H| = O(|K|)$, for otherwise (if $|K| < |H|/3$) then are empty rows or columns across the matrices that we can drop and reduce their size. Finally, we assume that $|H| \leq |\mathbb{F}|/2$.

This section is organized as follows: in Section 5.1 we introduce algebraic notations and facts used in this section, and then in Section 5.2 we describe an AHP for checking linear relations. Due to space constraints, we describe how to use this latter AHP to construct our AHP for R1CS only in the full version.

Throughout we assume that H and K come equipped with bijections $\phi_{H} : H \to [|H|]$ and $\phi_{K} : K \to [|K|]$ that are computable in linear time. Moreover, we define the two sets $H[\leq k] := \{\kappa \in H \colon 1 \leq \phi_{H}(\kappa) \leq k\}$ and $H[> k] := \{\kappa \in H \colon \phi_{H}(\kappa) > k\}$ to denote the first k elements in H and the remaining elements, respectively. We can then write that $x \in \mathbb{F}^{H[\leq |x|]}$ and $w \in \mathbb{F}^{H[>|x|]}$.

5.1 Algebraic preliminaries

Polynomial encodings. For a finite field $\mathbb{F}$, subset $S \subseteq \mathbb{F}$, and function $f \colon S \to \mathbb{F}$ we denote by $\hat{f}$ the (unique) univariate polynomial over $\mathbb{F}$ with degree less than $|S|$ such that $\hat{f}(a) = f(a)$ for every $a \in S$. We sometimes abuse notation and write $\hat{f}$ to denote *some* polynomial that agrees with f on S, which need not equal the (unique) such polynomial of smallest degree.

Vanishing polynomials. For a finite field $\mathbb{F}$ and subset $S \subseteq \mathbb{F}$, we denote by v_S the unique non-zero monic polynomial of degree at most $|S|$ that is zero everywhere on S; v_S is called the *vanishing polynomial* of S. If S is an additive or multiplicative coset in $\mathbb{F}$ then v_S can be evaluated in $\mathrm{polylog}(|S|)$ field operations. For example, if S is a multiplicative subgroup of $\mathbb{F}$ then $v_S(X) = X^{|S|} - 1$ and, more generally, if S is a ξ-coset of a multiplicative subgroup S_0 (namely, $S = \xi S_0$) then $v_S(X) = \xi^{|S|} v_{S_0}(X/\xi) = X^{|S|} - \xi^{|S|}$; in either case, v_S can be evaluated in $O(\log |S|)$ field operations.

Derivative of vanishing polynomials. We rely on various properties of a bivariate polynomial u_S introduced in [BCG+19]. For a finite field $\mathbb{F}$ and subset $S \subseteq \mathbb{F}$, we define

$$u_S(X, Y) := \frac{v_S(X) - v_S(Y)}{X - Y} \quad ,$$

which is a polynomial of individual degree $|S| - 1$ because $X - Y$ divides $X^i - Y^i$ for any positive integer i. Note that $u_S(X, X)$ is the formal derivative of the vanishing polynomial $v_S(X)$. The bivariate polynomial $u_S(X, Y)$ satisfies two useful algebraic properties. First, the univariate polynomials $(u_S(X, a))_{a \in S}$ are linearly independent, and $u_S(X, Y)$ is their (unique) low-degree extension. Second, $u_S(X, Y)$ vanishes on the square $S \times S$ except for on the diagonal, where it takes on the (non-zero) values $(u_S(a, a))_{a \in S}$.

If S is an additive or multiplicative coset in $\mathbb{F}$, $u_S(X, Y)$ can be evaluated at any $(\alpha, \beta) \in \mathbb{F}^2$ in $\mathrm{polylog}(|S|)$ field operations because in this case both v_S (and its derivative) can be evaluated in polylog($|S|$) field operations. For example, if S is a multiplicative subgroup then $u_S(X,Y) = (X^{|S|} - Y^{|S|})/(X - Y)$ and $u_S(X,X) = |S|X^{|S|-1}$, so both can be evaluated in $O(\log|S|)$ field operations.

Univariate sumcheck for subgroups. Prior work [BCR+19] shows that, given a multiplicative subgroup S of $\mathbb{F}$, a polynomial $f(X)$ sums to σ over S if and only if $f(X)$ can be written as $h(X)v_S(X) + Xg(X) + \sigma/|S|$ for some $h(X)$ and $g(X)$ with $\deg(g) < |S| - 1$. This can be viewed as a univariate sumcheck protocol, and we shall rely on it throughout this section.

5.2 AHP for the lincheck problem

The *lincheck problem* for univariate polynomials considers the task of deciding whether two polynomials encode vectors that are linearly related in a prescribed way. In more detail, the problem is parametrized by a field $\mathbb{F}$, two subsets H and K of $\mathbb{F}$, and a matrix $M \in \mathbb{F}^{H\times H}$ with $|K| \geq \|M\| > 0$. Given oracle access to two low-degree polynomials $f_1, f_2 \in \mathbb{F}^{<d}[X]$, the problem asks to decide whether for every $a \in H$ it holds that $f_1(a) = \sum_{b\in H} M_{a,b} \cdot f_2(b)$, by asking a small number of queries to f_1 and f_2. The matrix M thus prescribes the linear relations that relate the values of f_1 and f_2 on H.

Ben-Sasson et al. [BCR+19] solve this problem by reducing the lincheck problem to a sumcheck problem, and then reducing the sumcheck problem to low-degree testing (of univariate polynomials). In particular, this prior work achieves a 2-message algebraic *non-holographic* protocol that solves the lincheck problem with linear proof length and constant query complexity. In this section we show how to achieve a 6-message algebraic *holographic* protocol, again with linear proof length and constant query complexity. In Section 5.2.1 we describe the indexer algorithm, in Section 5.2.2 we describe the prover and verifier algorithms. In the full version we provide a diagram that summarizes the protocol, and provide completeness, soundness, and efficiency analyses.

5.2.1 Offline phase: encoding the linear relation

The indexer $\mathbf{I}$ for the lincheck problem receives as input a field $\mathbb{F}$, two subsets H and K of $\mathbb{F}$, and a matrix $M \in \mathbb{F}^{H\times H}$ with $|K| \geq \|M\|$. The non-zero entries of M are assumed to be presented in some canonical order (e.g., row-wise or column-wise). The output of $\mathbf{I}$ is three univariate polynomials $\hat{\mathsf{row}}, \hat{\mathsf{col}}, \hat{\mathsf{val}}$ over $\mathbb{F}$ of degree less than $|K|$ such that the following polynomial is a low-degree extension of M:

$$\hat{M}(X,Y) := \sum_{\kappa\in K} u_H(X, \hat{\mathsf{row}}(\kappa)) u_H(Y, \hat{\mathsf{col}}(\kappa)) \hat{\mathsf{val}}(\kappa) \ . \tag{1}$$

The three three aforementioned polynomials are the (unique) low-degree extensions of the three functions $\mathsf{row}, \mathsf{col}, \mathsf{val} \colon K \to \mathbb{F}$ that respectively represent the row index, column index, and value of the non-zero entries of the matrix M. In more detail, for every $\kappa \in K$ with $1 \leq \phi_K(\kappa) \leq \|M\|$:

- $\mathsf{row}(\kappa) := \phi_H^{-1}(t_\kappa)$ where t_κ is the row index of the $\phi_K(\kappa)$-th nonzero entry in M;

– $\mathsf{col}(\kappa) := \phi_H^{-1}(t_\kappa)$ where t_κ is the column index of the $\phi_K(\kappa)$-th nonzero entry in M;
– $\mathsf{val}(\kappa)$ is the value of the $\phi_K(\kappa)$-th nonzero entry in M, divided by $u_H(\mathsf{row}(\kappa), \mathsf{row}(\kappa))u_H(\mathsf{col}(\kappa), \mathsf{col}(\kappa))$.

Also, $\mathsf{val}(\kappa)$ returns the element 0 for every $\kappa \in K$ with $\phi_K(\kappa) > \|M\|$, while $\mathsf{row}(\kappa)$ and $\mathsf{col}(\kappa)$ return an arbitrary element in H for such κ. The evaluation tables of these functions can be found in $O(|K| \log |H|)$ operations, from which interpolation yields the desired polynomials in $O(|K| \log |K|)$ operations.

Recall from Section 5.1 that the bivariate polynomial $u_H(X, Y)$ vanishes on the square $H \times H$ except for on the diagonal, where it takes on the (non-zero) values $(u_H(a, a))_{a \in H}$. By construction of the polynomials $\hat{\mathsf{row}}, \hat{\mathsf{col}}, \hat{\mathsf{val}}$, the polynomial $\hat{M}(X, Y)$ agrees with the matrix M everywhere on the domain $H \times H$. The individual degree of $\hat{M}(X, Y)$ is less than $|H|$. Thus, $\hat{M}$ is the unique low-degree extension of M.

We rewrite the polynomial $\hat{M}(X, Y)$ in a form that will be useful later:

Claim 1.

$$\hat{M}(X, Y) = \sum_{\kappa \in K} \frac{v_H(X)}{(X - \hat{\mathsf{row}}(\kappa))} \cdot \frac{v_H(Y)}{(Y - \hat{\mathsf{col}}(\kappa))} \cdot \hat{\mathsf{val}}(\kappa) \ . \tag{2}$$

Proof. Note that $v_H(\hat{\mathsf{row}}(\kappa)) = v_H(\hat{\mathsf{col}}(\kappa)) = 0$ for every $\kappa \in K$ because $\hat{\mathsf{row}}(X)$ and $\hat{\mathsf{col}}(X)$ map K to H and v_H vanishes on H. Therefore:

$$\begin{aligned}
\hat{M}(X, Y) &= \sum_{\kappa \in K} u_H(X, \hat{\mathsf{row}}(\kappa)) \cdot u_H(Y, \hat{\mathsf{col}}(\kappa)) \cdot \hat{\mathsf{val}}(\kappa) \\
&= \sum_{\kappa \in K} \frac{v_H(X) - v_H(\hat{\mathsf{row}}(\kappa))}{X - \hat{\mathsf{row}}(\kappa)} \cdot \frac{v_H(Y) - v_H(\hat{\mathsf{col}}(\kappa))}{Y - \hat{\mathsf{col}}(\kappa)} \cdot \hat{\mathsf{val}}(\kappa) \\
&= \sum_{\kappa \in K} \frac{v_H(X)}{(X - \hat{\mathsf{row}}(\kappa))} \cdot \frac{v_H(Y)}{(Y - \hat{\mathsf{col}}(\kappa))} \cdot \hat{\mathsf{val}}(\kappa) \ .
\end{aligned}$$

□

5.2.2 Online phase: proving and verifying the linear relation

The prover $\mathbf{P}$ for the lincheck problem receives as input a field $\mathbb{F}$, two subsets H and K of $\mathbb{F}$, a matrix $M \in \mathbb{F}^{H \times H}$ with $|K| \geq \|M\|$, and two polynomials $f_1, f_2 \in \mathbb{F}^{<d}[X]$. The verifier $\mathbf{V}$ for the lincheck problem receives as input the field $\mathbb{F}$ and two subsets H and K of $\mathbb{F}$; $\mathbf{V}$ also has oracle access to the polynomials $\hat{\mathsf{row}}, \hat{\mathsf{col}}, \hat{\mathsf{val}}$ output by the indexer $\mathbf{I}$ invoked on appropriate inputs.

The protocol begins with a reduction from a lincheck problem to a sumcheck problem: $\mathbf{V}$ samples a random element $\alpha \in \mathbb{F}$ and sends it to $\mathbf{P}$. Indeed, letting $r(X, Y)$ denote the polynomial $u_H(X, Y)$, $\mathbf{P}$ is left to convince $\mathbf{V}$ that the following univariate polynomial sums to 0 on H:

$$q_1(X) := r(\alpha, X) f_1(X) - r_M(\alpha, X) f_2(X) \tag{3}$$

where $r_M(X,Y) := \sum_{\kappa \in H} r(X,\kappa)\hat{M}(\kappa,Y)$.

We rely on the univariate sumcheck protocol for this step: **P** sends to **V** the polynomials $g_1(X)$ and $h_1(X)$ such that $q_1(X) = h_1(X)v_H(X) + Xg_1(X)$. In order to check this polynomial identity, **V** samples a random element $\beta_1 \in \mathbb{F}$ with the intention of checking the identity at $X := \beta_1$. For the right-hand side, **V** queries g_1 and h_1 at β_1, and then evaluates $h_1(\beta_1)v_H(\beta_1)+\beta_1 g_1(\beta_1)$ in $O(\log|H|)$ operations. For the left-hand side, **V** queries f_1 and f_2 at β_1 and then needs to ask help from **P** to evaluate $r(\alpha,\beta_1)f_1(\beta_1) - r_M(\alpha,\beta_1)f_2(\beta_1)$. The reason is that while $r(\alpha,\beta_1)$ is easy to evaluate (it requires $O(\log|H|)$ operations), $r_M(\alpha,\beta_1) = \sum_{\kappa\in H} r(\alpha,\kappa)\hat{M}(\kappa,\beta_1)$ in general requires $\Omega(|H||K|)$ operations.

We thus rely on the univariate sumcheck protocol again. We define

$$q_2(X) := r(\alpha,X)\hat{M}(X,\beta_1) \tag{4}$$

V sends β_1 to **P**, and then **P** replies with the sum $\sigma_2 := \sum_{\kappa\in H} r(\alpha,\kappa)\hat{M}(\kappa,\beta_1)$ and the polynomials $g_2(X)$ and $h_2(X)$ such that $q_2(X) = h_2(X)v_H(X)+Xg_2(X)+\sigma_2/|H|$. In order to check this polynomial identity, **V** samples a random element $\beta_2 \in \mathbb{F}$ with the intention of checking the identity at $X := \beta_2$. For the right-hand side, **V** queries g_2 and h_2 at β_2, and then evaluates $h_2(\beta_2)v_H(\beta_2) + \beta_2 g_2(\beta_2) + \sigma_2/|H|$ in $O(\log|H|)$ operations. To evaluate the left-hand side, however, **V** needs to ask help from **P**. The reason is that while $r(\alpha,\beta_2)$ is easy to evaluate (it requires $O(\log|H|)$ operations), $\hat{M}(\beta_2,\beta_1)$ in general requires $\Omega(|K|)$ operations.

We thus rely on the univariate sumcheck protocol (yet) again: **V** sends β_2 to **P**, and then **P** replies with the value $\sigma_3 := \hat{M}(\beta_2,\beta_1)$, which the verifier must check. Note though that we *cannot* use the sumcheck protocol directly to compute the sum obtained from Eq. (1):

$$\hat{M}(\beta_2,\beta_1) = \sum_{\kappa\in K} u_H(\beta_2,\hat{\mathsf{row}}(\kappa))u_H(\beta_1,\hat{\mathsf{col}}(\kappa))\hat{\mathsf{val}}(\kappa) \ .$$

The reason is because the degree of the above addend, if we replace κ with an indeterminate, is $\Omega(|H||K|)$, which means that the degree of the polynomial h_3 sent as part of a sumcheck protocol also has degree $\Omega(|H||K|)$, which is not within our budget of an AHP with proof length $O(|H|+|K|)$. Instead, we make the minor modification that in the earlier rounds β_1 and β_2 are sampled from $\mathbb{F}\setminus H$ instead of $\mathbb{F}$, and we will leverage the sumcheck protocol to verify the equivalent (well defined) expression from Eq. (2):

$$\hat{M}(\beta_2,\beta_1) = \sum_{\kappa\in K} \frac{v_H(\beta_2)v_H(\beta_1)\hat{\mathsf{val}}(\kappa)}{(\beta_2-\hat{\mathsf{row}}(\kappa))(\beta_1-\hat{\mathsf{col}}(\kappa))} \ .$$

This may appear to be an odd choice, because if we replace κ with an indeterminate in the sum above, we obtain a rational function that is (in general) *not a polynomial*, and so does not immediately fit the sumcheck protocol. Nevertheless, we are still able to use the sumcheck protocol with it, as we now explain.

Define $f_3(X)$ to be the (unique) polynomial of degree less than $|K|$ such that

$$\forall\, \kappa\in K\ ,\ f_3(\kappa) = \frac{v_H(\beta_2)v_H(\beta_1)\hat{\mathsf{val}}(\kappa)}{(\beta_2-\hat{\mathsf{row}}(\kappa))(\beta_1-\hat{\mathsf{col}}(\kappa))} \ . \tag{5}$$

The prover computes the polynomials $g_3(X)$ and $h_3(X)$ such that

$$\begin{aligned} f_3(X) &= Xg_3(X) + \sigma_3/|K| \ , \\ h_3(X)v_K(X) &= v_H(\beta_2)v_H(\beta_1)\hat{\mathsf{val}}(X) - (\beta_2 - \hat{\mathsf{row}}(X))(\beta_1 - \hat{\mathsf{col}}(X))f_3(X) \ . \end{aligned}$$

The first equation demonstrates that f_3 sums to σ_3 over K, and the second equation demonstrates that f_3 agrees with the correct addends over K. These two equations can be combined in a single equation that involves only $g_3(X)$ and $h_3(X)$:

$$\begin{aligned} h_3(X)v_K(X) = &\ v_H(\beta_2)v_H(\beta_1)\hat{\mathsf{val}}(X) \\ &- (\beta_2 - \hat{\mathsf{row}}(X))(\beta_1 - \hat{\mathsf{col}}(X))(Xg_3(X) + \sigma_3/|K|) \ . \end{aligned}$$

The prover thus only sends the two polynomials $g_3(X)$ and $h_3(X)$. In order to check this polynomial identity, **V** samples a random element $\beta_3 \in \mathbb{F}$ with the intention of checking the identity at $X := \beta_3$. Then **V** queries g_3, h_3, $\hat{\mathsf{row}}$, $\hat{\mathsf{col}}$, $\hat{\mathsf{val}}$ at β_3, and then evaluates $v_H(\beta_2)v_H(\beta_1)\hat{\mathsf{val}}(\beta_3) - (\beta_2 - \hat{\mathsf{row}}(\beta_3))(\beta_1 - \hat{\mathsf{col}}(\beta_3))(\beta_3 g_3(\beta_3) + \sigma_3/|K|) = h_3(\beta_3)v_K(\beta_3)$ in $O(\log |K|)$ operations.

If this third test passes then **V** can use the value σ_3 in place of $\hat{M}(\beta_2, \beta_1)$ to finish the second test. If this latter passes, **V** can in turn use the value σ_2 in place of $r_M(\alpha, \beta_1)$ to finish the first test.

Acknowledgments

This research was supported by: an Engineering and Physical Sciences Research Council grant (EP/N028104/1), a Google Faculty Award, the RISELab at UC Berkeley, and donations from the Ethereum Foundation and the Interchain Foundation. The authors thank Dev Ojha and Nicholas Spooner for identifying and helping to fix an error in a prior version of our AHP for the lincheck problem.

References

[AHIV17] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup". In: CCS '17.

[BBB+18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: S&P '18.

[BBC+17] E. Ben-Sasson, I. Bentov, A. Chiesa, A. Gabizon, D. Genkin, M. Hamilis, E. Pergament, M. Riabzev, M. Silberstein, E. Tromer, and M. Virza. "Computational integrity with a public random string from quasi-linear PCPs". In: EUROCRYPT '17.

[BBHR19] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. "Scalable Zero Knowledge with No Trusted Setup". In: CRYPTO '19.

[BCC+16] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. "Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting". In: EUROCRYPT '16.

[BCG+15] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and M. Virza. "Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs". In: S&P '15.

[BCG+19] E. Ben-Sasson, A. Chiesa, L. Goldberg, T. Gur, M. Riabzev, and N. Spooner. “Linear-Size Constant-Query IOPs for Delegating Computation”. In: TCC ’19.

[BCGV16] E. Ben-Sasson, A. Chiesa, A. Gabizon, and M. Virza. “Quasilinear-Size Zero Knowledge from Linear-Algebraic PCPs”. In: TCC ’16-A.

[BCI+13] N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth. “Succinct Non-Interactive Arguments via Linear Interactive Proofs”. In: TCC ’13.

[BCR+19] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. “Aurora: Transparent Succinct Arguments for R1CS”. In: EUROCRYPT ’19.

[BCS16] E. Ben-Sasson, A. Chiesa, and N. Spooner. “Interactive Oracle Proofs”. In: TCC ’16-B.

[BCTV14a] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. “Scalable Zero Knowledge via Cycles of Elliptic Curves”. In: CRYPTO ’14.

[BCTV14b] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. “Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture”. In: USENIX Security ’14.

[BFLS91] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. “Checking computations in polylogarithmic time”. In: STOC ’91.

[BG12] S. Bayer and J. Groth. “Efficient Zero-Knowledge Argument for Correctness of a Shuffle”. In: EUROCRYPT ’12.

[BGG17] S. Bowe, A. Gabizon, and M. Green. “A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK”. ePrint Report 2017/602.

[BGM17] S. Bowe, A. Gabizon, and I. Miers. “Scalable Multi-party Computation for zk-SNARK Parameters in the Random Beacon Model”. ePrint Report 2017/1050.

[BISW17] D. Boneh, Y. Ishai, A. Sahai, and D. J. Wu. “Lattice-Based SNARGs and Their Application to More Efficient Obfuscation”. In: EUROCRYPT ’17.

[BISW18] D. Boneh, Y. Ishai, A. Sahai, and D. J. Wu. “Quasi-Optimal SNARGs via Linear Multi-Prover Interactive Proofs”. In: EUROCRYPT ’18.

[BS08] E. Ben-Sasson and M. Sudan. “Short PCPs with Polylog Query Complexity”. In: *SIAM J. Comp.* (2008).

[CD98] R. Cramer and I. Damgård. “Zero-Knowledge Proofs for Finite Field Arithmetic; or: Can Zero-Knowledge be for Free?” In: CRYPTO ’98.

[CFS17] A. Chiesa, M. A. Forbes, and N. Spooner. “A Zero Knowledge Sumcheck and its Applications”. ePrint Report 2017/305.

[FKL18] G. Fuchsbauer, E. Kiltz, and J. Loss. “The Algebraic Group Model and its Applications”. In: CRYPTO ’18.

[FS86] A. Fiat and A. Shamir. “How to prove yourself: practical solutions to identification and signature problems”. In: CRYPTO ’86.

[Gab19] A. Gabizon. “Improved prover efficiency and SRS size in a Sonic-like system”. ePrint Report 2019/601.

[GGPR13] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. “Quadratic Span Programs and Succinct NIZKs without PCPs”. In: EUROCRYPT ’13.

[GKM+18] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. “Updatable and Universal Common Reference Strings with Applications to zk-SNARKs”. In: CRYPTO ’18.

[GKR15] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. “Delegating Computation: Interactive Proofs for Muggles”. In: *JACM* (2015).

[Gro10] J. Groth. “Short Pairing-Based Non-interactive Zero-Knowledge Arguments”. In: ASIACRYPT ’10.

[Gro16] J. Groth. “On the Size of Pairing-Based Non-interactive Arguments”. In: EUROCRYPT ’16.

[GWC19] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. “PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge”. ePrint Report 2019/953.

[Kil92] J. Kilian. “A note on efficient zero-knowledge proofs and arguments”. In: STOC ’92.

[KZG10] A. Kate, G. M. Zaverucha, and I. Goldberg. “Constant-Size Commitments to Polynomials and Their Applications”. In: ASIACRYPT ’10.

[Lip12] H. Lipmaa. “Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments”. In: TCC ’12.

[MBKM19] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. “Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updateable Structured Reference Strings”. In: CCS ’19.

[PST13] C. Papamanthou, E. Shi, and R. Tamassia. “Signatures of Correct Computation”. In: TCC ’13.

[RRR16] O. Reingold, R. Rothblum, and G. Rothblum. “Constant-Round Interactive Proofs for Delegating Computation”. In: STOC ’16.

[Set19] S. Setty. “Spartan: Efficient and general-purpose zkSNARKs without trusted setup”. ePrint Report 2019/550.

[WTS+18] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish. “Doubly-Efficient zkSNARKs Without Trusted Setup”. In: S&P ’18.

[XZZ+19] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song. “Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation”. In: CRYPTO ’19.

[Zcash] “Zcash”. `https://z.cash/`.

[ZcashMPC] “The Zcash Ceremony”. `https://z.cash/blog/the-design-of-the-ceremony.html`.

[ZGK+17a] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou. “A Zero-Knowledge Version of vSQL”. ePrint Report 2017/1146.

[ZGK+17b] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou. “vSQL: Verifying Arbitrary SQL Queries over Dynamic Outsourced Databases”. In: S&P ’17.

[ZGK+18] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou. “vRAM: Faster Verifiable RAM with Program-Independent Preprocessing”. In: S&P ’18.