Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings

Mary Maller
mary.maller.15@ucl.ac.uk
University College London

Sean Bowe
sean@z.cash
Zcash Company

Markulf Kohlweiss
mkohlwei@ed.ac.uk
University of Edinburgh
IOHK

Sarah Meiklejohn
s.meiklejohn@ucl.ac.uk
University College London

ABSTRACT

Ever since their introduction, zero-knowledge proofs have become an important tool for addressing privacy and scalability concerns in a variety of applications. In many systems each client downloads and verifies every new proof, and so proofs must be small and cheap to verify. The most practical schemes require either a trusted setup, as in (pre-processing) zk-SNARKs, or verification complexity that scales linearly with the complexity of the relation, as in Bulletproofs. The structured reference strings required by most zk-SNARK schemes can be constructed with multi-party computation protocols, but the resulting parameters are specific to an individual relation. Groth et al. discovered a zk-SNARK protocol with a universal and updatable structured reference string, but the string scales quadratically in the size of the supported relations.

Here we describe a zero-knowledge SNARK, Sonic, which supports a universal and continually updatable structured reference string that scales linearly in size. Sonic proofs are constant size, and in the batch verification context the marginal cost of verification is comparable with the most efficient SNARKs in the literature. We also describe a generally useful technique in which untrusted "helpers" can compute advice that allows batches of proofs to be verified more efficiently.

1 INTRODUCTION

In the decades since their introduction, zero-knowledge proofs have been used to support a wide variety of potential applications, ranging from verifiable outsourced computation [11, 16, 24, 58] to anonymous credentials [6, 27, 28, 32, 39], with a multitude of other settings that also require a balance between privacy and integrity [17, 19, 29, 31, 36]. In recent years, cryptocurrerencies have been one increasingly popular real-world application [10, 44, 52, 56], with general zero-knowledge protocols now deployed in both Zcash and Ethereum. In the cryptocurrency setting it is common for clients to download and verify every transaction published to the network. This means that small proof sizes and fast verification time are important for the practical deployment of zero-knowledge protocols. There are several practical schemes from which to choose, with a vast space of tradeoffs in performance and cryptographic assumptions.

Currently, the most attractive proving system from the verifier's perspective is a (pre-processing) succinct non-interactive argument of knowledge, or zk-SNARK for short, which has a small constant proof size and constant-time verification costs even for arbitrarily large relations. The most efficient scheme described in the literature is a zk-SNARK by Groth [45] which contains only three group elements. These schemes require a trusted setup, a pairing-friendly elliptic curve, and rely on strong assumptions.

In contrast, proving systems such as Bulletproofs [26] do not require a trusted setup and depend on weaker assumptions. Unfortunately, although its proof sizes scale logarithmically with the relation size, Bulletproof verification time scales linearly, even when applying batching techniques. As a result, Bulletproofs are ideal for simpler relations.

Although zk-SNARKs have been deployed in applications that can tolerate the stronger assumptions, including for a private payment protocol in Zcash, the trusted setup has emerged as a barrier for the deployment of these proving systems. If the setup is compromised in Zcash, for example, an attacker could create counterfeit money without detection. It is possible to reduce risk by performing the setup with a multi-party computation (MPC) protocol, with the property that only one participant must be honest for the final parameters to be secure [25, 61]. However, the resulting parameters are specific to the individual relation, and so each distinct application must perform its own setup. Applications must also perform a new setup each time their construction changes, even for minor optimisations or bug fixes.

Groth et al. [46] recently proposed a zk-SNARK scheme with a *universal* structured reference string (SRS[1]) that allows a single setup to support all circuits of some bounded size. Moreover, the SRS is *updatable*, meaning an open set of participants can contribute secret randomness to it indefinitely. Although this is still a trusted setup, it increases confidence in the security of the parameters as only one previous contributor must have destroyed their secret randomness in order for the SRS to be secure.

In terms of efficiency, however, while the construction due to Groth et al. does have constant-size proofs and constant-time verification, it requires an SRS that is quadratic with respect to the number of multiplication gates in the supported arithmetic circuits. Moreover, updating the SRS requires a quadratic number of group exponentiations, and verifying the updates requires a linear number of pairings. Finally, while the prover and verifier only need a linear-size, circuit-specific string for a given fixed relation (rather than the whole SRS), deriving this from the SRS requires an expensive

[1] "Structured reference string" is the recommended language to use when referring to what was once called a "common reference string" [62].

Gaussian elimination process. In a concrete setting such as Zcash, which has a circuit with 2^{17} multiplication gates , the SRS would be on the order of terabytes and is thus prohibitively expensive.

1.1 Our Contributions

We present Sonic, a new zk-SNARK for general arithmetic circuit satisfiability. Sonic requires a trusted setup, but unlike conventional SNARKs the structured reference string supports all circuits (up to a given size bound) and is also updatable, so that it can be continually strengthened. This addresses many of the practical challenges and risks surrounding such setups. Sonic's structured reference string is linear in size with respect to the size of supported circuits, as opposed to the scheme by Groth et al. which scales quadratically. The structured reference string in Sonic also does not need to be specialized or pre-processed for a given circuit. This makes a large, distributed and never-ending setup process a practical reality.

Proof verification in Sonic consists of a constant number of pairing checks. Unlike other zk-SNARKs, all proof elements are in the same source group, which has several advantages. Most significantly, when verifying many proofs at the same time, the pairing operations only need to be computed once. Thus the marginal costs stem solely from a handful of exponentiations in the group. We also remove the requirement for operations in the second source group, which are typically more expensive.

Sonic's verification includes checking the evaluation of a sparse bivariate polynomial in the scalar field. We introduce a method to check this evaluation succinctly (given a circuit-dependent precomputation) and thus maintain our zk-SNARK properties. Our proof of correct evaluation introduces a new permutation argument and a *grand-product* argument.

Additionally Sonic can achieve better concrete efficiency if an untrusted "helper" party aggregates a batch of proofs. This batching operation computes advice to speed up the verifier. In a blockchain application, this helper could be a "miner" client that already processes and verifies transactions for inclusion in the next block.

We define security in this setting in Section 3, and present and prove secure the regular usage of Sonic in Section 6 and Section 7. In Section 8 we present the more efficient version of Sonic which is helper-assisted. Finally, we implement our protocol and discuss its performance in Section 9, demonstrating verification times that are competitive with state-of-the-art pre-processing zk-SNARKs for typical arithmetic circuits. For any size of circuit proof sizes are 256 bytes and the verification times for circuits with small instances and arbitrarily sized witnesses are approximately 0.7ms (assuming there are helpers).

1.2 Our Techniques

The goal of Sonic is to provide zero-knowledge arguments for the satisfiability of constraint systems representing hard languages. Sonic defines its constraint system with respect to the two-variate polynomial equation used in Bulletproofs that was designed by Bootle et al. [22]. In the Bulletproofs polynomial equation, there is one polynomial that is determined by the statement and a second that is determined by the constraints. The polynomial determined by the statement $\boldsymbol{a}$ is given by

$$\sum_{i,j} a_{i,j} X^i Y^j$$

i.e. each element of the statement is used to scale a monomial in the overall polynomial. For this reason, an SRS that only contains hidden monomial evaluations suffices for committing to the statement. Groth et. al. [46] showed that an SRS that contains monomials is updatable. The second polynomial that is determined by the constraints is known to the verifier. We use this knowledge to allow the verifier to obtain evaluations of the polynomial while avoiding putting constraint-specific secrets in the SRS.

To commit to our polynomials, we use a variation of a polynomial commitment scheme by Kate et al. [50]. We prove the commitment scheme secure in the algebraic group model [37], which is a model that lies somewhere between the standard model and the generic group model. This security proof does not follow from the initial reductions by Kate et al. because we additionally need to show that the adversary can extract the committed polynomials. Kate et al.'s scheme has constant size and verification time, but is designed for single-variate polynomials, whereas our polynomials are two-variate. To account for this, we hide only one evaluation point in the reference string. The polynomial defining the statement is of a special form where it can be committed to using a univariate scheme; i.e., it is of the form

$$\sum_{i} a_i X^i Y^i$$

The prover first commits to the polynomial defining the statement, and then the second evaluation point y is determined in the clear. The prover can then commit to other polynomials of the form

$$\sum_{i,j} t_{i,j} X^i y^j$$

using a univariate scheme.

When the prover and verifier both know a two-variate polynomial which the verifier wants to calculate, this work can be unloaded onto the prover. In our scheme we utilise this observation by placing the work of computing the polynomial specifying the constraints onto the prover. The prover then has to show that the polynomial has been calculated correctly. We provide two methods of achieving this. In the first, we simply provide a proof that the evaluation is correct. While asymptotically preferable, concretely this proof is 3 times the size of our second method. In this scenario, many proofs are calculated by many provers, and then a helper party calculates the circuit-specifying polynomial for each proof. The circuit-specifying polynomial contains no private information, so the helper can be run by anyone. The helper party then proves that they have calculated all of the polynomials correctly at the same time, which they can do succinctly with a one-off circuit-dependent cost that can be amortised over many proofs.

2 RELATED WORK

An efficiency comparison of all of the schemes we discuss is provided in Table 1. We also give a more concrete efficiency comparison in Table 2 of Sonic against the fastest zk-SNARK in the literature (Groth 2016 [45]) and Bulletproofs [26].

Scheme	Runtime		Size		PQ?	Universal?	Untrusted setup?	Assumptions
	Prover	Verifier	CRS	Proof				
Hyrax	$d(hc + c\log c) + w$	$\ell + d(h + \log(hc))$	$\sqrt{w}$	$d\log(hc) + \sqrt{w}$	○	●	●	DL
ZK vSQL	$n\log(c)$	$\ell + d$ polylog(n)	$\log(n)$	$d\log(c)$	○	●	◐	q-type, KOE
Ligero	$n\log(n)$	$c\log(c) + h\log(h)$	0	$\sqrt{n}$	◐	●	●	CRHF
Bootle et al. [23]	n	n	0	$\sqrt{n}$	◐	●	●	CRHF
Baum et. al. [4]	$n\log(n)$	n	$\sqrt{n}$	$\sqrt{n\log(n)}$	◐	●	●	SIS
STARKs	n polylog(n)	polylog(n)	0	$\log^2(n)$	◐	●	●	CRHF
Aurora	$n\log(n)$	n	0	$\log^2(n)$	◐	●	●	CRHF
Bulletproofs	$n\log(n)$	$n\log(n)$	n	$\log(n)$	○	●	●	DL
SNARKs	$n\log(n)$	ℓ	n	1	○	○	○	q-type, KOE
Groth et al. [46]	$n\log(n)$	ℓ	n^2	1	○	●	◐	q-type, KOE
This work	$n\log(n)$	ℓ	n	1	○	●	◐	AGM

Table 1: Asymptotic efficiency comparison of zero-knowledge proofs for arithmetic circuits. Here n is the number of gates, d is the depth of the circuit, h is the width of the subcircuits, c is the number of copies of the subcircuits, ℓ is the size of the instance, and w is the size of the witness. An empty circle denotes that the scheme does not have this property and a full circle denotes that the scheme does have this property. A half circle for post-quantum security denotes that it is feasibly post-quantum secure but that there is no proof. A half circle for untrusted setup denotes that the scheme is updatable. DL stands for discrete log, CRHF stands for collision-resistant hash functions, KOE stands for knowledge-of-exponent, and AGM stands for algebraic group model.

Scheme	Universal SRS	Circuit SRS	Size	Prover comp	Verifier comp
Groth 16 [45]	—	$3n + m\ \mathbb{G}$	$3\ \mathbb{G}$	$4n + m - \ell$ Ex	$3P + \ell$ Ex
Bulletproofs	$\frac{n}{2}\mathbb{G}$	—	$2\log_2(n) + 6\ \mathbb{G}$	$8n$ Ex	$4n$ Ex
This work (helped)	$4d\mathbb{G}$	$12n\ \mathbb{G}$	$7\ \mathbb{G}$, $3\ \mathbb{F}$	$18n$ Ex	$9P$
This work (unhelped)	$4d\mathbb{G}$	$12n\ \mathbb{G}$	$20\ \mathbb{G}$, $16\ \mathbb{F}$	$60n$ Ex	$13P$

Table 2: Comparison of helped and unhelped Sonic against a pairing-based zk-SNARK and against Bulletproofs (which do not require pairing groups) for arithmetic circuit satisfiability with ℓ-element known circuit inputs, m wires, and n gates. Computational costs are measured in terms of number of group exponentiations and pairings. $\mathbb{G}$ means group elements in either source group, $\mathbb{F}$ means field elements, Ex means group exponentiations, and P means pairings.

Hyrax [60] is a zero-knowledge protocol that processes circuits using a sum-check protocol originally introduced from the verifiable computation scheme by Goldwasser et al [43] and improved by Cormode et al. [33]. It is especially well-suited to circuits with a high level of parallelisation, such as showing that a committed value is included in a Merkle tree. Additionally, the protocol is ideal for circuits with small witnesses. It directly uses a parallelised sum-check protocol on the instance wires, and on the witness wires it applies a zero-knowledge variant of the sum-check protocol. Their sum-check protocol uses an adaptation of the inner-product argument from Bulletproofs to check multiplication constraints.

Originally designed for handling SQL queries, Zhang et al. designed a zero-knowledge variant of vSQL [64]. Their scheme also processes circuits using techniques by Cormode et al. [33]. This means that their techniques also have better efficiency for highly parallelised circuits. Like our scheme, they rely on a polynomial commitment scheme. However, rather than design their scheme around Kate et al.'s single variant scheme, they use Papamanthou et al.'s multivariate scheme [57]. The reason this multivariate scheme is useful for vSQL is because they can use multivariate polynomials where each variable has degree 1. For our scheme, there are two variables of degree $O(n)$, so Papamanthou et al.'s scheme would result in a quadratic-sized reference string and quadratic prover computation.

Symmetric primitives such as Reed-Solomon codes have recently been gaining attention for their post-quantum potential, as there are no known quantum attacks on error-correcting codes and protocols that use them do not require expensive and trusted pre-processing phases. Schemes that use these techniques [2, 9, 23] are typically made non-interactive in the random oracle model, as opposed to the quantum random oracle model, and designing efficient zero-knowledge protocols in the quantum random oracle model [21] remains an open problem. The codes are typically cheap to compute for the prover. The downside to this style of proof is that they require very large circuits before the asymptotics can take effect, because the constants are relatively large.

Ligero [2] uses Reed-Solomon codes for security. This work stems from the "MPC-in-the-head" paradigm [30, 42, 49]. The idea is to model the computation as being carried out by a multiparty computation, but then have the prover and verifier simulate multiple parties. A large part of its overhead comes from compiling the addition gates, and the authors observed that when there are many repetitions of the same addition gates in the same layer, it is possible to batch the compilation. Bootle et al. [23] introduce a model that they call the ideal linear commitment (ILC) model in which a prover can commit to vectors by sending them to a channel, and a verifier can query the channel on linear combinations of the committed vectors. They then compile the ILC programs into proofs using a code by Ishai et al. [48] which can be computed in linear time. As a result, they prove the possibility of zk-proofs that have linear prover overhead. STARKs [9] look to simultaneously minimise proof size and verifier computation and they show, with an implementation, that protocols based on interactive oracle proofs [14] can be practical. Indeed their prover, when applied to a circuit with 2^{27} gates, takes roughly 1 minute to run. However, proof sizes are still over 100KB, even for relatively small circuits. Aurora [13] uses similar techniques to STARKs, except that it is designed to run directly over constraint systems like those used in zk-SNARKs. As such, they avoid the concrete overhead that STARKs require for compiling a program into a constraint system. Baum et al. [4] introduced the first lattice based protocol with sublinear communication costs. They achieve this by designing a proof of knowledge for committed values using techniques by Cramer et al. [34]. The proof of knowledge is efficient in the amortised setting. They apply this proof of knowledge to circuits processed using Bulletproof techniques. As a result their verifier time is high.

Bulletproofs [22, 26] are based on the discrete logarithm problem and have no trusted setup. Their proof sizes are logarithmic, which is achieved through the use of an inner product argument. On the downside the verification time is high. Although Bulletproofs lend themselves well to batching, even batched proofs require a computation per proof that depends on the size of the circuit. The prover costs for Bulletproofs are typically high due to the use of expensive cryptographic operations. For very small circuits, such as for range proofs, Bulletproofs have the advantage of having relatively low concrete overhead.

Using knowledge assumptions, it is possible to build zk-SNARKs [15, 18, 35, 45, 47, 54, 58]. These have constant-sized proofs and verifier times that depend solely on the instance. However, they typically rely on using circuit-specific quadratic span programs or quadratic arithmetic programs [40]. As such the common reference strings are not updatable or universal [12]. The prover costs for zk-SNARKs are typically high due to the use of expensive cryptographic operations, although recent work has looked into methods to distribute these costs [63]. For the QAP-based SNARKs one could use ValiantâĂŹs universal circuit construction [55, 59] to achieve universality but this would introduce a $\log(n)$ multiplicative overhead.

Groth et al. [46] introduced the notion of updatability for structured reference strings and built a zk-SNARK from an updatable and universal string. They achieved these results by including a null space argument to show that a quadratic arithmetic circuit is satisfied. However, computing this null space requires expensive Gaussian elimination. Even as a one-off cost, this is often unrealistic. Further, although they can have linear-sized structured reference strings for the prover and verifier, to allow for updatability they require a global string with $O(n^2)$ elements.

3 DEFINITIONS FOR UPDATABLE REFERENCE STRINGS

In this section, we revisit the definitions around updatable SRS schemes due to Groth et al. [46], in terms of defining properties of zero-knowledge proofs in the case in which the adversary may subvert or participate in the generation of the common reference string. Given that our protocol in Section 6 is interactive (but made non-interactive in the random oracle model), we also present new definitions for interactive protocols that take into account these alternative methods of SRS generation.

3.1 Notation

If x is a binary string then $|x|$ denotes its bit length. If S is a finite set then $|S|$ denotes its size and $x \xleftarrow{\$} S$ denotes sampling a member uniformly from S and assigning it to x. We use $\lambda \in \mathbb{N}$ to denote the security parameter and 1^λ to denote its unary representation. We use ε to denote the empty string.

Algorithms are randomized unless explicitly noted otherwise. "PPT" stands for "probabilistic polynomial time" and "DPT" stands for "deterministic polynomial time." We use $y \leftarrow A(x; r)$ to denote running algorithm A on inputs x and random coins r and assigning its output to y. We write $y \xleftarrow{\$} A(x)$ or $y \xleftarrow{r} A(x)$ (when we want to refer to r later on) to denote $y \leftarrow A(x; r)$ for r sampled uniformly at random. For an adversary $\mathcal{A}(1^\lambda)$, we refer to the length of its randomness as $\mathcal{A}.\mathsf{rl}(\lambda)$, and sample $r \xleftarrow{\$} \{0,1\}^{\mathcal{A}.\mathsf{rl}(\lambda)}$.

We use code-based games in security definitions and proofs [8]. A game $\mathsf{Sec}_{\mathcal{A}}(\lambda)$, played with respect to a security notion Sec and adversary $\mathcal{A}$, has a MAIN procedure whose output is the output of the game. The notation $\Pr[\mathsf{Sec}_{\mathcal{A}}(\lambda)]$ is used to denote the probability that this output is 1.

3.2 The Subvertible SRS Model

Intuitively, the subvertible SRS model [7] allows the adversary to fully generate the reference string itself, and the updatable SRS model [46] allows the adversary to partially contribute to its generation by performing some update. Formally, an updatable SRS scheme is defined by two PPT algorithms Setup and Update, and a DPT algorithm VerifySRS. These behave as follows:

- $(\mathsf{srs}, \rho) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$ takes as input the security parameter and returns a SRS and proof of its correctness.
- $(\mathsf{srs}', \rho') \xleftarrow{\$} \mathsf{Update}(1^\lambda, \mathsf{srs}, (\rho_i)_{i=1}^n)$ takes as input the security parameter, a SRS, and a list of update proofs. It outputs an updated SRS and a proof of the correctness of the update.
- $b \leftarrow \mathsf{VerifySRS}(1^\lambda, \mathsf{srs}, (\rho_i)_{i=1}^n)$ takes as input the security parameter, a SRS, and a list of proofs. It outputs a bit indicating acceptance ($b = 1$), or rejection ($b = 0$).

Definition 3.1. An updatable SRS scheme is perfectly correct if

$$\Pr\left[(\mathsf{srs}, \rho) \xleftarrow{\$} \mathsf{Setup}(1^\lambda) : \mathsf{VerifySRS}(1^\lambda, \mathsf{srs}, \rho) = 1\right] = 1,$$

and if for all $(\lambda, \mathsf{srs}, (\rho_i)_{i=1}^n)$ where $\mathsf{VerifySRS}(1^\lambda, \mathsf{srs}, (\rho)_{i=1}^n) = 1$, we have that

$$\Pr\left[\begin{array}{l}(\mathsf{srs}', \rho_{n+1}) \xleftarrow{\$} \mathsf{Update}(1^\lambda, \mathsf{srs}, (\rho_i)_{i=1}^n) : \\ \mathsf{VerifySRS}(1^\lambda, \mathsf{srs}', (\rho)_{i=1}^{n+1}) = 1\end{array}\right] = 1.$$

In terms of the usage of these SRSs in NIZK arguments, a protocol cannot satisfy both subvertible zero-knowledge and subvertible soundness [7]. That is, assuming the adversary knows all the randomness used to generate the SRS, they can either break the zero-knowledge property of the scheme or they can break the soundness property of the scheme. We thus recall here the two strongest properties we can hope to satisfy, which are subvertible zero-knowledge and updatable knowledge soundness. The definitions of these properties are simplified versions of the ones given by Groth et al. [46], with the addition of a random oracle H (which behaves as expected, so we omit its description).

Let R be a polynomial-time decidable relation with triples (srs, ϕ, w). We say w is a witness to the instance ϕ being in the relation defined by srs when $(\mathsf{srs}, \phi, w) \in R$.

Definition 3.2 (Subvertible Zero-Knowledge). A NIZK argument for the relation R is S-*zero-knowledge* if for all probabilistic polynomial time (PPT) algorithms $\mathcal{A}$ the advantage $|2\Pr[\text{S-ZK}_{\mathcal{A}}(1^\lambda) = 1] - 1|$ is negligible in λ, where this game is defined as follows:

$\underline{\text{MAIN S-ZK}_{\mathcal{A}}(\lambda)}$
$b \xleftarrow{\$} \{0,1\}$
$r \xleftarrow{\$} \{0,1\}^{\mathcal{A}.\mathsf{rl}(\lambda)}$
$(\mathsf{st}, \mathsf{srs}, (\rho_i)_{i=1}^n) \leftarrow \mathcal{A}^H(1^\lambda; r)$
if $\mathsf{VerifySRS}(1^\lambda, \mathsf{srs}, (\rho_i)_{i=1}^n) = 0$ return $\xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{H, O_{\mathsf{pf}}}(\mathsf{st})$
return $b' = b$

$\underline{O_{\mathsf{pf}}(\phi, w)}$
if $(\mathsf{srs}, \phi, w) \notin R$ return $\perp$
if $b = 0$ return $\mathsf{SimProve}(\mathsf{srs}, r, \phi)$
else return $\mathsf{Prove}(\mathsf{srs}, \phi, w)$

Definition 3.3 (Updatable Knowledge Soundness). A NIZK argument for the relation R is U-*knowledge-sound* if for all PPT algorithms $\mathcal{A}$ there exists a PPT extractor $\mathcal{X}_{\mathcal{A}}$ such that the probability $|\Pr[\text{U-KSND}_{\mathcal{A},\mathcal{X}_{\mathcal{A}}}(1^\lambda)]|$ is negligible in λ, where this game is defined as follows:

$\underline{\text{MAIN U-KSND}_{\mathcal{A},\mathcal{X}_{\mathcal{A}}}(\lambda)}$
$\mathsf{srs} \leftarrow \perp$
$(\phi, \pi) \xleftarrow{r} \mathcal{A}^{H, \text{U-}O_{\mathsf{s}}}(1^\lambda)$
$w \xleftarrow{\$} \mathcal{X}_{\mathcal{A}}(\mathsf{srs}, r)$
return $\mathsf{Verify}(\mathsf{srs}, \phi, \pi) \land (\mathsf{srs}, \phi, w) \notin R$

$\underline{\text{U-}O_{\mathsf{s}}(\mathsf{intent}, \mathsf{srs}_n, (\rho_i)_{i=1}^n)}$
if $\mathsf{srs} \neq \perp$ return $\perp$
if $\mathsf{intent} = \mathsf{setup}$
  $(\mathsf{srs}', \rho') \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
  $Q \leftarrow Q \cup \{\rho'\}$
  return (srs', ρ')
if $\mathsf{intent} = \mathsf{update}$
  $b \leftarrow \mathsf{VerifySRS}(1^\lambda, \mathsf{srs}_n, (\rho_i)_{i=1}^n)$
  if $b = 0$ return $\perp$
  $(\mathsf{srs}', \rho') \xleftarrow{\$} \mathsf{Update}(1^\lambda, \mathsf{srs}_n, (\rho_i)_{i=1}^n)$
  $Q \leftarrow Q \cup \{\rho'\}$
  return (srs', ρ')
if $\mathsf{intent} = \mathsf{final}$
  $b \leftarrow \mathsf{VerifySRS}(1^\lambda, \mathsf{srs}_n, (\rho_i)_{i=1}^n)$
  if $b = 0$ or $Q \cap \{\rho_i\}_i = \emptyset$ return $\perp$
  $\mathsf{srs} \leftarrow \mathsf{srs}_n$; return srs
else return $\perp$

For zero knowledge, we do not need an interactive variant, as we can argue for the non-interactive definition directly. For soundness, we do not use the standard definition of special soundness because our verifier provides two challenges, but rather the generalized notion of *witness-extended emulation* [53]. In particular, we adapt the definition given by Bootle et al. [22] as follows:

Definition 3.4. Let P be an argument for the relation R. Then it satisfies *updatable witness-extended emulation* if for all DPT P* there exists an expected PT emulator $\mathcal{E}$ such that for all PPT algorithms $\mathcal{A}$:

$$\begin{aligned}&\Pr[(\mathsf{srs}', \rho') \xleftarrow{\$} \mathsf{Setup}(1^\lambda);\\ &\quad (\mathsf{srs}, (\rho_i)_i, \phi, w) \xleftarrow{\$} \mathcal{A}(\mathsf{srs}', \rho');\\ &\quad \mathsf{view} \leftarrow \langle \mathsf{P}^*(\mathsf{srs}, \phi, w), \mathsf{V}(\mathsf{srs}, \phi)\rangle :\\ &\quad \mathsf{VerifySRS}(1^\lambda, \mathsf{srs}, (\rho_i)_i) \land \mathcal{A}(\mathsf{view}) = 1]\\ \approx\ &\Pr[(\mathsf{srs}', (\rho_i')_i) \xleftarrow{\$} \mathsf{Setup}(1^\lambda);\\ &\quad (\mathsf{srs}, (\rho_i)_{i=1}^n, \phi, w) \xleftarrow{\$} \mathcal{A}(\mathsf{srs}', \rho');\\ &\quad (\mathsf{view}, w) \leftarrow \mathcal{E}^{\langle \mathsf{P}^*(\mathsf{srs}, \phi, w), \mathsf{V}(\mathsf{srs}, \phi)\rangle} :\\ &\quad \mathsf{VerifySRS}(1^\lambda, \mathsf{srs}, (\rho_i)_i) \land \mathcal{A}(\mathsf{view}) = 1 \land\\ &\quad \text{if view is accepting then } (\phi, w) \in R],\end{aligned}$$

where the oracle called by $\mathcal{E}^{\langle \mathsf{P}^*(\mathsf{srs}, \phi, w), \mathsf{V}(\mathsf{srs}, \phi)\rangle}$ permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards.

This definition uses a slightly different setup from the one in Definition 3.3: rather than interact arbitrarily with an update oracle to set the SRS, the adversary is instead given an initial one and is then allowed to update that in a one-shot fashion. Following Groth et al. [46, Lemma 6], these two definitions are equivalent for Sonic, so we opt for the simpler one.

4 BUILDING BLOCKS

4.1 Bilinear groups

Let $\mathsf{BilinearGen}(1^\lambda)$ be a bilinear group generator that given the security parameter 1^λ produces bilinear parameters $bp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order p with generators $g \in \mathbb{G}_1, h \in \mathbb{G}_2$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerative bilinear map. That is, $e(g^a, h^b) = e(g, h)^{ab}\ \forall a, b \in \mathbb{Z}_p$ and $e(g, h)$ generates $\mathbb{G}_T$.

We require bilinear groups such that the maximum size of our circuit is bounded by $d^2 \leq (p-1)/32$. In practice we expect that $d^2 \ll (p-1)/32$.

We employ bilinear group generators that produce what Galbraith, Paterson and Smart [38] classify as Type III bilinear groups. For such groups no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ exist. These are currently the most efficient bilinear groups.

4.2 The Algebraic Group Model

Sonic is proven secure in the algebraic group model (AGM) by Fuchsbauer et al [37], who used it to prove (among other things) that Groth's 2016 scheme [45] is secure under a "q-type" variant of the discrete log assumption. Previously the only security proof for this scheme was provided in the generic group model (GGM). Although proofs in the GGM can increase our confidence in the security of a scheme, its scope is limited since it does not capture group-specific algorithms that make use of its representation (such as index calculus approaches).

The AGM lies between the standard model and the GGM, and it is a restricted model of computation that covers group-specific attacks while allowing a meaningful security analysis. Adversaries are assumed to be restricted in the sense that they can only output group elements if they were obtained by applying the group operation to previously received group elements. Unlike the GGM, in the AGM one proves security implications via reductions to assumptions (just as in proofs in the standard model).

It is so far unknown how the AGM relates to knowledge-of-exponent (KOE) assumptions, which are used to build every known SNARK that has been proven secure in the standard model (and indeed it is known that SNARKs cannot be proven secure under more standard falsifiable assumptions [41]). The format of these KOE assumptions is similar to the AGM in the sense that proving the assumption incorrect would require showing that there is an adversary that can compute group elements of a given format but that cannot extract an algebraic representation. To use KOE assumptions, however, it is necessary to compute elements in the second source group. As we would like to avoid introducing proof elements in the second source group (as these are typically more expensive due to current implementations of asymmetric bilinear groups), we instead decided to work with the AGM.

An algorithm $\mathcal{A}_{\mathrm{alg}}$ is called algebraic if whenever it outputs an element Z in $\mathbb{G}$, it also outputs a representation $(z_1, \ldots, z_t) \in \mathbb{Z}_p^t$ such that $Z = \prod_{i=1}^t g_i^{z_i}$ where $\mathcal{L} = \{g_1, \ldots, g_t\}$ is the list of all group elements given to $\mathcal{A}_{\mathrm{alg}}$ in its execution thus far. Unlike the GGM, in the AGM one proves security implications via reductions.

To prove our scheme secure in the algebraic group model we use the q discrete log assumption (q-DLOG)

Assumption 4.1 (q-DLOG assumption). *Suppose that $\mathcal{A}$ is an algebraic adversary. Then*

$$\Pr\left[\begin{array}{l} bp \leftarrow \mathsf{BilinearGen}(1^\lambda);\ x \xleftarrow{\$} \mathbb{Z}_p; \\ x \xleftarrow{\$} \mathcal{A}(bp, \{g^{x^i}, h^{x^i}\}_{i=-q}^{q}) \end{array}\right]$$

is negligible in 1^λ.

4.3 Structured Reference String

In all of the following we require a structured reference string with unknowns x and α of the following form

$$\left\{\ \{g^{x^i}\}_{i=-d}^{d},\ \{g^{\alpha x^i}\}_{i=-d, i\neq 0}^{d},\ \{h^{x^i}, h^{\alpha x^i}\}_{i=-d}^{d},\ e(g, h^\alpha)\ \right\}$$

for some large enough d to support the circuit depth n.

This string is designed so that g^α is omitted from the reference string. Thus we can, when necessary, force the prover to demonstrate that a committed polynomial (in x) has a zero constant term.

4.4 Polynomial Commitment Scheme

We consider a polynomial commitment scheme with three DPT protocols

- $F \leftarrow \mathsf{Commit}(bp, \mathtt{srs}, \mathsf{max}, f(X))$ takes as input the bilinear group, the structured reference string, a maximum degree, and a Laurent polynomial with powers between $-d$ and max. It returns a commitment F.
- $(f(z), W) \leftarrow \mathsf{Open}(bp, \mathtt{srs}, \mathsf{max}, F, z, f(X))$ takes as input the same parameters as the commit algorithm in addition to a commitment F and a point in the field z. It returns an evaluation $f(z)$ and a proof of its correctness.
- $b \leftarrow \mathsf{pcV}(bp, \mathtt{srs}, \mathsf{max}, F, z, v, W)$ takes as input the bilinear group, the SRS, a maximum degree, a commitment, a point in the field, an evaluation and a proof. It outputs a bit indicating acceptance ($b = 1$), or rejection ($b = 0$).

We require that this scheme is *evaluation binding*; i.e., given a commitment F, an adversary cannot open F to two different evaluations v_1 and v_2. We also require that it is *bounded polynomial extractable*; i.e., any algebraic adversary that opens a commitment F knows an opening $f(X)$ with powers $-d \leq i \leq \mathsf{max}$, $i \neq d - \mathsf{max}$.

In Section 6.2 we provide an evaluation binding binding, bounded polynomial extractable polynomial commitment scheme. We prove its security in the algebraic group model in Lemma 6.3.

4.5 Signature of Correct Computation

We consider a signature of correct computation with two DPT protocols

- $(s(z, y), \mathsf{sc}) \leftarrow \mathsf{scP}(bp, \mathtt{srs}, s(X, Y), (z, y))$ takes as input the bilinear group, the SRS, a two-variate polynomial $s(X, Y)$,

and two points in the field (z, y). It returns an evaluation $s(z, y)$ and a proof sc.
- $b \leftarrow \mathsf{scV}(bp, \mathtt{srs}, s(X,Y), (z,y), s, \mathsf{sc})$ takes as input the same parameters as the scP algorithm in addition to an evaluation and a proof. It outputs a bit indicating acceptance ($b = 1$), or rejection ($b = 0$).

We require that this scheme is sound i.e. given $(z, y), s$, an algebraic adversary can only convince an scV verifier if $s = s(z, y)$.

We provide two competing constructions: one in Section 8 and the other in Section 7. The first has linear verifier computation, but can be aggregated by an untrusted helper to achieve constant verifier computation in the batched setting. The second has constant verifier computation but higher concrete overhead. Both constructions have constant size.

5 SYSTEM OF CONSTRAINTS

Sonic uses a form of constraint system proposed by Bootle et al. [22]. We make several modifications so that their approach is practical in our setting.

Our constraint system has three vectors of length n, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ representing the left inputs, right inputs, and outputs of multiplication constraints respectively

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{c}.$$

We also have Q linear constraints of the form

$$\mathbf{a} \cdot \mathbf{u_q} + \mathbf{b} \cdot \mathbf{v_q} + \mathbf{c} \cdot \mathbf{w_q} = k_q$$

where $\mathbf{u_q}, \mathbf{v_q}, \mathbf{w_q} \in \mathbb{F}^n$ are fixed vectors for the q-th linear constraint, with instance value $k_q \in \mathbb{F}_p$. We proceed to compress the n multiplication constraints into an equation in formal indeterminate Y.

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

In order to support our later argument, we (redundantly) encode these constraints into negative exponents of Y

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

We compress the Q linear constraints similarly, scaling by Y^n to preserve linear independence.

$$\sum_{q=1}^{Q} \left(\mathbf{a} \cdot \mathbf{u_q} + \mathbf{b} \cdot \mathbf{v_q} + \mathbf{c} \cdot \mathbf{w_q} - k_q \right) Y^{q+n} = 0.$$

Let us define the polynomials

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

and combine our multiplicative and linear constraints to form the equation

$$\mathbf{a} \cdot \mathbf{u}(Y) + \mathbf{b} \cdot \mathbf{v}(Y) + \mathbf{c} \cdot \mathbf{w}(Y) + \sum_{i=1}^{n} a_i b_i (Y^i + Y^{-i}) - k(Y) = 0. \quad (1)$$

Given a choice of $(\mathbf{a}, \mathbf{b}, \mathbf{c}, k(Y))$, we have that Equation 1 will hold at all points if the constraint system is satisfied. If the constraint system is not satisfied the equation will fail to hold with high probability, given a large enough field.

We apply a technique from Bootle et al. [22] to embed the left hand side of Equation 1 into the constant term of a polynomial $t(X, Y)$ in second formal indeterminate X. We design the polynomial $r(X, Y)$ such that $r(X, Y) = r(XY, 1)$.

$$\begin{aligned} r(X,Y) &= \sum_{i=1}^{n} \left(a_i X^i Y^i + b_i X^{-i} Y^{-i} + c_i X^{-i-n} Y^{-i-n} \right) \\ s(X,Y) &= \sum_{i=1}^{n} \left(u_i(Y) X^{-i} + v_i(Y) X^i + w_i(Y) X^{i+n} \right) \\ r'(X,Y) &= r(X,Y) + s(X,Y) \\ t(X,Y) &= r(X,1) r'(X,Y) - k(Y) \end{aligned}$$

The coefficient of X^0 in $t(X, Y)$ is the left-hand side of Equation 1. Sonic demonstrates that the constant term of $t(X, Y)$ is zero, thus demonstrating that our constraint system is satisfied.

6 THE BASIC SONIC CONSTRUCTION

Sonic is a zero-knowledge argument of knowledge that allows a prover to demonstrate that a constraint system (described in Section 5) is satisfied for a hidden witness $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ and for known instance $\mathbf{k}$. Given a choice of $r(X, Y)$ from Section 5, if for random $y \in \mathbb{F}_p$ we have that the constant term of $t(X, y)$ is zero, the constraint system is satisfied with high probability.

Our Sonic protocol is built directly from a polynomial commitment scheme and a signature of correct computation, as visualised in Figure 1. We discuss here the basic Sonic protocol, assuming these building blocks are in place, and provide a suitable bounded extractable polynomial commitment scheme in Section 6.2 that we prove secure in the AGM. In Sections 7 and 8 we discuss two different methods of constructing the signature of correct computation, one which gives rise to a standalone zk-SNARK and one which achieves better practical results through the use of an untrusted helper.

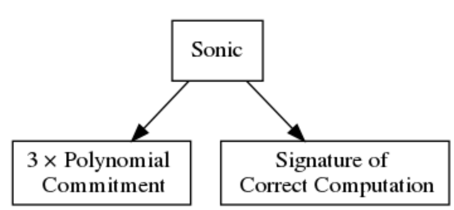

Figure 1: The basic Sonic protocol is built on top of a bounded extractable polynomial commitment scheme and a signature of knowledge.

Our protocol begins by having the prover construct $r(X, Y)$ using their hidden witness. They commit to $r(X, 1)$, setting the maximum degree to n. The verifier sends a random challenge y. The prover commits to $t(X, y)$, and our commitment scheme ensures that this polynomial has no constant term. The verifier sends a second challenge z. The prover opens their committed polynomials to $r(z, 1)$, $r(z, y)$ and $t(z, y)$. The verifier can calculate $r'(z, y)$ for itself from these values and thus can check that $r(z, y)r'(z, y) - k(y) = t(z, y)$. If this holds then the verifier learns that the evaluated polynomials were computed by a prover that knows a valid witness. A more formal description of this protocol is given in Figure 2.

The verifier's check that the quadratic polynomial equation is satisfied is performed in the field. This means we avoid having proof elements on both sides of the pairing, which is useful for efficiency, without contradicting Groth's result about NILPs requiring a quadratic constraint [45]. As a result, when batching we avoid having to check one pairing equation per proof (pairing operations expensive) and can instead check one field equation per proof.

The Fiat-Shamir transformation takes an interactive argument and replaces the verifier challenges with the output of a hash function. The idea is that the hash function will produce random-looking outputs and therefore be a suitable replacement for the verifier. We describe Sonic in the interactive setting where all verifier challenges are random field elements. In practice we assume that the Fiat-Shamir heuristic would be applied in order to obtain a non-interactive zero-knowledge argument in the random oracle model.

Theorem 6.1. *Sonic satisfies perfect subversion zero-knowledge.*

Proof. To prove subversion zero-knowledge, we need to both show the existence of an extractor $X_{\mathcal{A}}$ that can compute a trapdoor from the updated proofs, and describe a SimProve algorithm that produces indistinguishable proofs when provided with the extracted trapdoor. The existence of an extractor that can compute a trapdoor from the updated proofs follows from Lemma 4 of [46].

The simulator is given the trapdoor g^{α} and chooses random vectors $\boldsymbol{a}, \boldsymbol{b}$ from $\mathbb{F}_p$ of length n and sets $\boldsymbol{c} = \boldsymbol{a} \cdot \boldsymbol{b}$. It computes $r(X, Y)$, $r'(X, Y)$, $t(X, Y)$ as in Section 5 where (unlike for the prover) $t(X, Y)$ can have a non-zero coefficient in X^0. The simulator then behaves exactly as the prover in Figure 2 with its random polynomials.

Both the prover and the simulator evaluate $g^{r(x,1)}$, $r(z, 1)$, and $r(zy, 1)$. This reveals 3 evaluations (some of these are in the exponent). The prover has four blinders for $r(X)$ with respect to the powers $-2n-1, -2n-2, -2n-3, -2n-4$. Thus for a verifier that obtains less than three evaluations, the prover's polynomial is indistinguishable from the simulator's random polynomial. All other components in the proofs are either uniquely determined given the previous components for both prover and simulator, or are calculated independently from the witness (and are chosen in the same method by both prover and simulator). Consequently, subversion zero-knowledge follows from the extraction of the trapdoor, which we show below. □

Theorem 6.2. *Sonic has witness extended emulation, when instantiated using an evaluation binding and bounded polynomial extractable polynomial commitment scheme and a sound signature of correct computation.*

Proof. Soundness of the signature of correct computation gives us that that $s = s(z, y)$.

Bounded polynomial extractability tells us that R contains the polynomial

$$r(X, 1) = \sum_{i=-d, i\neq -d+n}^{n} r_i X^i$$

and that T contains the polynomial

$$\tau(X) = \sum_{i=-d, i\neq 0}^{d} \tau_i X^i.$$

Observe that in our polynomial constraint system $3n < d$ (otherwise we cannot commit to $t(X, Y)$), thus $r(X, Y)$ has no $-d + n$ term.

We show that the element T can only be computed if the circuit is satisfied by the polynomial coefficients extracted from R. Evaluation binding tells us that $a = r(z, 1)$, $b = r(zy, 1) = r(z, y)$ and the verifier checks that $t = a(b+s) - k(y) = \tau(z)$. Suppose this holds for $n+Q+1$ different challenges $y \in \mathbb{F}_p$. Then we have equality of polynomials in Section 5 since a non-zero polynomial of degree $n + Q + 1$ cannot have $n + Q$ roots; i.e.,

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

has no constant term. This implies that $r(X, y)$ defines a valid witness. □

6.1 Efficiency

Our prover uses two polynomial commitments which it opens at three points. It also uses one signature of correct computation. Two of these openings can be batched using techniques we describe in Appendix C. The idea of behind the batching is that given two polynomial commitments F_1 and F_2, if a verifier chooses random values r_1 and r_2, then an adversary can only open $F_1^{r_1} F_2^{r_2}$ if it can also (with high probability) open F_1 and F_2 separately.

6.2 Polynomial Commitment Scheme

Sonic uses a polynomial commitment scheme which is an adaptation of a scheme by Kate, Zaverucha, and Goldberg [50]. This scheme has constant-sized proofs for any size polynomial and verification consists of checking a single pairing. We require that the scheme is evaluation binding; i.e., given a commitment F, an adversary cannot open F to two different evaluations v_1 and v_2. Our proof of evaluation binding is directly taken from Kate et al.'s reduction to q-SDH. However, we also require that the scheme is bounded polynomial extractable; i.e., any algebraic adversary that opens a commitment F knows an opening $f(X)$ with powers $-d \leq i \leq \max, i \neq 0$. Kate et al. only prove that their scheme is "strongly correct"; i.e., if an adversary knows an opening $f(X)$ with polynomial degree to a commitment then $f(X)$ has degree bounded by d. In this sense Kate et al. are implicitly relying on a knowledge assumption, because there is no guarantee that an adversary that can open a commitment knows a polynomial inside the commitment. We prove our adapted polynomial commitment scheme secure in the algebraic group model and this proof may be of independent interest.

Common Input: info = $bp,\ \mathsf{srs},\ s(X,Y),\ k(Y),\ \phi,\ e(g,h^{\alpha})$
Prover's input: $\mathbf{a},\mathbf{b},\mathbf{c}$

$\underline{\mathsf{zkP}_1(\mathsf{info},\mathbf{a},\mathbf{b},\mathbf{c})\mapsto R}$:
$c_{n+1},c_{n+2},c_{n+3},c_{n+4}\xleftarrow{\$}\mathbb{F}_p$
$r(X,Y)\leftarrow r(X,Y)+\sum_{i=1}^{4}c_{n+i}X^{-2n-i}$
$R\leftarrow\mathsf{Commit}(bp,\mathsf{srs},n,r(X,1))$
return R

$\underline{\mathsf{zkV}\mapsto\mathsf{zkP}}$: Send $y\xleftarrow{\$}\mathbb{F}_p$ to prover

$\underline{\mathsf{zkP}_2(y)\mapsto T}$:
$T\leftarrow\mathsf{Commit}(bp,\mathsf{srs},d,t(X,y))$
return T

$\underline{\mathsf{zkV}\mapsto\mathsf{zkP}}$: Send $z\xleftarrow{\$}\mathbb{F}_p$ to prover

$\underline{\mathsf{zkP}_3(z)\mapsto(a,W_a,b,W_b,W_t,s,\mathsf{sc})}$:
$(a=r(z,1),W_a)\leftarrow\mathsf{Open}(R,z,r(X,1))$
$(b=r(z,y),W_b)\leftarrow\mathsf{Open}(R,yz,r(X,1))$
$(t=t(z,y),W_t)\leftarrow\mathsf{Open}(T,z,t(X,y)))$
$(s=s(z,y),\mathsf{sc})\leftarrow\mathsf{scP}(\mathsf{info},s(X,Y),(z,y))$
return $(a,W_a,b,W_b,W_t,s,\mathsf{sc})$

$\underline{\mathsf{zkV}(\mathsf{info},(R,T,a,W_a,b,W_b,W_t,s,\mathsf{sc}),y,z)\mapsto 0/1}$:
$t\leftarrow a(b+s)-k(y)$
check $\mathsf{scV}(\mathsf{info},s(X,Y),(z,y),(s,\mathsf{sc}))$
check $\mathsf{pcV}(bp,\mathsf{srs},n,R,z,(a,W_a))$
check $\mathsf{pcV}(bp,\mathsf{srs},n,R,yz,(b,W_b))$
check $\mathsf{pcV}(bp,\mathsf{srs},d,T,z,(t,W_t))$
return 1 if all checks pass, else return 0.

Figure 2: The interactive Sonic protocol to check that prover knows a valid assignment of the wires in the circuit.

Our proof uses the fact that $f(X)-f(z)$ is divisible by $(X-z)$, even for Laurent polynomials. To see this observe that

$$\begin{aligned}f(X)-f(z)&=\sum_{-d}^{d}a_iX^i-a_iz^i\\&=\sum_{i=1}^{d}a_i(X-z)(X^{i-1}+zX^{i-2}+\ldots z^{i-1})+0a_0\\&\quad+\sum_{i=-1}^{-d}a_i(X-z)(-z^{-1}X^{-i}-z^{-2}X^{-i+1}-\ldots-z^{-i}X^{-1})\end{aligned}$$

Theorem 6.3. *The polynomial commitment scheme in Figure 3 is evaluation binding and bounded polynomial extractable. These properties also hold for an algebraic adversary that can update the SRS.*

Proof. We closely follow the structure used by Fuchsbauer et al. [37, Theorem 7.2]. We calculate a polynomial that can be extracted by an adversary with access to the transcript of a succeeding adversary. This polynomial is defined by the verification equations. We then show that if the polynomial does not equal 0 with overwhelming probability then our extracting adversary can break q-DLOG.

We consider an algebraic adversary $\mathcal{B}_{\mathsf{alg}}$ against a q-DLOG instance (which is additionally given a zero-knowledge proof of knowledge of x). We assume that there is an algebraic adversary $\mathcal{A}_{\mathsf{alg}}$ that generates a commitment which it can open. We show that either $\mathcal{A}_{\mathsf{alg}}$ can extract an opening $f(X)$ of the commitment within the correct bounds, or $\mathcal{B}_{\mathsf{alg}}$ returns the q-DLOG challenge

Common Input: info = $bp,\mathsf{srs},\max$
Prover's input: $f(X)$

$\underline{\mathsf{Commit}(\mathsf{info},f(X))\mapsto F}$:
$F\leftarrow g^{\alpha x^{d-\max}f(x)}$
return F

$\underline{\mathsf{Open}(\mathsf{info},F,z,f(X))\mapsto(f(z),W)}$:
$w(X)\leftarrow\frac{f(X)-f(z)}{X-z}$
$W\leftarrow g^{w(x)}$
return $(f(z),W)$

$\underline{\mathsf{pcV}(\mathsf{info},F,z,(v,W))\mapsto 0/1}$:
check $e(W,h^{\alpha x})e(g^{v}W^{-z},h^{\alpha})=e(F,h^{x^{-d+\max}})$
return 1 if all check passes, else return 0.

Figure 3: Polynomial commitment scheme inspired by Kate et al [50].

i.e.

$$\mathsf{Adv}^{\text{bounded-extractable}}_{bp,\mathcal{A}_{\text{alg}}} \leq 2\mathsf{Adv}^{q\text{-DLOG}}_{bp,\mathcal{B}_{\text{alg}}}$$

We then consider another algebraic adversary C_{alg} against a q-SDH instance and show that if $\mathcal{A}_{\text{alg}}$ can open their commitment at z to $v \neq f(z)$ then C_{alg} returns the q-SDH challenge. Following the generic bound for Boneh and Boyen's SDH assumption [20] we may assume that $\mathsf{Adv}^{q\text{-DLOG}}_{bp,C_{\text{alg}}} \geq \mathsf{Adv}^{q\text{-SDH}}_{bp,C_{\text{alg}}}$ and so

$$\mathsf{Adv}^{\text{evaluation binding}}_{bp,\mathcal{A}_{\text{alg}}} \leq \mathsf{Adv}^{q\text{-DLOG}}_{bp,C_{\text{alg}}}.$$

Overall this gives us that

$$\mathsf{Adv}^{\text{polycommit}}_{bp,\mathcal{A}_{\text{alg}}} \leq 2\mathsf{Adv}^{q\text{-DLOG}}_{bp,\mathcal{B}_{\text{alg}}} + \mathsf{Adv}^{q\text{-DLOG}}_{bp,C_{\text{alg}}}.$$

An adversary $\mathcal{B}_{\text{alg}}(g^1, g^x, \ldots, g^{x^q}, \mathsf{POK}(x))$ simulates an adversary against the polynomial commitment scheme $\mathsf{U\text{-}SND}_{\mathcal{A}_{\text{alg}}}$. It behaves as follows.

(1) Choose a random value (u_1, u_2) and generates an SRS with randomness $(u_1 x, u_2 x)$
(2) When $\mathcal{A}_{\text{alg}}$ updates the string using randomness (x_i, α_i) reset $(u_1, u_2) = (x_i u_1, \alpha_i u_2)$.
(3) If $\mathcal{A}_{\text{alg}}$ queries its update oracle, choose a fresh (u_1, u_2), and uses both own and $\mathcal{A}'_{\text{alg}}s$ randomness to simulate an update proof.
(4) Choose z and run $(F, v, W) \xleftarrow{r} \mathcal{A}_{\text{alg}}(bp, \mathsf{srs}, \mathsf{max}, z)$.
(5) The randomness r determines multivariate polynomials

$$\begin{aligned} f(X, X_\alpha) &= f_x(X) + X_\alpha f_\alpha(X), \\ w(X, X_\alpha) &= w_x(X) + X_\alpha w_\alpha(X), \end{aligned}$$

such that

$$F = g^{f(xu_1, xu_2)} \text{ and } W = g^{w(xu_1, xu_2)}.$$

From these polynomials, computes the polynomial

$$Q_1(X, X_\alpha) = X_\alpha(X - z)w(X, X_\alpha) + vX_\alpha - X^{-d+\mathsf{max}} f(X, X_\alpha).$$

Abort if $Q_1(X, X_\alpha) = 0$.
(6) Define the univariate polynomial $Q_1'(X) = Q_1(u_1 X, u_2 X)$. Abort if $Q_1'(X) = 0$.
(7) Factor $Q_1'(X)$ to obtain its roots (of which there are at most $4d$) and check them against the q-DLOG instance to determine whether x is among them. If so, return x. Else return $\perp$.

Now let us analyse the probability that

$$f(X, X_\alpha) \neq X_\alpha X^{d-\mathsf{max}} \sum_{i=-d, i\neq 0}^{\mathsf{max}} a_i X^i.$$

but $\mathcal{B}_{\text{alg}}$ does not return the target x. First consider the scenario where $Q_1(X, X_\alpha) = 0$. Then

$$X_\alpha(X - z)w(X, X_\alpha) + vX_\alpha - (X^{-d+\mathsf{max}})f(X, X_\alpha) = 0$$

which implies that

$$(X - z)w_x(X) + v - (X^{-d+\mathsf{max}})f_\alpha(X) = 0.$$

and $(X - z)$ divides $(X^{-d+\mathsf{max}})f_\alpha(X) - v$ and $f_\alpha(X)$ has non-zero terms between $-\mathsf{max} < i < d$. Thus $f_\alpha(X)$ has no terms less than $-\mathsf{max}$. Moreover $f_\alpha(X)$ has no zero term because this is not given in the reference string. Thus in Step 5, $\mathcal{B}$ only aborts if $f(X, X_\alpha)$ is as assumed.

In Step 6, $\mathcal{B}_{\text{alg}}$ aborts only if $Q_1(u_1 X, u_2 X) = 0$. By the Schwartz-Zippel lemma, the probability of this occurring is bounded by $\frac{(4d)^2}{p-1}$ where d is the total degree of Q (recall we have negative powers). Following the generic bound for Boneh and Boyen's SDH assumption [20] we may assume that $\mathsf{Adv}^{q\text{-DLOG}}_{bp,\mathcal{B}_{\text{alg}}} \geq \frac{q^2}{p-1}$.

In Step 7, observe that $Q_1(u_1 x, u_2 x)$ exactly defines the verifiers equation, thus if $\mathcal{A}_{\text{alg}}$ succeeds then $Q_1(u_1 x, u_2 x) = 0$. Thus $Q_1'(x) = 0$ and x is a root of $Q_1'(X)$.

Thus when $\mathcal{A}_{alg}$ succeeds at convincing the polycommit verifier despite its algebraic extractor not outputting a polynomial of the expected form, then $\mathcal{B}_{alg}$ returns x unless $Q_1(u_1 X, u_2 X) = 0$. Thus

$$\mathsf{Adv}^{\text{bounded-extractable}}_{bp,\mathcal{A}_{\text{alg}}} \leq 2\mathsf{Adv}^{q\text{-DLOG}}_{bp,\mathcal{B}_{\text{alg}}}.$$

An adversary $C_{\text{alg}}(g^1, g^x, \ldots, g^{x^q}, \mathsf{POK}(x))$ simulates an adversary against the polynomial commitment scheme $\mathsf{U\text{-}SND}_{\mathcal{A}_{\text{alg}}}$. It behaves as follows.

(1) Run $\mathcal{B}_{\text{alg}}(g^1, g^x, \ldots, g^{x^q}, \mathsf{POK}(x))$. Extract

$$f(X) = X_\alpha X^{d-\mathsf{max}} \sum_{i=-d, i\neq 0}^{\mathsf{max}} a_i X^i,$$

$z, (F, v, W)$.
(2) Run $v', W' \leftarrow \mathsf{Open}(F, bp, \mathsf{srs}, \mathsf{max}, F, z, f(X))$
(3) If $\mathcal{A}_{\text{alg}}$ queries its update oracle, choose a fresh (u_1, u_2), and uses both own and $\mathcal{A}'_{\text{alg}}s$ randomness to simulate an update proof.
(4) Choose z and run $(F, v, W) \xleftarrow{r} \mathcal{A}_{\text{alg}}(bp, \mathsf{srs}, \mathsf{max}, z)$.
(5) If $v \neq v'$ return $(z, (WW'^{-1})^{\frac{1}{v'-v}})$. Else return $\perp$.

Now let us analyse the probability that

$$v \neq z^{-d+\mathsf{max}} f_x(z)$$

but $\mathcal{B}_{\text{alg}}$ does not return the target $(c, g^{\frac{1}{x-c}})$. If $v \neq z^{-d+\mathsf{max}} f_x(z)$ then $v' \neq v$ and C_{alg} does not abort. Then

$$e(W, h^\alpha) e(W^{-z} g^v, h^\alpha) = e(W, h^\alpha) e(W'^{-z} g^{v'}, h^\alpha)$$

and rearrangement yields

$$e(WW'^{-1}, h^{\alpha(x-z)}) = e(g^{v'-v}, h^\alpha).$$

Thus C_{alg} returns $(z, g^{\frac{1}{x-z}})$ and

$$\mathsf{Adv}^{\text{evaluation binding}}_{bp,\mathcal{A}_{\text{alg}}} \leq \mathsf{Adv}^{q\text{-DLOG}}_{bp,C_{\text{alg}}}$$

as required. □

7 SUCCINCT SIGNATURES OF CORRECT COMPUTATION

We consider signatures of correct computation (scP, scV) [57]. Recall from Section 6 that Sonic uses a signature of correct computation to ensure that an element s is equal to $s(z, y)$ for a known polynomial

$$s(X, Y) = \sum_{i,j=-d}^{d} s_{i,j} X^i Y^j.$$

We require the soundness notion that no algebraic adversary can convince an scV verifier unless $s = s(z, y)$. We provide two competing realisations of signatures of correct computation. The first one is described in this section and it is calculated by a prover, and has succinct size and verifier computation. Alternatively, in some settings one can use untrusted helpers to improve practical efficiency, which we describe in the next section.

We use the structure of $s(X, Y)$ in order to prove its correct calculation using a *permutation argument* and a *grand-product argument.* We take inspiration from our main construction and from the permutation and grand-product arguments described by Bayer and Groth [5] and by Bootle et al [23]. We restrict ourselves to constraint systems for which $s(X, Y)$ can be expressed as the sum of M polynomials, where the j-th such polynomial is of the form

$$\Psi_j(X, Y) = \sum_{i=1}^{n} \psi_{j,\sigma_{j,i}} X^i Y^{\sigma_{j,i}}$$

for (fixed) polynomial permutation $\boldsymbol{\sigma}_j$ and coefficients $\psi_{j,i} \in \mathbb{F}$. By introducing additional multiplication constraints to replace any linear constraints that do not fit this format, we can coerce any constraint system in Section 5 into the correct form. This increases the number of multiplication constraints by approximately a factor of 3 when $M = 3$.

Our signature of correct computation uses a polynomial permutation argument, which itself uses a grand-product argument. The permutation argument allows us to verify that each a polynomial commitment contains $\Psi_j(X, y)$, and this can then be opened at z to verify that $\Psi_j(z, y)$ has been calculated correctly. After using batching techniques described in Appendix C, we get that proof sizes are approximately 1 kB.

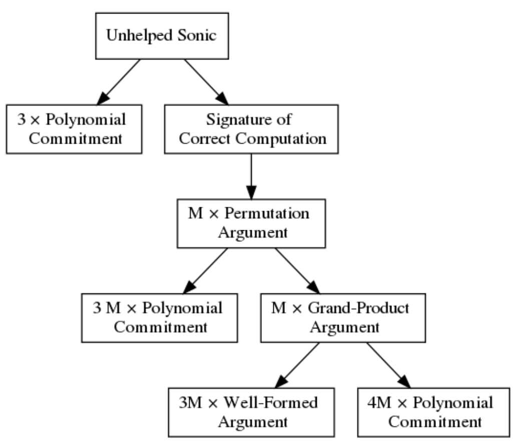

Figure 4: Sonic is built using a bounded extractable polynomial commitment scheme and a signature of correct computation. Here we describe how the prover can construct the signature of correct computation using permutation arguments, grand-product arguments, and the polynomial commitment scheme described in Section 6.2.

7.1 Polynomial Permutation Argument

We consider a polynomial permutation argument with two DPT protocols

- $(\psi, \mathsf{perm}) \leftarrow \mathsf{permP}(bp, \mathsf{srs}, y, z, \Psi(X, Y))$ takes as input the bilinear group, the structured reference string, two points in the field, and a polynomial $\Psi(X, Y) = \sum_{i=1}^{n} \psi_{\sigma_i} X^i Y^{\sigma_i}$. It outputs $\psi = \Psi(z, y)$ and a proof perm.
- $0/1 \leftarrow \mathsf{permV}(bp, \mathsf{srs}, y, z, \Psi(X, Y), (\psi, \mathsf{perm}))$ takes as input the same parameters as the prover, an evaluation, and a proof. It outputs a bit indicating acceptance ($b = 1$), or rejection ($b = 0$).

We require that this scheme is sound; i.e., an algebraic adversary can convince a verifier only if $\psi = \Psi(z, y)$. Our polynomial permutation argument is given in Appendix A.

7.2 Grand-Product Argument

We consider a grand-product argument with two DPT protocols

- $\mathsf{gprod} \leftarrow \mathsf{gprodP}(bp, \mathsf{srs}, A, B, a(X), b(X))$ takes as input the bilinear group, the SRS, two polynomial commitments, and two openings such that $\prod_i a_i = \prod_i b_i$.
- $0/1 \leftarrow \mathsf{permV}(bp, \mathsf{srs}, A, B, \mathsf{gprod})$ takes as input the bilinear group, the SRS, two polynomial commitments, and a proof. It outputs a bit indicating acceptance ($b = 1$), or rejection ($b = 0$).

We require that this scheme is knowledge-sound; i.e., an algebraic adversary can convince a verifier only if it knows openings to A and B whose coefficients have the same grand-product. Our grand-product argument is given in Appendix B.

Common Input: info = $bp, \mathsf{srs}, y, z,$
$s(X, Y) = \sum_{j=1}^{M} \Psi_j(X, Y)$

$\mathsf{scP}(\mathsf{info}) \mapsto (s, \{\psi_j, \mathsf{perm}_j\}_{j=1}^{M})$:

for $1 \le j \le M$:
  $(\psi_j, \mathsf{perm}_j) \leftarrow \mathsf{permP}(bp, \mathsf{srs}, y, z, \Psi_j(X, Y))$
$s \leftarrow \sum_{j=1}^{M} \psi_j$
return $(s, \{\psi_j, \mathsf{perm}_j\}_{j=1}^{M})$

$\mathsf{scV}(\mathsf{info}, (s, \{\psi_j, \mathsf{perm}_j\}_{j=1}^{M})) \mapsto 0/1$:

check $s = \sum_{j=1}^{M} \psi_j$
for $1 \le j \le M$:
  check $\mathsf{permV}(bp, \mathsf{srs}, y, z, \Psi_j(X, Y), (\psi_j, \mathsf{perm}_j))$
return 1 if all checks pass, else return 0

Figure 5: A signature of correct computation using a permutation argument.

Theorem 7.1. *The signature of computation scheme in Figure 5 is sound when instantiated using a sound permutation argument. If the permutation argument is sound against an algebraic adversary that can update the SRS, then so is the signature of computation scheme.*

Proof. The soundness of the permutation argument gives us that no algebraic adversary can convince the verifier of ψ_j unless

	Helper	Verifier	Proof size
Helped	$O(mn\log(n))$	$O(m)+O(n)$	$3m+1\ \mathbb{G}_1$, $m\ \mathbb{F}$
Unhelped	-	$O(m)$	$16m\ \mathbb{G}_1$, $14m\ \mathbb{F}$

Table 3: Computational efficiency and proof size for the SC with respect to the helped verifier. Here n is the number of multiplication gates and m is the number of proofs for the same constraints system. Observe that although the unhelped version has better asymptotic efficiency, in practice the helped verifier is more efficient.

$\psi_j = \sum_{i=1}^{n} \psi_{j,i} z^i y^{\sigma_{j,i}}$. Thus the verifier is convinced if and only if $s(z,y) = \sum_j \psi_j$. □

8 SIGNATURES OF CORRECT COMPUTATION WITH EFFICIENT HELPED VERIFICATION

Recall that Sonic uses a signature of correct computation to ensure that an element s is equal to $s(z,y)$ for a known polynomial

$$s(X,Y) = \sum_{i,j=-d}^{d} s_{i,j} X^i Y^j.$$

We require the soundness notion that no algebraic adversary can convince an scV verifier unless $s = s(z,y)$. In the previous section we described a construction that is calculated directly by a prover, and has succinct size and verifier computation. Alternatively, in some settings one can use untrusted helpers to improve practical efficiency, which we describe in this section. In the helper setting, proof sizes are improved by a factor of three.

In the amortised setting, where one is proving the same thing many times, we can use what we refer to as "helpers" in order to aggregate many signatures of knowledge at the same time. The proofs provided by the helper are succinct and the helper can be run by anyone (i.e., they do not need any secret information from the prover). Verification requires a one-off linear-sized polynomial evaluation in the field and an addition two pairing equations per proof. Compared to the unhelped costs (which require an additional 4 pairings per proof) this is more efficient assuming there is a sufficiently large number of proofs in the batch. As discussed in the introduction, the natural candidate for this role in the setting of blockchains is a miner, as they are already investing computational energy into the system. An efficiency overview is given in Table 3.

The algorithm for our helped signature of correct computation is given in Figure 7. The helper is denoted by hscP and the verifier is denoted by hscV. Roughly the idea is as follows. The helper commits to $s(X, y_j)$ for each element y_j. The verifier provides a random challenge u. The helper commits to $s(u,X)$, and then opens both its commitment to $s(X,y_j)$ at u and its commitment $s(u,X)$ at y_j and checks the two are equivalent. The verifier provides a random challenge z. The helper opens $s(u,X)$ at z. The verifier computes $s(u,z)$ for itself and checks that the helpers opening is correct.

Theorem 8.1. *The aggregated signature of computation scheme in Figure 7 is sound when instantiated using an evaluation binding and bounded polynomial extractable polynomial commitment scheme. If the polynomial commitment is secure against an algebraic adversary*

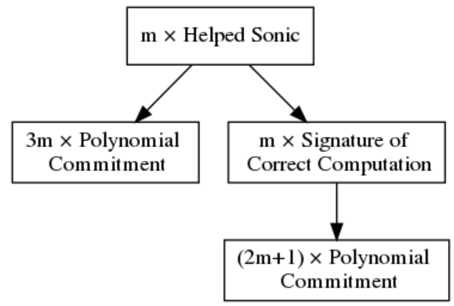

Figure 6: Sonic can be constructed using a signature of correct computation that is calculated by a helper as opposed to directly by the prover. The helper algorithm is run on a batch of proofs, and provides the setting in which Sonic obtains the best practical efficiency.

Common Input: info $=, bp, \mathsf{srs}, \{y_j\}_{j=1}^m, s(X,Y)$

$\underline{\mathsf{hscP}_1(\mathsf{info}) \mapsto (\{S_j\}_{j=1}^m)}$:
for $1 \le j \le m$:
  $S_j \leftarrow \mathsf{Commit}(bp, \mathsf{srs}, d, s(X, y_j))$
return $\{S_j\}_{j=1}^m$

$\underline{\mathsf{hscV} \mapsto \mathsf{hscP}}$: Send $u \xleftarrow{\$} \mathbb{Z}_p$ to prover

$\underline{\mathsf{hscP}_2(u) \mapsto \{s_j, W_j Q_j\}_{j=1}^m}$:
$C \leftarrow \mathsf{Commit}(bp, \mathsf{srs}, d, s(u,x)) g^{s(u,x)}$
for $1 \le j \le m$:
  $(s_j, W_j) \leftarrow \mathsf{Open}(S_j, u, s(X, y_j))$
  $(s_j, Q_j) \leftarrow \mathsf{Open}(C, y_j, s(u, X))$
return $\{s_j, W_j, Q_j\}_{j=1}^m$

$\underline{\mathsf{hscV} \mapsto \mathsf{hscP}}$: Send $z \xleftarrow{\$} \mathbb{Z}_p$ to prover

$\underline{\mathsf{hscP}_2(z) \mapsto Q_z}$:
$(s(u,z), Q_z) \leftarrow \mathsf{Open}(C, z, s(u,X))$
return Q_z

$\underline{\mathsf{hscV}(\mathsf{info}, \{S_j, s_j, W_j, Q_j\}_{j=1}^m, Q_z, u, z) \mapsto 0/1}$:
$s_z \leftarrow s(u,z)$
for $1 \le j \le m$:
  check $\mathsf{pcV}(bp, \mathsf{srs}, S_j, d, u, (s_j, W_j))$
  check $\mathsf{pcV}(bp, \mathsf{srs}, C, d, y_j, (s_j, Q_j))$

check $\mathsf{pcV}(bp, \mathsf{srs}, C, d, z, (s_z, Q_z))$
return 1 if all checks pass, else return 0

Figure 7: The helper protocol for computing aggregated signatures of correct computation.

that can update the SRS, then so is the signature of computation scheme.

Proof. Bounded polynomial extraction gives us that there exists algebraic extractors that output degree-d Laurent polynomials $s'_j(X)$ and $c'(X)$ such that $S_j = g^{\alpha s'_j(x)}$ and $C = g^{\alpha c'(x)}$. First observe that the probability that $c'(z) = s(u, z)$ at a randomly chosen z but that $c'(X) \neq s(u, X)$ is negligible in a sufficiency large field. Second observe that given $c'(X) = s(u, X)$, a PPT algebraic adversary can only open C at a (not randomly chosen) value y_j to $s(u, y_j)$. Finally observe that the probability that $s'_j(X) = s(u, y_j)$ at a randomly chosen u but that $s'_j(X) \neq s(X, y_j)$ is negligible in a sufficiency large field. Thus soundness follows from the evaluation binding property of the commitment. □

9 IMPLEMENTATION

In order to compare the concrete performance of our construction to other protocols we provide an open-source implementation in Rust [1] of Sonic implemented with helpers. The numbers in Table 4 were obtained on CPU i7 2600K with 32 GB of RAM, running at 3.4 GHz.

In terms of our parameters, we make use of the BLS12-381 elliptic curve construction, which is designed so that its group order is a prime p such that $\mathbb{F}_p$ is equipped with large 2^n root of unity for performing fast polynomial multiplications with radix-2 fast-Fourier transforms. BLS12-381 targets the 128-bit security level. Kim and Babalescu [51] describe an optimization to the Number Field Sieve algorithm, analyzed further by Babalescu and Duquesne [3], which may reduce security to 117 bits, but the attack requires a (currently unknown) efficient algorithm for scanning a large space of polynomials.

Proof verification is dominated by a set of pairing equation checks and an evaluation of $s(X, Y)$ in the scalar field. Most of the pairings within (and amongst many) proof verifications involve fixed elements in $\mathbb{G}_2$, so the verifier can combine all of them into a single equation with a probabilistic check. In the context of batch verification each individual proof thus requires arithmetic only in $\mathbb{G}_1$. Only a small, fixed number of pairing operations are performed at the end.

As mentioned in Section 8, the evaluation of $s(X, Y)$ can be done once for a batch of proofs given some post-processing by an untrusted "helper". We consider the performance of batch verification with this post-processing.

In each individual proof we must compute $k(y)$ depending on our instance. We keep this polynomial sparse by only having coefficients in our instance variables, and keeping all other coefficients zero. If constants are needed in the circuit, they are expressed with coefficients of an instance variable that is fixed to one.

We provide an adaptor which translates circuits written in the form of quadratic "rank-1 constraint systems" (R1CS) , a widely deployed NP language currently undergoing standardisation, into the system of constraints natural to our proving system. This adds some constant amount of overhead during proving and verifying steps, but eases implementation and comparison with existing constructions.

The numbers obtained are only relevant to batched proofs, so we wrote an idealized verifier of the Groth 2016 scheme [45], where a batch of proofs are verified together. In this idealized version we assume the $\mathbb{G}_2$ elements do not need to be deserialised and that there is only one public input. We found the marginal cost of verification was around 0.6ms, compared to Sonic's 0.7ms. We thus claim that Sonic has verification time which is competitive with the state-of-the-art for zk-SNARKs, but unlike prior zk-SNARKs has a universal and updatable SRS.

In Table 4 we mimicked Bulletproofs [26, Table 3] in measuring the results of our Sonic implementation. Our implementation is not constant time, however, which may affect this comparison (or indeed the comparison of prover performance to any implementation with constant-time algorithms). We measured the efficiency of the prover, the verifier, and the helped verifier in proving knowledge of x such that $H(x) = y$. Proof sizes are always 256 bytes and verifier computation is always around 0.7ms. In Bulletproofs, in contrast, the proof size for the unpadded 512-bit SHA256 preimage is 1376 bytes and verification time is 41.52 ms, although as we mention this comparison is not exact give in particular that their system was throttled to 2 GHz and that there are optimised implementations for fixed circuits.[2] The runtime of our prover goes up in a roughly linear fashion, as expected. The cost of the helped verifier, in contrast, remains the same for all circuit sizes.

10 CONCLUSIONS

Zero-knowledge protocols have gained significant traction in recent years in the application domain of cryptocurrencies, which has led to the development of new protocols with significant performance gains. At the same time, the requirements of this application have given rise to protocols with new features, such as an untrusted setup and a reference string that allows one to prove more than a single relation. In this paper, we present Sonic, which captures a valuable set of tradeoffs between these key functional requirements of untrusted setup and universality. At the same time, as we demonstrate via a prototype implementation, Sonic has competitive proof sizes and verification time with the state-of-the-art.

ACKNOWLEDGMENTS

We thank Daira Hopwood and Ariel Gabizon for helpful discussions. Mary Maller and Sarah Meiklejohn are supported by EPSRC Grant EP/N028104/1.

REFERENCES

[1] Sonic reference implementation. https://github.com/zknuckles/sonic.

[2] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of ACM CCS*, 2017.

[3] R. Barbulescu and S. Duquesne. Updating key size estimations for pairings. Cryptology ePrint Archive, Report 2017/334, 2017. https://eprint.iacr.org/2017/334.

[4] C. Baum, J. Bootle, A. Cerulli, R. del Pino, J. Groth, and V. Lyubashevsky. Sublinear lattice-based zero-knowledge arguments for arithmetic circuits. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 669–699, 2018.

[5] S. Bayer and J. Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 263–280, 2012.

[2] https://github.com/dalek-cryptography/zkp

Input size (bits)	Gates	Proof size (bytes)	Timing	
			Prove (s)	Helped batch (ms)
Pedersen hash preimage (input size)				
48	203	256	0.15	0.69
384	1562	256	0.84	0.72
Unpadded SHA256 preimage				
512	39,516	256	14.63	0.68
1024	78,263	256	28.93	0.68
1536	117,010	256	38.86	0.68

Table 4: Sonic's efficiency for proving knowledge of x such that $H(x) = y$ for different sized x's. Numbers are given to one significant digit. The first rows are for the Pedersen hash function and the final rows are for SHA256. "Helped batch" is the marginal cost of verifying an additional proof assuming that the helper has been run. These are calculated by batch-verifying 100 proofs, subtracting the cost to verify one, and dividing by 99.

[6] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 108–125, 2009.

[7] M. Bellare, G. Fuchsbauer, and A. Scafuro. NIZKs with an untrusted CRS: security in the face of parameter subversion. In *ASIACRYPT*, pages 777–804, 2016.

[8] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *EUROCRYPT*, pages 409–426, 2006.

[9] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. https://eprint.iacr.org/2018/046.

[10] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In *Proceedings of the IEEE Symposium on Security & Privacy*, 2014.

[11] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. Snarks for C: verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 90–108, 2013.

[12] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and M. Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *Proceedings of the IEEE Symposium on Security & Privacy*, 2015.

[13] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. *IACR Cryptology ePrint Archive*, 2018:828, 2018.

[14] E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 31–60, 2016.

[15] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Scalable zero knowledge via cycles of elliptic curves. In *CRYPTO*, 2014.

[16] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 781–796, 2014.

[17] D. Bernhard, G. Fuchsbauer, and E. Ghadafi. Efficient signatures of knowledge and DAA in the standard model. In *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, pages 518–533, 2013.

[18] N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth. Succinct non-interactive arguments via linear interactive proofs. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 315–333, 2013.

[19] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, pages 757–788, 2018.

[20] D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.

[21] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. Random oracles in a quantum world. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 41–69, 2011.

[22] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT*, 2016.

[23] J. Bootle, A. Cerulli, E. Ghadafi, J. Groth, M. Hajiabadi, and S. Jakobsen. Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In *Proceedings of Asiacrypt 2017*, 2017.

[24] J. Bootle, A. Cerulli, J. Groth, S. K. Jakobsen, and M. Maller. Nearly linear-time zero-knowledge proofs for correct program execution. *IACR Cryptology ePrint Archive*, 2018:380, 2018.

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

[26] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *Proceedings of the IEEE Symposium on Security & Privacy*, 2018.

[27] J. Camenisch, M. Dubovitskaya, K. Haralambiev, and M. Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 262–288, 2015.

[28] J. Camenisch and T. Groß. Efficient attributes for anonymous credentials. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 345–356, 2008.

[29] P. Chaidos, V. Cortier, G. Fuchsbauer, and D. Galindo. Beleniosrf: A non-interactive receipt-free electronic voting scheme. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1614–1625, 2016.

[30] M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1825–1842, 2017.

[31] M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn. Succinct malleable nizks and an application to compact shuffles. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 100–119, 2013.

[32] M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn. Malleable signatures: New definitions and delegatable anonymous credentials. In *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*, pages 199–213, 2014.

[33] G. Cormode, M. Mitzenmacher, and J. Thaler. Practical verified computation with streaming interactive proofs. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 90–112, 2012.

[34] R. Cramer, I. Damgård, and M. Keller. On the amortized complexity of zero-knowledge protocols. *J. Cryptology*, 27(2):284–316, 2014.

[35] G. Danezis, C. Fournet, J. Groth, and M. Kohlweiss. Square span programs with applications to succinct NIZK arguments. In *ASIACRYPT*, pages 532–550, 2014.

[36] J. Frankle, S. Park, D. Shaar, S. Goldwasser, and D. J. Weitzner. Practical accountability of secret processes. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, pages 657–674, 2018.

[37] G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology*

Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II, pages 33–62, 2018.
[38] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
[39] C. Garman, M. Green, and I. Miers. Decentralized anonymous credentials. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*, 2014.
[40] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT*, pages 626–645, 2013.
[41] C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 99–108, New York, NY, USA, 2011. ACM.
[42] I. Giacomelli, J. Madsen, and C. Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 1069–1083, 2016.
[43] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 113–122, 2008.
[44] M. Green and I. Miers. Bolt: Anonymous payment channels for decentralized currencies. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 473–489, 2017.
[45] J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT*, pages 305–326, 2016.
[46] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-snarks. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, pages 698–728, 2018.
[47] J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In *CRYPTO*, pages 581–612, 2017.
[48] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography with constant computational overhead. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 433–442, 2008.
[49] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.
[50] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 177–194. Springer, 2010.
[51] T. Kim and R. Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 543–571, 2016.
[52] A. E. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 839–858, 2016.
[53] Y. Lindell. Parallel coin-tossing and constant-round secure two-party computation. *J. Cryptology*, 16(3):143–184, 2003.
[54] H. Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, pages 41–60, 2013.
[55] H. Lipmaa, P. Mohassel, and S. S. Sadeghian. Valiant's universal circuit: Improvements, implementation, and applications. *IACR Cryptology ePrint Archive*, 2016:17, 2016.
[56] I. Meckler and E. Shapiro. Coda: Decentralized cryptocurrency at scale. 2018.
[57] C. Papamanthou, E. Shi, and R. Tamassia. Signatures of correct computation. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 222–242, 2013.
[58] B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *Proceedings of the IEEE Symposium on Security & Privacy*, 2013.
[59] L. G. Valiant. Universal circuits (preliminary report). In *Proceedings of the 8th Annual ACM Symposium on Theory of Computing*, pages 196–203, 1976.
[60] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish. Doubly-efficient zksnarks without trusted setup. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 926–943, 2018.
[61] Z. Wilcox. The design of the ceremony. https://z.cash/blog/the-design-of-the-ceremony.html, Oct. 2016.
[62] Z. S. Workshop, 2018. https://zkproof.org/proceedings-snapshots/zkproof-implementation-20180801.pdf.
[63] H. Wu, W. Zheng, A. Chiesa, R. A. Popa, and I. Stoica. DIZK: A distributed zero knowledge proof system. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, pages 675–692, 2018.
[64] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou. A zero-knowledge version of vsql. *IACR Cryptology ePrint Archive*, 2017:1146, 2017.

A THE POLYNOMIAL PERMUTATION ARGUMENT

The prover wishes to demonstrate the correct evaluation of

$$\Psi(X, Y) = \sum_{i=1}^{n} \psi_{\sigma_i} X^i Y^{\sigma_i}$$

for $y, z \in \mathbb{F}$. Observe that the permutation of this polynomial

$$\Phi(X, Y) = \sum_{i=1}^{n} \psi_i X^i Y^i$$

is such that $\Phi(X, Y) = \Phi(XY, 1)$. Therefore we can use arguments about the correct calculation of Φ together with a permutation argument to obtain arguments about the correct calculation of Ψ.

Our permutation argument given in Figure 8 is similar to that in [23]. If the prover commits to $f(X) = \sum_{i=1}^{n} a_i X^i$, then we have for random challenges $\beta, \gamma \in \mathbb{F}_p$ that

$$\prod_{i=1}^{n} a_i + \sigma_i \beta + \gamma = \prod_{i=1}^{n} \psi_i y^i + i\beta + \gamma \tag{2}$$

holds with non-negligible probability if and only if for all i, $a_i = \psi_{\sigma_i} y^{\sigma_i}$. Notice that if $a_i = \psi_{\sigma_i} y^{\sigma_i}$ then $\mathbf{a} + \beta\boldsymbol{\sigma}$ contains a permutation of the entries in $(\psi_1 y^1 + \beta, \ldots, \psi_n y^n + n\beta)$. However, if $\boldsymbol{a}$ is not a permutation, then overwhelming probability over β there will be entries that do not appear anywhere in $\mathbf{a} + \beta\boldsymbol{\sigma}$. The prover will now convince the verifier that (2) holds. By the Schwartz-Zippel lemma this is unlikely to hold over the random choice of γ unless indeed $\mathbf{a} + \beta\boldsymbol{\sigma}$ contains the correct permutation.

We include the following additional elements in our shared information.

$$\begin{aligned}
P_1 &= \mathsf{Commit}(bp, \mathtt{srs}, d, \textstyle\sum_{i=1}^{n} X^i) \\
P_2 &= \mathsf{Commit}(bp, \mathtt{srs}, d, \textstyle\sum_{i=1}^{n} \psi_i X^i) \\
P_3 &= \mathsf{Commit}(bp, \mathtt{srs}, d, \textstyle\sum_{i=1}^{n} i X^i) \\
P_4 &= \mathsf{Commit}(bp, \mathtt{srs}, d, \textstyle\sum_{i=1}^{n} \sigma_i X^i)
\end{aligned}$$

The prover calculates S', a commitment to $\Psi(X, y)$. The verifier checks that the commitment to Φ is computed correctly. The srs contains P_2, a commitment to $\Psi(X, 1)$. The prover opens S' at u and P_2 at uy. If the opening are equal and verify then with overwhelming probability the commitment is correct.

The prover commits to $S = \Phi(X, y)$. In order to check that the coefficients of S are the permutation of the coefficients of $\Psi(X, y)$, the verifier chooses random challenges $\beta, \gamma \in \mathbb{F}$ and asks the prover to demonstrate that the product of the coefficients of $SP_4^{\beta} P_1^{\gamma}$ is equal to the product of the coefficients of $S' P_3^{\beta} P_1^{\gamma}$, thus simulating the argument from Equation 2.

The prover then opens S at z to ψ which the verifier checks. If all the checks hold then we have that $\psi = \Psi(z, y)$.

Lemma A.1. *The permutation argument in Figure 8 is sound when instantiated using a bounded extractable polynomial commitment*

Common Input: info = $bp, \mathsf{srs}, y, z,$
$P_1 = \mathsf{Commit}(bp, \mathsf{srs}, d, \sum_{i=1}^{n} X^i), P_2 = \mathsf{Commit}(bp, \mathsf{srs}, d, \sum_{i=1}^{n} \psi_i X^i),$
$P_3 = \mathsf{Commit}(bp, \mathsf{srs}, d, \sum_{i=1}^{n} iX^i), P_4 = \mathsf{Commit}(bp, \mathsf{srs}, d, \sum_{i=1}^{n} \sigma_i X^i),$
$\Psi(X, Y) = \sum_{i=1}^{n} \psi_{\sigma_i} X^i Y^{\sigma_i}, \Phi(X, Y) = \sum_{i=1}^{n} \psi_i X^i Y^i$

$\underline{\mathsf{permP}_1(\mathsf{info}) \mapsto (S, S')}$:
for $1 \le j \le M$:
  $S \leftarrow \mathsf{Commit}(bp, \mathsf{srs}, d, \Psi(X, y))$
  $S' \leftarrow \mathsf{Commit}(bp, \mathsf{srs}, d, \Phi(X, y))$
return S, S'

$\underline{\mathsf{permV} \mapsto \mathsf{permP}}$:
Send $u, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p$ to prover

$\underline{\mathsf{permP}_2(u, \beta, \gamma) \mapsto (s, v, W, W', Q', \pi)}$:
$\bar{S} \leftarrow S P_4^{\beta} P_1^{\gamma}$
$\bar{P} \leftarrow S' P_3^{\beta} P_1^{\gamma}$

$(\psi = \Psi(z, y), W) \leftarrow \mathsf{Open}(bp, \mathsf{srs}, S, z, \Psi(X, y))$
$(v, W') \leftarrow \mathsf{Open}(bp, \mathsf{srs}, S', u, \Phi(X, y))$
$(v, Q') \leftarrow \mathsf{Open}(bp, \mathsf{srs}, P_2, uy, \sum_{i=1}^{n} \psi_i X^i)$

$\bar{s}(X) \leftarrow \sum_{i=1}^{n} \psi_{\sigma_i} y^{\sigma_i} X^i + \beta \sigma_i X^i + \gamma X^i$
$\bar{p}(X) \leftarrow \sum_{i=1}^{n} \psi_i y^i X^i + \beta i X^i + \gamma X^i$

gprod $\leftarrow \mathsf{gprodP}(\bar{S}, \bar{P}, \bar{s}(X), \bar{p}(X))$
return $(\psi, W, v, W', Q', \pi)$

$\underline{\mathsf{permV}(\mathsf{info}, (\psi, (S, S', W, v, W', Q', \mathsf{gprod})), u, \beta, \gamma) \mapsto 0/1}$:
$\bar{S} \leftarrow S P_4^{\beta} P_1^{\gamma}$
$\bar{P} \leftarrow S' P_3^{\beta} P_1^{\gamma}$
check $\mathsf{pcV}(bp, \mathsf{srs}, S, d, z, (s, W))$
check $\mathsf{pcV}(bp, \mathsf{srs}, S', d, u, (v, W'))$
check $\mathsf{pcV}(bp, \mathsf{srs}, P_2, d, uy, (v, Q'))$
check $\mathsf{gprodV}(\bar{S}, \bar{P}, \mathsf{gprod})$
return 1 if all checks pass, else return 0

Figure 8: The permutation argument.

scheme and a sound grand-product argument. If the polynomial commitment scheme and the grand-product argument are both sound against an algebraic adversary that can update the SRS, then so is the permutation argument.

Proof. The extractability of the polynomial commitment scheme gives us that there exists algebraic extractors that output degree d Laurent polynomials $s(X), s'(X), p_2(X)$ such that $s = s(z), v = s'(u)$ and $v = p_2(uy)$. If $p_2(uy) \neq \sum_{i=1}^{n} \psi_i u^i y^i$ then the adversary can find a second opening for P_2 and in doing so break evaluation binding of the commitment scheme. The probability that $s'(u) = v$ but $s'(u) \neq p_2(Xy)$ is negligible by the Schwartz-Zippel Lemma.

The soundness of the grand-product argument gives us that

$$\begin{aligned}&\textstyle\prod_{i=1}^{n} s_i + \beta\sigma_i + \gamma = \prod_{i=1}^{n} s'_i + \beta i + \gamma \\ &\Leftarrow \\ &\textstyle\prod_{i=1}^{n} s_i + \beta\sigma_i + \gamma = \prod_{i=1}^{n} \psi_i y^i + \beta i + \gamma\end{aligned}$$

Again by the Schwartz-Zippel Lemma, this implies that $s_i = \psi_i$ with all but negligible probability. This concludes the argument. □

B THE GRAND PRODUCT ARGUMENT

For our signature of correct computation in Section 7 we require a grand product argument. Namely we need the prover to demonstrate that the product of the coefficients of two commitments U and V are equal, where U and V are fully well-formed commitments to n-degree polynomials

$$U = g^{\alpha \sum_{i=1}^{n} a_i x^i},\ V = g^{\alpha \sum_{i=1}^{n} a_{i+n+1} x^i}.$$

We further assume that the polynomials do not have a constant term. We can interpret $UV^{x^{n+1}}$ as a commitment to

$$f(X) = \sum a_i X^i.$$

We wish to demonstrate that

$$\prod_{i=1}^{n} a_i = \prod_{i=n+2}^{2n+2} a_i. \tag{3}$$

We can represent these requirements with the following constraints system.

(1) $\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{c}$

(2) $\boldsymbol{b} = (1, c_1, \ldots, c_{2n+1})$
(3) $c_{n+1} = 1$
(4) $c_{2n+1} = c_n$

We shall show that this constraint system is satisfied using our Sonic argument described in Section 6. However, because all but two of our constraints are shift constraints, we can adapt the polynomial that the verifier must compute. Our adapted polynomial can be computed using a small number of field operations, thus the signature of correct computation is not required (otherwise we would be using a signature of computation to build a signature of computation).

B.1 Polynomial Encoding of Constraints

We follow the principles of our main argument by encoding the constraint system into a single equation in formal indeterminate Y.

$$c_{n+1} - 1 + (c_n - c_{2n+1})Y + \sum_{i=1}^{2n+1} (a_i b_i - c_i) Y^{i+1} = 0 \qquad (4)$$

We design a polynomial $t(X, Y)$ for which the left hand side of Equation 4 is the constant term.

$$\begin{array}{lcl} r(X,Y) & = & Y\left(\sum_{i=1}^{2n+1} a_i X^i Y^i + c_n^{-1} X^{n+1} Y^{n+1}\right) \\ s(X,Y) & = & X^{n+2} + X^{n+1}Y - X^{2n+2}Y \\ r'(X,Y) & = & \sum_{i=1}^{2n+2} b_i X^{-i} \\ & = & \sum_{i=1}^{2n+2} c_i X^{-i-1} + X^{-1} \\ k(Y) & = & 1 + \sum_{i=1}^{2n+1} c_i Y^{i+1} \\ t(X,Y) & = & (r(X,Y) + s(X,Y))r'(X,Y) - k(Y) \end{array}$$

If we have that Equation 4 is satisfied then at all y we have that $t_j(X, y)$ has a constant term of zero. Otherwise, it has a nonzero constant term at most y and so also at random y with high probability, given a large enough field.

B.2 Protocol for the Grand-Product Argument

Our protocol for a grand product argument is given in Figure 9. It begins by asking the prover to provide the commitments

$$\left\{ \; C = g^{\alpha \sum_{i=1}^{2n+1} c_i x^i}, \quad c_n^{-1} \; \right\}_{j=1}^{M}$$

for which the prover must show that C has no negative exponents of X. The verifier samples challenge $y \xleftarrow{\$} \mathbb{F}_p$ and asks the prover to commit to

$$T = g^{\alpha t(x,y)}.$$

The verifier now samples $z \xleftarrow{\$} \mathbb{F}_p$ and asks the helper to open

$$\begin{array}{ll} g^{\alpha c_n^{-1} x^{n+1}} U V^{x^{n+1}} \text{ at } yz \text{ to } v_a & C \text{ at } z^{-1} \text{ to } v_c \\ C \text{ at } y \text{ to } v_k & T \text{ at } z \text{ to } t. \end{array}$$

Given these evaluation, the verifier can compute

$$\begin{array}{lcl} r(z,y) & = & y v_a \\ s(z,y) & = & z^{n+2} + z^{n+1} y - z^{2n+2} y \\ r'(z,y) & = & v_c z^{-1} + z^{-1} \\ k(y) & = & v_k y + 1 \end{array}$$

and we have

$$t(z,y) = (r(z,y) + s(z,y))r'(z,y) - k(y)$$

The verifier can now check that $t = t(z, y)$ demonstrating that the earlier commitment to $t(X, y)$ was computed correctly with respect to $UV^{x^{n+1}}$ and C, and that it has a constant term of zero, completing the argument.

Lemma B.1. *The grand-product argument in Figure 9 is sound when instantiated with a bounded extractable polynomial commitment scheme and a sound well-formness argument. If the permutation argument and the well-formedness argument are sound against an algebraic adversary that can update the SRS, then so is the grand-product argument.*

Proof. By the extractability of the polynomial commitment scheme, there exists an algebraic extractor that outputs polynomials $a(X), c(X), t(X)$ such that $v_a = a(yz)$, $v_c = c(z^{-1})$, $v_k = c(y)$ and $t = t(z)$. By the well-formedness argument, $c(X)$ cannot have negative powers. By the well-formedness argument, U and V have algebraic representations with powers between 1 and n. The pairing equation gives us that

$$a(X) = c_n^{-1} X^{n+1} + u(X) + x^{n+1} v(X).$$

The verifier computes $s(z, y)$ for itself. The verifier also learns that the coefficients of v_c and v_k are consistent, otherwise an adversary could open the same commitment to two different polynomial evaluations and break evaluation binding. Thus r' and k are calculated correct. Further, because the prover opens T to

$$t = a(b + s) - k(y)$$

$t(X)$ cannot have a non-zero X^0 coefficient (otherwise an adversary could break the bounded property of the polynomial commitment scheme).

Suppose this holds for $2n + 4$ different challenges $y \in \mathbb{Z}_p$. Then we have equality of polynomials in Appendix B.1 since a non-zero polynomial of degree $2n + 4$ cannot have $2n + 3$ roots i.e.

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

has no constant term. This implies that $u(X)$ and $v(X)$ define a valid opening. □

B.3 Well-formedness Argument

Our techniques for the grand-product argument require us to ensure that a number of elements computed during the protocol are commitments to polynomials of the form

$$f(X) = \sum_{i=1}^{n} a_i X^i$$

for some n-length vector $\mathbf{a}$. If we have that

$$F = g^{\alpha f(x)}$$

the prover sends

$$\begin{array}{ll} L & = g^{x^{-d} f(x)} \\ R & = g^{x^{d-n} f(x)} \end{array}$$

which the verifier can check with the pairings

$$\begin{array}{ll} e(F,h) & = e(L, h^{\alpha x^d}) \\ e(F,h) & = e(R, h^{\alpha x^{n-d}}) \end{array}$$

Common Input: $\mathsf{info} = bp, \mathsf{srs}, U, V, e(g^{x^{n+1}}, h^{\alpha})$
Prover's Input: $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ such that $U = g^{\alpha \sum_{i=1}^{n} a_i x^i}, V = g^{\alpha \sum_{i=1}^{n} a_{i+n+1} x^i}$

$\underline{\mathsf{gprodP}_1(\mathsf{info}, (a_1, \ldots, a_{2n+1})) \mapsto (A, C, C_w, U_w, U_v, c_n^{-1})}$:
$a_{n+1} \leftarrow c_n^{-1}$
$A \leftarrow g^{a_{n+1}\alpha x^{n+1}} U V^{x^{n+1}}$
$C \leftarrow \mathsf{Commit}(bp, \mathsf{srs}, d, c(X))$

$C_w \leftarrow \mathsf{wformP}(bp, \mathsf{srs}, 2n+1, C, \boldsymbol{c})$
$U_w \leftarrow \mathsf{wformP}(bp, \mathsf{srs}, n, U, (a_1, \ldots, a_n))$
$U_v \leftarrow \mathsf{wformP}(bp, \mathsf{srs}, n, V, (a_{n+2}, \ldots, a_{2n+1}))$
return $(A, C, C_w, U_w, U_v, c_n^{-1})$

$\underline{\mathsf{gprodV} \mapsto \mathsf{gprodP}}$:
Send $y \xleftarrow{\$} \mathbb{Z}_p$ to prover

$\underline{\mathsf{gprod}_2(y) \mapsto T}$:
$T \leftarrow \mathsf{Commit}(bp, \mathsf{srs}, d, t(X, y))$ return T

$\underline{\mathsf{gprodV} \mapsto \mathsf{gprodP}}$:
Send $z \xleftarrow{\$} \mathbb{Z}_p$ to prover

$\underline{\mathsf{gprod}_3(z) \mapsto T}$:
$(v_a, W_a) \leftarrow \mathsf{Open}(A, yz, a(X))$
$(v_c, W_c) \leftarrow \mathsf{Open}(C, z^{-1}, c(X))$
$(v_k, W_k) \leftarrow \mathsf{Open}(C, y, c(X))$
$(t, W_t) \leftarrow \mathsf{Open}(T, z, t(X))$
return $((v_a, W_a), (v_c, W_c), (v_k, W_k), W_t)$

$\underline{\mathsf{gprodV}\begin{pmatrix} \mathsf{info}, (A, C, C_w, U_w, U_v, c_n^{-1}, T, \\ (v_a, W_a), (v_c, W_c), (v_k, W_k), W_t), y, z \end{pmatrix} \mapsto 0/1}$:
$r \leftarrow y v_a$
$s \leftarrow z^{n+2} + z^{n+1} y - z^{2n+2} y$
$r' \leftarrow v_c z^{-1}$
$k \leftarrow v_k y + 1$
$t \leftarrow (r + s) r' - k$

check $e(A, h) = e(g^{\alpha a_{n+1} x^{n+1}} U, h) e(V, h^{x^{n+1}})$
check $\mathsf{pcV}(bp, \mathsf{srs}, A, d, yz, (v_a, W_a))$
check $\mathsf{pcV}(bp, \mathsf{srs}, C, d, z^{-1}, (v_c, W_c))$
check $\mathsf{pcV}(bp, \mathsf{srs}, C, d, y, (v_k, W_k))$
check $\mathsf{pcV}(bp, \mathsf{srs}, T, d, (t, W_t))$

check $\mathsf{wformV}(bp, \mathsf{srs}, 2n+1, C, C_w)$
check $\mathsf{wformV}(bp, \mathsf{srs}, n, U, U_w)$
check $\mathsf{wformV}(bp, \mathsf{srs}, n, V, V_w)$
return 1 if all checks pass, else return 0

Figure 9: The grand-product argument.

C BATCHING ARGUMENTS FOR IMPROVED EFFICIENCY

The unhelped Sonic protocol uses $3 + 7M$ polynomial commitments where M is the number of permutations required to represent the computation. Assuming $M = 3$, this means there are 24 polynomial commitment arguments. By having the prover batch some of these arguments together, we can reduce the total number of polynomial commitments to $7 + 3M$. As a result, the proofs for our unhelped Sonic protocol has 20 elements in $\mathbb{G}_1$ and 16 elements in $\mathbb{Z}_p$. Assuming a group size and field size of 256 bits, this means the proof sizes are approximately $1kb$.

C.1 Batching Polynomial Commitments

Suppose that the prover is required to open commitments

$$F_1, \ldots, F_k$$

with maximum degree $\mathsf{max}_1, \ldots, \mathsf{max}_k$ at the same randomly chosen point z. To avoid encountering the same costs k times, the prover first engages with the verifier.

$\underline{\mathsf{P} \mapsto \mathsf{V}:}$
The prover sends $F_1, \ldots, F_k$.

$\underline{\mathsf{V} \mapsto \mathsf{P}:}$
The verifier sends random z to the prover.

$\underline{\mathsf{P} \mapsto \mathsf{V}:}$
The prover sends $v_1, \ldots, v_k$
It claims these are the correct openings at z.

$\underline{\mathsf{V} \mapsto \mathsf{P}:}$
The verifier sends random γ to the prover.

$\underline{\mathsf{P} \mapsto \mathsf{V}:}$
The prover sets $w(X) = \frac{\sum_{i=1}^{k} \gamma^i (f_i(X) - f_i(z))}{X - z}$.
They return $g^{w(x)}$.

$\underline{\mathsf{V} \mapsto \mathsf{P}:}$
The verifier sets $F_T = \prod_{i=1}^{k} e(F_i^{\gamma^i}, h^{\alpha x^{-d+\max_i}})$.
They set $v = \prod_{i=1}^{k} v_i \gamma^i$.
They check $e(W, h^{\alpha x}) e(g^v W^z, h^\alpha) = F$.

Observe that the probability that at random γ,

$$\sum_{i=1}^{k} v_i \gamma^i = \sum_{i=1}^{k} f_i(z) \gamma^i$$

but at some i

$$v_i \neq f_i(z)$$

is negligible in a sufficiently large field. Further observe that F_T contains $\sum_{i=1}^{k} f_i(x) \gamma^i$ in the target group. This can be proven secure using a similar argument to that in Theorem 6.3.

C.2 Batching Grand-Product Arguments

The prover is required to show that

$$(\bar{S}_1, \bar{P}_1), \ldots, (\bar{S}_M, \bar{P}_M)$$

all satisfy a grand-product argument. Thus they know

$$(\bar{s}_1(X), \bar{p_1}(X)), \ldots, (\bar{s}_M(X), \bar{p}_M(X))$$

such that

$$\prod_j \bar{s}_{i,j} = \prod_j \bar{p}_{i,j}$$

for all $1 \leq i \leq M$ and

$$\bar{S}_i = g^{\alpha \bar{s}_i(x)} \text{ and } \bar{P}_i = g^{\alpha \bar{p}_i(x)}.$$

Each grand-product argument costs a well-formedness argument and 4 polynomial commitments. To avoid encountering these costs M times, the prover first engages with the verifier.

$\underline{\mathsf{P} \mapsto \mathsf{V}:}$
The prover sends $(\bar{S}_1, \bar{P}_1), \ldots, (\bar{S}_M, \bar{P}_M)$.

$\underline{\mathsf{V} \mapsto \mathsf{P}:}$
The verifier sends random z to the prover.

$\underline{\mathsf{P} \mapsto \mathsf{V}:}$
The prover sets $\bar{S} = \prod_{i=1}^{M} \bar{S}_i^{z^i}$ and $\bar{P} = \prod_{i=1}^{M} \bar{P}_i^{z^i}$.
They send a grand-product argument for $\bar{S}$ and $\bar{M}$.

Observe that the probability that at random z,

$$\sum_{i=1}^{M} \prod_j \bar{s}_{i,j} z^i = \sum_{i=1}^{M} \prod_j \bar{p}_{i,j} z^i$$

but at some i

$$\prod_j \bar{s}_{i,j} \neq \prod_j \bar{p}_{i,j}$$

is negligible in a sufficiently large field.