AuroraLight: Improved prover efficiency and SRS size in a Sonic-like system

Ariel Gabizon
Protocol Labs

Abstract

Using ideas from the recent Aurora zk-STARK of Ben-Sasson et al. [BCR+19], we present a zk-SNARK with a universal and updatable SRS similar to the recent construction of Maller et al. [MBKM19], called Sonic. Compared to Sonic, our construction achieves significantly better prover run time (less than half) and smaller SRS size (one sixth). However, we only achieve amortized succinct verification time for batches of proofs, either when the proofs are generated in parallel or in [MBKM19]'s helper setting, and our proofs are longer than those of [MBKM19] (but still contain a *constant* number of field and group elements).

1 Introduction

Due to real-world deployments of zk-SNARKs, e.g. Zcash; it has become of significant interest to have the structured reference string (SRS) be constructible in a "universal and updatable" fashion. Meaning that the same parameters (a different term for the SRS) can be used for statements about all circuits/computations of a certain bounded size; and that at any point in time the parameters can be updated by a new party, such that the honesty of only one party from all updaters up to that point is required for soundness. A natural direction to achieve this, is by having an "SRS of monomials" meaning that the SRS consists solely of elements g^{x^i} for a secret uniform x. Roughly speaking, such elements are easy to update as the new party can just raise them to power s^i for their secretly chosen s.

Maller et al. [MBKM19] noticed that the polynomial commitment scheme of Kate, Zaverucha and Goldberg [KZG10] is very helpful in this context, as such an SRS of monomials is sufficient in [KZG10] to verifiably evaluate values of low-degree polynomials committed to by the prover - this being perhaps the main problem to be dealt with (without the luxury of an SRS) in PCP constructions using techniques such as low-degree testing (starting from [BFL91]) and PCP's of proximity [BS08] (and in more modern works also IOP's of proximity [BCG+17]).

The Sonic approach [MBKM19]'s Sonic uses a variant of the arithmetization of Bootle et al. [BCC+16] to create a situation where a certain polynomial of the prover can have a zero constant coefficient only when the prover has a satisfying assignment. Roughly, this is achieved by using Laurent polynomials with both negative and positive powers, such that their product cancels out in the constant term only when a satisfying assignment was used to construct them. A lot of the ingenuity of [MBKM19] is that though the polynomials from [BCC+16] are *bi-variate* and [KZG10] is only efficient for univariate polynomials (in the sense of linear time proving in the degree) they

are able to maneuver in a way that only requires to commit and evaluate *univariate restrictions* of the original bi-variates. However, using negative powers for the cancellation trick comes at a price of having the polynomials ultimately plugged in to the [KZG10] scheme have a larger range of powers - roughly a $7n$ size range from $-4n$ to $3n$ (where n is the number of multiplication gates in our circuit). Furthermore, [MBKM19] need to put a "hole" in this range to disallow the constant coefficient, which doubles the SRS size.

The main point is that the number of prover exponentiations and SRS size is directly tied to the size of this range when using [KZG10].

The Aurora approach Instead, we use a nice algebraic trick from the Aurora IOP[BCR+19]. [BCR+19] uses a lemma (Lemma 4.1) connecting between the value of a polynomial's constant coefficient, and its sum on a multiplicative subgroup. Together with a randomized sum check arithmetization in the style of [BCG+17, BCR+19], this allows us to create a "constant coefficient=0 iff prover has witness" situation without using negative powers. This enables only dealing with (regular not Laurent) polynomials of degree at most $2n$; i.e. we have reduced the "monomial range" of Sonic from $7n$ to $2n$, and do not need to deal with a "hole" in this range disallowing a constant coefficient. Morever, all but one of the prover polynomials will have degree at most n, which further helps reduce prover run time. On the other hand, we have five polynomials to deal with rather than two as in [MBKM19], which is why our proofs are longer than Sonic's.

Before comparing performance with [MBKM19] in more detail, we discuss the three settings in which both our construction and Sonic can be used.

1.1 The three modes of Sonic

The basic version of the Sonic verifier, as well as ours, is succinct (i.e. polylog(λ) running time) *except* for the need to evaluate a polynomial $s(X, Y)$ of $O(n)$ monomials. The evaluation of s in Sonic is done once per proof, at a point (z, y) chosen by the verifier during the protocol. There are three ways to deal with this evaluation.

Parallel proof generation for a batch: The crucial point is that the values z, y are only revealed to the prover at certain points in the protocol; but beyond this, there is no dependence between the values z, y and the specific statement proven - they just need to be uniform. It follows that if proofs are generated in parallel, e.g. in practice by posting the first part of all proofs on a blockchain and getting the random challenge z afterwords and continuing similarly for y - then the verifier can use the same (z, y) for a whole batch of proofs and do the non-succinct s evaluation only once per batch.[1] Our construction also has the property that the non-succinct verifier computations can be done only once per batch.

Arbitrary batching using an untrusted helper: [MBKM19] develop a method where a batch of evaluations $\{s(z_j, y_j)\}_{j\in[m]}$ can be performed by an untrusted helper and a proof can be given to the verifier that the evaluations are all correct. Verification of this proof only requires *one evaluation of* s. Hence, using this "helper mode" we can verify an arbitrary batch of proofs generated without any synchronization, with a non-succinct operation done only once per batch. This mode incurs the

[1] [MBKM19] do not explicity discuss this mode, but we think it may be the most practical both for their construction and ours.

cost of needing such a helper, and adding the evaluation $s(z_j, y_j)$ and proof of correct evaluation, which increase the proof size. Our scheme can also leverage the Sonic helper, however with larger cost to proof size as we have six polynomials per proof that need to be evaluated by the verifier instead of one. See details in Section 5.

Fully succinct mode: [MBKM19] in fact manage to design a proof system that allows a fully succinct (i.e. not just in the amortized sense) verifier to verify that the evaluation $s(z, y)$ is correct. However, their fully succinct mode requires significant increase of the constants in proof size, proving and verification time and may be less practical. Our construction does not currently support this mode, but it seems possible an extension to this mode can be done. We pose this as an open question with more details in Section 7.

1.2 Our results compared to [MBKM19]

We compare the performance of Sonic to our system when generating proofs for arithmetic circuits with n multiplication gates, or analogously, an R1CS system with n constraints. Motivated by the discussion in Section 1.1 about settings of batched proofs - we omit from the tables the once per batch computations of the verifier. These are similar in Sonic and our system and are linear in the circuit size.

The first two columns in Table 1 describe SRS size when only knowing a bound d on the circuit size/number of R1CS constraints; and then the reduced size possible to work with when knowing the exact size n of the circuit.

When describing proof sizes, we separate between the elements from the prover in the parallel generation setting, and the additional elements from the helper in the helped setting.

We omit $O(1)$ factors. For example, in the fixed circuit size SRS size, we omit the constant number of $\mathbb{G}_2$ elements needed in the SRS both in Sonic and our system. We also mention that adding zero-knoweldge increases by a small constant the number of prover exponentiations both in our system, and it seems, in Sonic.

In a nutshell, our construction has better prover run time and SRS size; while Sonic has smaller proofs, less auxiliary data and extra verifier work in helper mode, and a fully succinct verifier mode. Thus, the advantage of this work is most prominent in the parallel proof generation setting.

	size $\leq d$ SRS	**size $= n$ SRS**	**prover work**	**proof length**
Sonic	$12d$ $\mathbb{G}_1$, $12d$ $\mathbb{G}_2$	$12n$ $\mathbb{G}_1$	$18n$ $\mathbb{G}_1$ exp	4 $\mathbb{G}_1$, 2 $\mathbb{F}$
This work	$2d$ $\mathbb{G}_1$, $2d$ $\mathbb{G}_2$	$2n$ $\mathbb{G}_1$	$8n$ $\mathbb{G}_1$ exp	6 $\mathbb{G}_1$, 4 $\mathbb{F}$

Table 1: Prover comparison

1.3 Organization of paper

Section 2 contains terminology we will use. Section 3 contains an adaptation of the [KZG10] scheme similar to that of [MBKM19] that we will use. Section 4 contains our main construction assuming a polynomial commitment scheme as a black box. Section 5 shows how to adapt the Sonic helper

	verifier work	elem. from helper	extra verifier work in helper mode	fully succinct ver. mode?
Sonic	$5P$	3 $\mathbb{G}_1$, 2 $\mathbb{F}$	$4P$	Yes
This work	$5P$	8 $\mathbb{G}_1$, 10 $\mathbb{F}$	$12P$	No

Table 2: Verifier comparison per proof in batch, P=pairing

mode to our construction. Section 6 shows how to add zero-knoweldge to our construction. Section 7 discusses a possible extension to get a fully succinct verifier.

2 Terminology/conventions

We assume our field $\mathbb{F}$ is of prime order. We denote by $\mathbb{F}_{<d}[X]$ the set of univariate polynomials over $\mathbb{F}$ of degree smaller than d. We assume all algorithms described receive as an implicit parameter the security parameter λ.

Whenever we use the term "efficient", we mean an algorithm running in time poly(λ). Furthermore, we assume an "object generator" $\mathcal{O}$ that is run with input λ before all protocols, and returns all fields and groups used. Specifically, in our protocol $\mathcal{O}(\lambda) = (\mathbb{F}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e, g_1, g_2, g_t)$ where

- $\mathbb{F}$ is a prime field of super-polynomial size $r = \lambda^{\omega(1)}$.
- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ are all groups of size r, and e is an efficiently computable non-degenerate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$.
- g_1, g_2 are uniformly chosen generators such that $e(g_1, g_2) = g_t$.

We usually let the λ parameter be implicit, i.e. write $\mathbb{F}$ instead of $\mathbb{F}(\lambda)$. We write $\mathbb{G}_1$ and $\mathbb{G}_2$ additively. We use the notations $[x]_1 := x \cdot g_1$ and $[x]_2 := x \cdot g_2$.

Random oracles We describe public-coin interactive protocols between a prover and verifier; when deriving results for non-interactive protocols, we implicitly assume we can get a proof length equal to the total communcation of the prover, using the Fiat-Shamir transform/a random oracle. Using this reduction between interactive and non-interactive protocols, we can refer to the "proof length" of an interactive protocol.

3 The [KZG10] scheme as used in Sonic

We recall the [KZG10] scheme with two enhancements of [MBKM19] that will be important for us: The first is enabling a commitment to all degrees up to a certain size, rather than just the maximal degree. The second is "batch openings" of several polynomials at the same point $z \in \mathbb{F}$. We define a polynomial commitment scheme with these enhancements in mind.

Definition 3.1. *A d-polynomial commitment scheme consists of*

- $\mathsf{Gen}(d)$- *is a randomized algorithm that outputs an SRS σ.*

- $\mathsf{com}(f, \mathbf{max}, \sigma)$- *that given a polynomial* $f \in \mathbb{F}_{<\mathbf{max}}[X]$*, where* $\mathbf{max} \leq d$*, returns a commitment* cm *to* f*.*

- *A public coin protocol* open *between parties* $\mathrm{P}_{\mathsf{poly}}$ *and* $\mathrm{V}_{\mathsf{poly}}$*.* $\mathrm{P}_{\mathsf{poly}}$ *is given* $f_1, \ldots, f_t \in \mathbb{F}_{<d}[X]$*.* $\mathrm{P}_{\mathsf{poly}}$ *and* $\mathrm{V}_{\mathsf{poly}}$ *are both given integer* $t = \mathrm{poly}(\lambda)$*,* $\mathsf{cm}_1, \ldots, \mathsf{cm}_t$ *- the alleged commitments to* $f_1, \ldots, f_t$*, integers* $0 < d_1, \ldots, d_t \leq d$*,* $z \in \mathbb{F}$ *and* $s_1, \ldots, s_t \in \mathbb{F}$ *- the alleged correct openings* $f_1(z), \ldots, f_t(z)$*. At the end of the protocol* $\mathrm{V}_{\mathsf{poly}}$ *outputs* acc *or* rej*.*

such that

- ***Completeness:*** *Fix integer* t*,* $z \in \mathbb{F}$*,* $f_1, \ldots, f_t \in \mathbb{F}_{<d}[X]$ *and* $0 < d_1, \ldots, d_t \leq d$ *such that* $\deg(f_i) < d_i$*. Suppose that for each* $i \in [t]$*,* $\mathsf{cm}_i = \mathsf{com}(f_i, d_i, \sigma)$*. Then if* open *is run correctly with values* $t, z, \{\mathsf{cm}_i, d_i, f_i(z)\}_{i \in [t]}$*,* $\mathrm{V}_{\mathsf{poly}}$ *outputs* acc *with probability one.*

- ***Knowledge soundness in the generic group model:*** *There exists an e cient* E *such that for any e cient generic group model adversary* $\mathcal{A}$ *and arbitrary e cient* $\mathcal{A}'$*, the probability of* $(\mathcal{A}, \mathcal{A}')$ *winning the following game is* $\mathsf{negl}(\lambda)$ *over the randomness of* $(\mathcal{A}, \mathcal{A}')$ *and* Gen*.*

  1. *Given* σ*,* $\mathcal{A}$ *outputs* $t, \mathsf{cm}_1, \ldots, \mathsf{cm}_t$*.*
  2. E*, given access to the state of* $\mathcal{A}$ *outputs* $f_1, \ldots, f_t \in \mathbb{F}_{<d}[X]$*.*
  3. $\mathcal{A}'$ *outputs* $0 < d_1, \ldots, d_t \leq d$*,* $s_1, \ldots, s_t \in \mathbb{F}, z \in \mathbb{F}$*.*
  4. $\mathcal{A}'$ *takes the part of* $\mathrm{P}_{\mathsf{poly}}$ *in the protocol* open *with inputs* $\mathsf{cm}_1, \ldots, \mathsf{cm}_t, d_1, \ldots, d_t, s_1, \ldots, s_t$*.*
  5. $(\mathcal{A}, \mathcal{A}')$ *wins if*
     - $\mathbf{V}$ *outputs* acc *at the end of the protocol.*
     - *For some* $i \in [t]$*,* $s_i \neq f_i(z)$ *or* $\deg(f_i) \geq d_i$*.*

We describe the following scheme based on [KZG10, MBKM19]. It is in fact a slightly simpler scheme than in [MBKM19] because, as explained in the introduction, there is no need to deal with holes in the allowed range of degrees.

1. $\mathsf{Gen}(d)$ - choose uniform $x \in \mathbb{F}$. Output $\sigma = \left([1]_1, [x]_1, \ldots, \left[x^{d-1}\right]_1, [1]_2, \left[x^{-1}\right]_2, \ldots, \left[x^{-(d-1)}\right]_2\right)$.
2. $\mathsf{com}(f, d', \sigma) := \left[x^{d-d'} \cdot f(x)\right]_1$.
3. $\mathsf{open}(\{cm_i\}, \{d_i\}, \{s_i\}, z)$
   (a) $\mathrm{V}_{\mathsf{poly}}$ sends random $\gamma \in \mathbb{F}$.
   (b) $\mathrm{P}_{\mathsf{poly}}$ computes the polynomial
   $$h(X) := \sum_{i=1}^{t} \gamma^i \cdot \frac{f_i(X) - f_i(z)}{X - z}$$
   and using σ computes and sends $W := [h(x)]_1$.
   (c) $\mathrm{V}_{\mathsf{poly}}$ computes the elements
   $$F := \prod_{i \in [t]} e\left(\gamma^i \cdot \mathsf{cm}_i, \left[x^{d_i - d}\right]_2\right), v := \left[\sum_{i \in [t]} \gamma^i \cdot s_i\right]_1$$

(d) $\mathrm{V}_{\mathsf{poly}}$ computes outputs acc if and only if

$$F = e\left(v - z \cdot W, [1]_2\right) \cdot e(W, [x]_2).$$

Note that $|\sigma| = 2d$ in the above scheme. ***However***, a crucial point is that once we fix ℓ values from which we will always choose $\{d_1, \ldots, d_t\}$, we can work with a subvector of σ of size $d + \ell$. In our SNARK, given a circuit size, we will only need $\ell = 3$ of the $\mathbb{G}_2$ values from σ in our SRS.

The following is implied almost directly by Theorem 6.1 and Appendix C.1 of [MBKM19]. We leverage that the pairings in step 3c can be batched for indices i, j such that $d_i = d_j$.

Lemma 3.2. *Fix any d and assume the d-power bi-linear Strong Di e Hellman assumption holds. Then the above scheme is a d-polynomial commitment scheme such that*

1. $\mathrm{P}_{\mathsf{poly}}$ *requires* $\mathbf{max}$ $\mathbb{G}_1$ *exponentiations for computing* $\mathsf{com}(f, \mathbf{max}, \sigma)$.
2. $\mathrm{P}_{\mathsf{poly}}$ *requires* $\sum_{i=1}^{t} O(d_i \log(d_i))$ *eld operations and* $\sum_{i=1}^{t} d_i$ *exponentiations for computing* $\mathsf{open}(\{cm_i\}_{i\in[t]}, \{d_i\}_{i\in[t]}, \{s_i\}_{i\in[t]}, z)$.
3. $\mathrm{V}_{\mathsf{poly}}$ *requires* $t^* + 2$ *pairings, where* t^* *is the number of distinct values amongst* $d_1, \ldots, d_t$.

4 The main construction

We begin by converting R1CS to a format that is convenient for us. We denote by n the number of private inputs and the number of constraints which we assume our equal (can be achieved by adding dummy variables or constraints if needed). We denote by ℓ the number of public inputs; and define $N := n + \ell$. We assume the first n variables correspond to the private inputs.

We assume we have a multiplicative subgroup $H \subset \mathbb{F}$ of size n. Somewhat confusingly, it will be convenient to identify the elements of H with the integers $\{0, 1, \ldots, n-1\}$ when using them as exponents.

Tweaking r1cs to our format Our original R1CS consists of the constraints:
For all $i \in [n]$

$$(a_i \cdot x)(b_i \cdot x) - (c_i \cdot x) = 0,$$

where $\cdot$ denotes inner product of vectors of length N. We modify the system to "flatten" linear combinations to variables. That is, we add two vectors of variables $y, z \in \mathbb{F}^n$ and look at the following system with $3n$ constraints:
For all $i \in [n]$

1. $y_i \cdot z_i - (c_i \cdot x) = 0$.
2. $y_i - (a_i \cdot x) = 0$.
3. $z_i - (b_i \cdot x) = 0$.

Note that above $y_i, z_i \in \mathbb{F}$ but $a_i, b_i, c_i, x \in \mathbb{F}^N$. Similarly to the original system, we call $x_n{+}1, ..., x_N$ the public variables of this system, and all other variables private variables.

A lemma about sums on subgroups We use the following fact mentioned in Remark 5.6 in [BCR+19] that is crucial to the Aurora system, as well as ours:

Lemma 4.1. *Fix any $f \in \mathbb{F}_{<n}[X]$. Then for any multiplicative subgroup $H \subset \mathbb{F}$ with $|H| = n$,*

$$\sum_{a \in H} f(a) = 0$$

if and only if f has a zero constant term.

4.1 The main protocol

When describing the protocol we assume we have a $2n$-polynomial commitment scheme as defined in Section 3.

Step 1: prover commiting to witness The prover **P** starts by computing three polynomials representing the satisfying assignment $(x, y, z) \in \mathbb{F}^{3n}$ of the private variables. Specificaly, polynomials $W, Y, Z \in \mathbb{F}_{<n}[X]$, such that for each $i \in H$, $W(i) = x_i$, $Y(i) = y_i$, and $Z(i) = z_i$. **P** sends polynomial commitments to W, Y, Z, with parameter $\mathbf{max} = n$, to the verifier **V**.

Step 2: verifier choosing a challenge; prover and verifier reducing to sumcheck **V** chooses random $r, r', r'' \in \mathbb{F}$ which it sends to **P**. They both now independently reduce the satisfiability check to a sum check as follows: Look at the sum

$$\sum_{i \in H} r^i (y_i z_i - c_i \cdot x) + \sum_{i \in H} r'^i (y_i - a_i \cdot x) + \sum_{i \in H} r''^i (z_i - b_i \cdot x).$$

Note that the sum is always zero for a satisfying assignment, and non-zero e.w.p $n/|\mathbb{F}|$ over r, r', r'' for a non-satisfying one. Rearrange the sum as

$$\sum_{i \in H} r^i y_i z_i + \sum_{i \in H} r'^i y_i + \sum_{i \in H} r''^i z_i + \sum_{i \in H} \alpha_i x_i + \alpha_0$$

where α_i is a coefficient containing some polynomial expression in r, r', r'', $\{a_{i,j}, b_{i,j}, c_{i,j}\}$; and also the public inputs $x_{n+1}, ..., x_N$ in the case of α_0. Now compute $R, R', R'', Q \in \mathbb{F}_{<n}[X]$, such that for $i \in [n], R(i) = r^i, R'(i) = r'^i, R''(i) = r''^i, Q(i) = \alpha_i$. Define the polynomial $D \in \mathbb{F}_{<3n}[X]$ by

$$D := R \cdot Y \cdot Z + R' \cdot Y + R'' \cdot Z + Q \cdot W + \alpha_0/n.$$

Our sum above becomes

$$\sum_{i \in H} D(i).$$

We have thus reduced our problem to a polynomial sumcheck. To be able to use Lemma 4.1, as in [BCR+19], we use polynomial division.

Let $Z_H(X) := \prod_{a \in H}(X - a) = X^n - 1$. **P** computes $g \in \mathbb{F}_{<2n}[X], f \in \mathbb{F}_{<n-1}[X]$ such that

$$D(X) = g(X) \cdot Z_H(X) + X \cdot f(X).$$

Note that from polynomial division combined with Lemma 4.1 if the sum vanishes - such f, g indeed exist. **P** sends commitments of g with parameter $\mathbf{max} = 2n$ and f with parameter $\mathbf{max} = n - 1$ to **V**.

Step 3: Verifier verifying the sumcheck by opening commitments and checking a polynomial identity Note that if g and f are well-formed in the sense that indeed $D(X) = g(X) \cdot Z_H(X) + X \cdot f(X)$, then $D(a) = a \cdot f(a)$ for any $a \in H$. In such a case it thus suffices that $\mathbf{V}$ check that

$$\sum_{a \in H} a \cdot f(a) = 0.$$

However, since $\deg(f) < n - 1$ is guaranteed by the commitment scheme, we have that $f'(X) := X \cdot f(X)$ is a polynomial of degree $< n$ with zero constant coefficient; therefore the sum vanishes by Lemma 4.1. Hence, $\mathbf{V}$ need only check that indeed $D = g \cdot Z_H + X \cdot f$.

For this purpose, $\mathbf{V}$ chooses a random $z \in \mathbb{F}$ and asks for the openings $Y(z), Z(z), W(z), g(z), f(z)$. Using $Y(z), Z(z), W(z)$ together with $R(z), R'(z), R''(z), Q(z), \alpha_0$ that it can compute by itself, $\mathbf{V}$ computes $D(z)$. Now $\mathbf{V}$ checks if

$$D(z) = g(z) \cdot Z_H(z) + z \cdot f(z),$$

and outputs acc if and only if the equality holds. Using Schwartz-Zippel, the probability of a false assignment leading to acceptance (via this identity or a bad choice of r, r', r'' above) is at most $4n/|\mathbb{F}|$.

4.2 Protocol summary

For convenience, we summarize the main protocol steps deferring to the more detailed description above for missing details.

1. $\mathbf{P}$ sends $\mathsf{cm}(W, n, \sigma), \mathsf{cm}(Y, n, \sigma), \mathsf{cm}(Z, n, \sigma)$ to $\mathbf{V}$.
2. $\mathbf{V}$ chooses random $r, r', r'' \in \mathbb{F}$ and sends them to $\mathbf{P}$. They both derive the polynomials R, R', R''.
3. $\mathbf{P}$ computes D. $\mathbf{P}$ computes $f \in \mathbb{F}_{<n-1}[X], g \in \mathbb{F}_{<2n}[X]$ such that $D(X) = g(X) \cdot Z_H(X) + X \cdot f(X)$.
4. $\mathbf{P}$ sends $\mathsf{cm}(g, 2n, \sigma), \mathsf{cm}(f, n-1, \sigma)$ to $\mathbf{V}$.
5. $\mathbf{V}$ chooses random $z \in \mathbb{F}$ and sends z to $\mathbf{P}$.
6. $\mathbf{P}$ sends $s_W = W(z), s_Y = Y(z), s_Z = Z(z), s_f = f(z), s_g = g(z)$ to $\mathbf{V}$.
7. $\mathbf{P}$ and $\mathbf{V}$ engage in the protocol $\mathsf{open}(5, \{n, n, n, n-1, 2n\}, \{W, Y, Z, f, g\}, \{s_W, s_Y, s_Z, s_f, s_g\})$. $\mathbf{V}$ outputs rej if the protocol verifier did.
8. $\mathbf{V}$ computes the alleged value of $D(z)$, as

   $$s_D := R(z)s_Y s_Z + R'(z)s_Y + R''(z)s_Z + Q(z)s_W + \alpha_0$$

   and outputs acc if and only if

   $$s_D = s_g \cdot Z_H(z) + z \cdot s_f.$$

Parallel proof generation Parallel proof generation with efficient amortized verification, as mentioned in Section 1.1, is achieved by running most steps separately for each public input, but jointly in steps 2 and 5, i.e. use the same verifier randomness r, r', r'', z for all proofs. When this is done the values $R(z), R'(z), R''(z), Q(z)$ can be computed just once for all proofs in step 8.

Saving one field element as in Sonic: Note that in the equation checked by $\mathbf{V}$ in step 8, when $z \neq 0$, the value s_f that will cause acceptance is uniquely determined by the other four values s_W, s_Y, s_Z, s_g. Thus $\mathbf{V}$ can compute it himself as

$$s_f := (s_D - s_g \cdot Z_H(z)) / z,$$

and check its correctness in the protocol open of step 7.

5 Batching arbitrary proofs with the Sonic helper

As in [MBKM19], the heavy/non-succinct part of the verifier computation is evaluating polynomials whose size is as large as the circuit/number of R1CS constraints. Sonic has a clever solution for this[2] - using an untrusted helper. The helper computes the polynomial evaluations for the verifier, and is able to prove the evaluations are correct. This is what [MBKM19] call a *signature of correct computation*. As we need to use the Sonic helper on several different polynomials in related evaluation points, it will be convenient to use a definition that supports multiple polynomials and evaluation points, at the cost of a more complicated definition of SCC.

Definition 5.1. *A protocol $\mathscr{P}$ between two-parties $(\mathsf{P_{sc}}, \mathsf{V_{sc}})$ is a $(t, \{m_i\}_{i\in[t]}, d)$-*signature of correct computation *($(t, \{m_i\}_{i\in[t]}, d)$-SCC) if, when both parties are given $s_1, \ldots, s_t \in \mathbb{F}_{<d}[X, Y]$ and a sequence of values $S = \left\{(x_{i,j}, y_{i,j}, s_{i,j}) \in \mathbb{F}^3\right\}_{i\in[t], j\in[m_i]}$, $\mathsf{V_{sc}}$ ends up outputting a value $\mathsf{res} \in \{\mathsf{acc}, \mathsf{rej}\}$ such that*

1. ***Completeness:** If for every $i \in [t], j \in [m_i]$, $s_{i,j} = s_i(x_{i,j}, y_{i,j})$, and $(\mathsf{P_{sc}}, \mathsf{V_{sc}})$ follow the protocol then $\mathsf{V_{sc}}$ outputs acc with probability one.*
2. ***Soundness:** For any e cient generic group adversary $\mathcal{A}$ playing the part of $\mathsf{P_{sc}}$, if for some $i \in [t], j \in [m_i]$, $s_{i,j} \neq s_i(x_{i,j}, y_{i,j})$, then the probability that $\mathsf{V_{sc}}$ outputs acc is $\mathsf{negl}(\lambda)$.*

The following lemma follows from [MBKM19]'s helped-SCC protocol, together with their batched opening commitment scheme as described in Section 3.

Lemma 5.2. *Fix integers $t, \{m_i\}_{i\in[t]}, d$. There is a pubic coin $(t, \{m_i\}_{i\in[t]}, d)$-SCC such that the following holds. Let $S = \left\{(x_{i,j}, y_{i,j}, s_{i,j}) \in \mathbb{F}^3\right\}_{i\in[t], j\in[m_i]}$ be the common input to $\mathsf{P_{sc}}$ and $\mathsf{V_{sc}}$ in the beginning of the protocol.*

Suppose that

1. *m^* is the sum over $i \in [t]$ of the number of distinct values in $\{y_{i,j}\}_{j\in[m_i]}$.*
2. *m^{**} is the number of distinct values in $\{y_{i,j}\}_{i\in[t], j\in[m_i]}$.*
3. *m^{***} is the number of distinct values in $\{x_{i,j}\}_{i\in[t], j\in[m_i]}$.*

[2] In fact, two different solutions for this and we'll discuss the second in Section 7.

Then

- *the total prover communication in the protocol consists of* $m^* + m^{**} + m^{***} + t$ $\mathbb{G}_1$ *elements and* $m^* + t$ *field elements.*
- *The verifier computation consists of one evaluation of each* s_i *and* $3(m^{**} + m^{***} + t)$ *pairings.*
- *The SRS required for the scheme is the same as that of the* d*-polynomial commitment scheme described in Section 3.*

Let us see how Lemma 5.2 helps us improve verifier efficiency in the batch helped model: The heavy verifier computations - meaning those linear instead of polylogarithmic in the number of constraints - are the evaluation at the point z of the polynomials Q, R, R', R''. Recall that R, R', R'' were defined by a random choice of $r \in \mathbb{F}$. We could in fact think of them all as *bi-variate* polynomials evaluated at (r, z) (R', R'' are the same bi-variate polynomial as R when viewed this way); we claim that as bivariate polynomials they are all members of $\mathbb{F}_{<n}[X, Y]$: Let $\{L_i \in \mathbb{F}[Y]\}_{i \in H}$ be the Lagrange basis with respect to H. That is, L_i is the unique polynomial of degree less than n with $L(i) = 1$ and $L(j) = 0$ for $i \neq j \in H$. Now, we can see that

$$R(X, Y) = \sum_{i \in H} X^i \cdot L_i(Y)$$

Similarly, Q was defined by a random choice of $r, r', r'' \in \mathbb{F}$ and thus can be thought of as a sum of three bivariates Q_1, Q_2, Q_3 with

$$Q_j(X, Y) = \sum_{i \in H} \alpha_{i,j}(X) \cdot L_i(Y)$$

for some $\alpha_{i,j}(X) \in \mathbb{F}_{<n}[X]$, such that the total number of non-zero coefficients in $\{\alpha_{i,j}(X)\}$ is bounded by the number of wires in the circuit we constructed the R1CS from; hence we can assume it is $O(n)$.

What will be important for us is that given $z \in \mathbb{F}$, the values $\{L_i(z)\}_{i \in H}$ can be computed in at most $O(n)$ field operations (As $L_i(Y) = \frac{Y^n - 1}{Y - i} \cdot \frac{i - 1}{i^n - 1}$).

Thus, a helper can use the protocol of Lemma 5.2, with to compute $Q(r, r', r'', z), R(r, z), R(r', z), R(r'', z)$ for $\mathbf{V}$ and convince him the values are correct.

Mapping our situation to Lemma 5.2, for one execution of the main protocol we have

1. $t = 4$: Our polynomials are Q_1, Q_2, Q_3, R.
2. $m^* = 4$: Our second coordinate is always z.
3. $m^{**} = 3$: The options for the first coordinate in all polynomials are r, r', r''.
4. $m^{***} = 1$: Again, cause our second coordinate is always z

On the other hand, when batching arbitrary proofs, there will not be necessarily any overlap between the values of the coordinates of evaluation points. It follow from Lemma 5.2 that, for a batch of T proofs, the helper mode adds $8T + 4$ $\mathbb{G}_1$ elements and $10T + 4$ field elements to the proofs (including a $6T$ factor for the evaluations of R, Q_1, Q_2, Q_3 themselves), and $12T + 12$ pairing operations to the verifier.

This while requiring the verifier to only perform $O(n)$ field operations to compute the polynomials Q_1, Q_2, Q_3, R at a single point, (instead of $O(n \cdot T)$ without the helper).

6 Getting zero-knowledge

We sketch how to add zero-knowledge to our scheme. The information given by the prover is limited to two evaluations of the polynomials W, Y, Z, f, g (one evaluation in the exponent during commitment, and one during opening). We need these evaluations not to leak any information. Note first that when the prover is honest, the evaluations of g are a function of the previous ones determined by the verifier equation. Thus, it is enough to show the evaluations of W, Y, Z, f don't leak information.

The natural method in such cases see e.g. [GGPR13, BCGV16, BCR+19], is to add to the polynomials random multiples of Z_H. This doesn't completely work in our case as f is the result of a modulu Z_H operation that would neutralize this randomization.[3] So, we must take a slightly more cumbersome "two-layered" randomization approach:

We assume the last two indices $x_{n-2}, x_{n-1}, y_{n-2}, y_{n-1}, z_{n-2}, z_{n-1}$ of x, y, z do not participate in any constraints (this can be achieved by padding the original R1CS with two unused variables). Let $H_0 := H \setminus \{n-2, n-1\}$.

The prover $\mathbf{P}$ will choose six random degree one polynomials $L_W^0, L_W^1, L_Y^0, L_Y^1, L_Z^0, L_Z^1$ and define

- $W' := W + L_W^0 Z_{H_0} + L_W^1 Z_H$.
- $Y' := Y + L_Y^0 Z_{H_0} + L_Y^1 Z_H$.
- $Z' := Z + L_Z^0 Z_{H_0} + L_Z^1 Z_H$.

$\mathbf{P}$ will conduct the protocol with W', Y', Z' instead of W, Y, Z. We claim this results in a zero-knowledge protocol.

Note first that W', Y', Z' satisfy the R1CS constraints when W, Y, Z did as they conicide on H_0.

Also, fixing any $\tau, z \in \mathbb{F}$; The twelve values $W'(\tau), W'(z), (W' \ mod \ Z_H)(\tau), (W' \ mod \ Z_H)(z)$, $Y'(\tau), Y'(z), (Y' \ mod \ Z_H)(\tau), (Y' \ mod \ Z_H)(z)$, $Z'(\tau), Z'(z), (Z' \ mod \ Z_H)(\tau), (Z' \ mod \ Z_H)(z)$ are all uniform and independent.

Thus, the two evaluations of W', Y', Z' give no information. The two evaluations of f are a function of the verifier randomness together with the evaluations at τ, z of $(W' \ mod \ Z_H), (Y' \ mod \ Z_H)$, $(Z' \ mod \ Z_H)$, which can thus be simulated by chosing the latter evaluations independently and uniformly and computing f from them.

7 Open question: getting a fully succinct veri er

[MBKM19] give a $(1, d)$-SCC where the run time of $\mathsf{V_{sc}}$ is polylogarithmic in d; i.e. the verifier is fully succinct, in the following restricted case. The polynomial $s(X, Y)$ can be written as a sum of a constant number of polynomials of the form

$$\sum_{i \in [d]} a_i X^i Y^{\sigma(i)}$$

where σ is a permutation of $[d]$. Our polynomials Q_1, Q_2, Q_3, R from Section 5 are not of this form. However, it seems plausible the construction can be made to work while changing the polynomials into this form.

[3] [BCR+19] has a very similar problem, that is more complex as their proof is a function, specifically the RS-IOPP part, of much more evaluations of f.

Acknowledgements

We thank Mary Maller for sharing details of the [MBKM19] construction, even while it was a work in progress, which inspired and motivated this work. We thank Sean Bowe for discussions on Sonic.

References

[BCC+16] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 327–357, 2016.

[BCG+17] E. Ben-Sasson, A. Chiesa, A. Gabizon, M. Riabzev, and N. Spooner. Interactive oracle proofs with constant rate and query complexity. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 40:1–40:15, 2017.

[BCGV16] E. Ben-Sasson, A. Chiesa, A. Gabizon, and M. Virza. Quasi-linear size zero knowledge from linear-algebraic pcps. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, pages 33–64, 2016.

[BCR+19] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, pages 103–128, 2019.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BS08] E. Ben-Sasson and M. Sudan. Short pcps with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

[GGPR13] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

[KZG10] 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, 2010.

[MBKM19] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. *IACR Cryptology ePrint Archive*, 2019:99, 2019.