Varuna Specification

Provable, Inc.

Last updated: July 16th, 2025

1 Introduction

Marlin-style proof systems are preprocessing zkSNARKs making use of holography. Some notable features of the construction include: 1. the structured reference string (SRS) is universal and updatable 2. it targets rank-1 constraint satisfiability (R1CS) 3. it has a modular proving pipeline with solid theoretical foundations. In the following, we introduce several improvements to the original Marlin construction [CHM+20]. For a high level introduction to Marlin, we recommend [Inb23].

In section 2 we show a glossary of notation. In section 3 we briefly compare our construction to previous work. In sections 4, 5 and 6 we describe the non-interactive protocol. Changes related to batch proving are marked in blue for clarity. We finish by showing condensed diagrams of the interactive protocol for batch proving. It is similar to the diagrams in [CHM+20] (Figure 5 in Section 5 and Figure 7 in Appendix E, in the ePrint version from 2021-10-04), but with some changes: it shows not just the information-theoretic Interactive Oracle Proof (IOP) but also the use of a polynomial commitment scheme (the cryptography layer); and it aims to be fully up-to-date with the recent optimizations in `https://github.com/ProvableHQ/snarkVM/tree/96854aa473a6419de61bc84df000653e7e488417`. This document can act as a "bridge" between the codebase and the theory that our references describe.

2 Quick Preliminaries

$\mathbb{F}$	the finite field over which the R1CS instance is defined
x	public input
$\hat{x}$	polynomial of degree less than $\|x\|$ that agrees with the instance x in $H[\leq \|x\|]$
w	secret witness
$\bar{w}$	shifted witness, defined as $\frac{w(X)-\hat{x}(X)}{v_{H[\leq \|x\|]}(X)}$
$\mathcal{D}$	the tuple of circuit frequencies ocurring in one batch proof (this is relevant in the context of multi-circuit batch proofs)
$\mathcal{M}$	the set of all R1CS matrices: $\{M_i\}_{M\in\{A,B,C\},i\in[\|\mathcal{D}\|]}$
R_i	constraint domain for R1CS matrices $\{A_i, B_i, C_i\}$ of a single circuit $i \in [\|\mathcal{D}\|]$ (throughout this text, by a *domain* we mean a subgroup of the group of units $\mathbb{F}^*$)
R	smallest subgroup of $\mathbb{F}^*$ containing all R_i (equivalently by the cyclicity of $\mathbb{F}^*$, the unique subgroup of $\mathbb{F}^*$ of order $\text{lcm}_i(\|C_i\|)$)
C_i	variable domain for R1CS matrices $\{A_i, B_i, C_i\}$ of a single circuit $i \in [\|\mathcal{D}\|]$
C	smallest subgroup of $\mathbb{F}^*$ containing all C_i defined analogously to R
A_i, B_i, C_i	R1CS instance matrices of the i-th circuit, which belong to $M_{\|R_i\|\times\|C_i\|}(\mathbb{F})$
K_M	domain for non-zero entries (but cf. footnote 1) of matrix M, approx. the number of addition gates
K	smallest subgroup of $\mathbb{F}^*$ containing all K_{M_i} for $M \in \{A, B, C\}, i \in [\|\mathcal{D}\|]$
X	domain sized for input (not including witness)
$v_D(X)$	vanishing polynomial over domain D, i. e. $x^{\|D\|} - 1$
$s_{D_1,D_2}(X)$	"selector" polynomial over domains $D_1 \supseteq D_2$, i. e. $\frac{\|D_2\|v_{D_1}}{\|D_1\|v_{D_2}}$ (note that $s_{D_1,D_2}(X)$ equals 1 for all $x \in D_2$ and 0 for all $x \in D_1 \setminus D_2$)
L_a^S	univariate Lagrange polynomial for the set $S \subseteq \mathbb{F}$ at $a \in S$ ($\deg(L_a^S) = \|S\| - 1$, $L_a^S(a) = 1$ and $L_a^S(b) = 0$ for all $b \in S \setminus \{a\}$)
$\mathsf{row}_M, \mathsf{col}_M$	LDEs of (respectively) row positions, column positions of matrix M^* over the domain K_M
rowcol_M	LDE over K_M of the element-wise product of row_M and col_M, given separately for efficiency
$\mathsf{rowcolval}_M$	LDE over K_M of the element-wise product of rowcol_M and the values of non-zero elements of matrix M^*
b	bound on the number of queries
$\mathcal{H}$	a stateful, duplex-sponge-based hash function using the Poseidon permutation (originally proposed in [COS20])
transcript	the concatenation of preprocessing inputs, the public input and proof elements written up to that point in time
d	the supported degree bound of the trusted setup
$\mathcal{P}$	prover
$\mathcal{V}$	verifier
$\mathcal{V}^p$	$\mathcal{V}$ with "oracle" access to polynomial p (via commitments provided by the indexer, later opened as necessary by $\mathcal{P}$)
row	the rowcheck
lin	the lineval sumcheck
mat	the matrix sumcheck
Q	the query set, consisting of the queries to $g_A, g_B, g_C, g_1, \mathbf{row}, \mathbf{lin}, \mathbf{mat}$

3 Comparing Varuna to Marlin

Varuna is a proving system based on Marlin. It adopts almost all of the improvements for the AHP noted in section 9.1 of the Marlin paper. Only the suggestion "Single low-degree extension for each matrix" is not implemented because although it reduces the proof size, it increases the prover time too much. Regarding the polynomial commitment scheme, Varuna

adopts all of the improvements noted in section 9.2 of the Marlin paper. We use a slightly adjusted version of the [KZG10]-like scheme with improvements introduced in [MBKM19], [Gab19], [CHM$^+$20] and [BCMS20]. More precisely, we use the equation described in [Gab19] section 3 that avoids negative $\mathbb{G}_1$ powers.

We also separate out the rowcheck to more easily support custom gates, add support for rectangular matrices, and replace the lincheck by a lineval - allowing us to evaluate $z_M = Mz$ without materializing z_M as an oracle [Gab20].

After an offline phase, the proof requires six rounds of interaction between a prover and verifier. Besides making the protocol more efficient for single proofs, we also introduce batch proving.

It is worth pointing out for clarity that snarkVM decomposes variables used in high-level-program operations into linear combinations before constraining them. This allows for reducing the number of actual variables in the system. Each linear combination corresponds to a partial or full row of an R1CS matrix, and consists of:

- constant coefficient and variable coefficients, representing the multipliers of the public constant 1 and the variables at 'compile time' (these define the structure of the circuit).
- value, representing the value of the assigned linear combination at 'proving time' (these determine the values the given circuit takes under the given variable assignment).

A full R1CS row is "completed" when we introduce an equality between the product of two linear combinations and a third linear combination: $a \cdot b = c$. A simple example using the public constant 1 and 3 variables which are assigned the value 2 at proving time:

- $a = \{\text{constant} : 1, \text{coefficients} : [var_1 : 1], \text{value} : 3\}$
- $b = \{\text{constant} : 2, \text{coefficients} : [var_2 : 1], \text{value} : 4\}$
- $c = \{\text{constant} : 6, \text{coefficients} : [var_3 : 3], \text{value} : 12\}$

Leads to a single row of R1CS:

- A = [1, 1, 0, 0]
- B = [2, 0, 1, 0]
- C = [6, 0, 0, 3]

4 Offline phase

The goal of the offline phase is to encode the R1CS linear relationships A, B and C as low degree polynomials. However, calculating $M(X, Y)$ naively by using X and Y as the row and column indices would cost $\Omega(|R| \cdot |C|)$ operations. To overcome this issue, we implement section 5.3.1 from the original Marlin paper, with three differences:

1. The indexer also outputs univariate polynomials for $\{\mathsf{rowcol}_M\}_{M \in \{A,B,C\}}$ and $\{\mathsf{rowcolval}_M\}_{M \in \{A,B,C\}}$, the latter of which is used instead of val. These achieve savings for the proving time in Round 4 below.
2. We adopt an optimization from [Gab20]. We do not encode the R1CS matrices using bivariate derivatives, but rather using Lagrange polynomials, which simplifies the online phase of the protocol.
3. The prover will initialize $A' := \begin{bmatrix} A & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$, $B' := \begin{bmatrix} B & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$, $C' := \begin{bmatrix} C & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. These random variables will ensure the scheme is zero-knowledge.

For each $M \in \mathcal{M}$, $\mathsf{row}_M(X)$, $\mathsf{col}_M(X)$, $\mathsf{rowcol}_M(X)$ and $\mathsf{rowcolval}_M(X)$ are obtained such that the following polynomials are low-degree extensions of M: $\hat{M}(X, Y) = \sum_{\kappa \in K} val_M(\kappa) \cdot L^R_{\mathsf{row}_M(\kappa)}(X) \cdot L^C_{\mathsf{col}_M(\kappa)}(Y)$. The polynomials encode the row index, column index and value of the **non-zero** entries of the matrix M[*1]. Constructing these polynomials for a circuit takes only $\Omega(|K| \log |K|)$ operations.

The preprocessing inputs which are added to the transcript consist of:

- the protocol name "VARUNA-2023"
- the instance batch sizes $\mathcal{D}$

[1] In order to be able to interpolate over subgroups K_M of $\mathbb{F}^*$, these polynomials may need to also interpolate entries of the matrix with the value zero. We abuse terminology and refer to all of these distinguished entries as "non-zero entries" whenever there is no risk of ambiguity.

- the circuit commitments from each circuit's verifying keys

We will also collect and prepare values from the universal setup for our proving and verification keys. In order to create commitments in the context of multi-circuit batch proofs, we take the union of the proving key's committer key.

5 Prover algorithm

In the following sections, for a single batch with circuit frequencies $\mathcal{D}$, we use the index $i \in [|\mathcal{D}|]$ to refer to a circuit and the index $j \in [\mathcal{D}_i]$ to refer to an instance of that circuit.

Round 1

The prover:

1. initializes $A' := \begin{bmatrix} A & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$, $B' := \begin{bmatrix} B & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$, $C' := \begin{bmatrix} C & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $z' := (x, w')$, and $w' := [w, \rho_A, \rho_B, \rho_C]$ where $\rho_A, \rho_B, \rho_C \xleftarrow{\$} \mathbb{F}$ satisfy $\rho_A \cdot \rho_B - \rho_C = 0$. These random variables ensure the scheme is zero-knowledge.
2. uses the full assignments $z_{i,j} = (x_{i,j}, w_{i,j})$ to compute the linear combinations $z_{A,i,j} := A_i z_{i,j}$, $z_{B,i,j} := B_i z_{i,j}$ and $z_{C,i,j} := C_i z_{i,j}$ and compute polynomials $\hat{z}_{M,i,j}(X) \in \mathbb{F}^{|z_{M,i,j}|+b}[X]$ which agrees with $\bar{z}_{M,i,j}$ on R_i and is random elsewhere.
3. **commits** to $\{[w_{i,j}]\}$, $[m]$:
   - $\hat{w}_{i,j}(X) \in \mathbb{F}^{|w_{i,j}|+b}[X]$ which agrees with $\bar{w}_{i,j}$ on $R_i[\geq |x_{i,j}|]$ and is random elsewhere
   - $m(X) \in \mathbb{F}^{<2|C|+2\mathtt{b}-2}[X]$ such that $\sum_{\kappa \in C} m(\kappa) = 0$. This is also needed to make the scheme zero knowledge. It is sufficient for the masking polynomial to be sampled with respect to the biggest C, as it serves to hide queries to g_1 and h_1.

The prover computes:

- batch combiners $\tau_{i,1} = 1$ and $\tau_{i,j} = \mathcal{H}(\textbf{transcript})[2+k]$ for $i \in [|\mathcal{D}|]$, $j \in [\mathcal{D}_i - 1]$ and $k = 0$ initially but incrementing by 1 for each invocation. These are used to bundle the sumcheck problems into $|\mathcal{D}|$ sumcheck problems.
- batch combiners $\nu_1 = 1$ and $\nu_i = \mathcal{H}(\textbf{transcript})[2 + \sum_i \mathcal{D}_i - 1 + k]$ for $k \in [|\mathcal{D}| - 1]$. These are used to bundle the previous $|\mathcal{D}|$ sumcheck problems into 1 sumcheck problem.

Round 2

Next, the prover will run a rowcheck in order to prove that, for each i and each $j \in |\mathcal{D}_i|$, one has $A_i z_{i,j} \odot B_i z_{i,j} = C_i z_{i,j}$. This is done through a BatchZeroCheck on the polynomials $\hat{z}_{A,i,j}(X) \cdot \hat{z}_{B,i,j}(X) - \hat{z}_{C,i,j}(X)$, each of which should vanish over all of R_i.

One can combine checks for different circuits and instances using the batch combiners ν_i and $\tau_{i,j}$, which additionally requires multiplying each i-th polynomial by the selector polynomial $s_{R,R_i}(X)$ so that all resulting individual ZeroChecks refer to the same domain R (without introducing new zeros). In other words, the prover needs to compute the quotient polynomial h_0 satisfying

$$\sum_i \nu_i s_{R,R_i}(X) \sum_j \tau_{i,j}(\hat{z}_{A,i,j}(X) \cdot \hat{z}_{B,i,j}(X) - \hat{z}_{C,i,j}(X)) = h_0(X) v_R(X).$$

The prover **commits** to $[h_0]$.

The prover computes $\alpha = \mathcal{H}(\textbf{transcript})[0]$, which will be used to reduce the lincheck problems to sumcheck problems.

Round 3

The rest of the protocol allows the prover to convince the verifier that each $\hat{z}_{M,i,j}$ is indeed an extension of the vector $M_i z_{i,j}$ over R_i (without revealing $w_{i,j}$). This amounts to $3 \cdot \sum_i \mathcal{D}_i$ lineval claims, each proved by testing $v_{R_i}(x)$ divides $\sum_{c \in C} \hat{M}_i(x, c)\hat{z}_{i,j}(c) - \hat{z}_{M,i,j}(x)$ at the random point α. The prover must hence provide $\sum_{c \in C} \hat{M}_i(\alpha, c)\hat{z}_{i,j}(c)$, which we denote by $\sigma_{M,i,j}$.

The prover **sends** $\sigma_{M,i,j}$ for each $M \in \{A, B, C\}$, i and j.

The prover computes the following batch combiners:

- $\eta_A = 1$, $\eta_B = \mathcal{H}(\textbf{transcript})[1]$ and $\eta_C = \mathcal{H}(\textbf{transcript})[2]$.
- $\mu_1 = 1$, $\mu_i = \mathcal{H}(\textbf{transcript})[3+k]$ for $2 \leq i \leq |\mathcal{D}|$, with k starting at 0 and incrementing by 1 with each invocation.
- For each i as above and each $j \in \{2, \ldots, \mathcal{D}_i\}$, $\rho_{i,1} = 1$ and $\rho_{i,j} = \mathcal{H}(\textbf{transcript})[2 + |\mathcal{D}| + k]$ with k starting at 0 and incrementing by 1 with each invocation *across all* i.

Round 4

The fact that each $\sigma_{M,i,j}$ was computed correctly can be proved with a univariate sumcheck over C_i. In order to batch all of them, one needs to sample batch combiners and multiply each sum by s_{C,C_i} to extend all sumcheck claims to the common domain C. The prover resorts to a single BatchUnivariateSumcheck stating that

$$\sum_{c \in C} m(c) + \sum_i \mu_i s_{C,C_i}(c) \sum_j \rho_{i,j} \sum_M \eta_M \hat{M}_i(\alpha, c) \hat{z}_{i,j}(c) = \sum_{M,i,j} \sigma_{M,i,j},$$

which in turn necessitates computing the unique $g_1(X) \in \mathbb{F}^{|C|-1}[X]$ and $h_1(X)$ of suitable degree satisfying

$$m(X) + \sum_i \mu_i s_{C,C_i}(X) \sum_j \rho_{i,j} \sum_{M_i} \eta_M \hat{M}_i(\alpha, X) \hat{z}_{i,j}(X) = h_1(X) v_C(X) + X g_1(X) + \sigma/|C| \qquad (**)$$

with $\sigma = \sum_{M,i,j} \sigma_{M,i,j}$.

The prover commits to **commits** to $[g_1(X)]$ and $[h_1(X)]$.

The prover computes the next challenge $\beta = \mathcal{H}(\textbf{transcript})[0]$ to evaluate the polynomial identity.

Round 5

What is left is for the prover to prove the value of $\hat{M}(\alpha, \beta)$, which we denote by σ_M. This can be reformulated as new rational sumchecks:

$$\textbf{for each } M \in \mathcal{M}, \ \textbf{RatSumcheck for } \frac{V_{R_M}(\alpha) V_{C_M}(\beta) \mathsf{rowcolval}_M(\kappa)}{|R_M||C_M|(\alpha - \mathsf{row}_M(\kappa))(\beta - \mathsf{col}_M(\kappa))} = \sigma_M \ \textbf{over} \ K_M$$

The prover computes and **commits** to $\{[g_M(X)]\}$ and sends $\{[\sigma_M]\}$ to the Verifier adhering to: $a_M(X) - b_M(X)(X \cdot g_M(X) + \sigma_M/|K_M|) = h_M(X) v_{K_M}(X) \qquad (***)$

The prover then computes randomizers $\delta_1 = 1$, $\delta_{i,j} = \mathcal{H}(\textbf{transcript})[k]$ for $i \in [|\mathcal{D}|]$, $j \in [\mathcal{D}_i - 1]$ and $k = 0$ initially but incrementing by 1 for each invocation.

Round 6

When doing a batch zero check (to conclude the rational sumchecks), we could multiply all terms $h_M(X) v_{K_M}(X) = a_M(X) - b_M(X)(X \cdot g_M(X) - \sigma_M/|K_M|))$ by an appropriate selector polynomial, and divide by $v_K(X)$. Instead, we perform the following optimization. Substituting in the selector we get that:

$$\frac{h_M(X) v_{K_M}(X) s_{K/K_M}}{v_K(X)} = h_M(X) \frac{|K_M|}{|K|}$$

The prover **commits** to $[h_2(X)]$:

let $h_2(X) := \sum_{M \in \mathcal{M}} \delta_M h_M(X) |K_M|/|K|$

Next, the prover computes a challenge $\gamma = \mathcal{H}(\textbf{transcript})[0]$.

Completing the proof

The prover uses a polynomial commitment scheme to prove that previous commitments $\{\mathsf{cm}_{g_M}\}$ and cm_{g_1} open to their evaluations at the challenges β, respectively γ, in order to enforce their degree bounds. However, the prover still needs to commit to various terms to prove the rowcheck, lineval sumcheck and matrix sumcheck problems. This can be done in an efficient way by constructing linear combinations, which we call virtual commitments (as outlined in section 9.2 of [CHM+20] and more formally introduced in [CBBZ23]). The prover proves that the following virtual commitments evaluate to 0 at the challenges α, β, γ:

- $\mathsf{vcm}_{rowcheck}$ is a linear combination of cm_{h_0} and sums $\{\sigma_{i,j,M}\}$.
- $\mathsf{vcm}_{\mathsf{lin}}$ is a linear combination of $\mathsf{cm}_m, \{\mathsf{cm}_{\hat{w}_{i,j}}\}, \mathsf{cm}_{h_1}$ using evaluations $g_1(\beta)$ and sums $\{\sigma_M\}_{M\in\mathcal{M}}$ and $\{\sigma_{i,j,M}\}_{M\in\{A,B,C\}}$ as factors.
- $\mathsf{vcm}_{\mathsf{mat}}$ is a linear combination of cm_{h_2} and for each $M \in \mathcal{M}$, index commitments to row_M, col_M, rowcol_M, $\mathsf{rowcolval}_M$, using evaluations $\{g_M(\gamma)\}$, and sums $\{\sigma_M\}$ as factors

The prover then runs **PC.Open** to construct a batch opening proof, using i.a. the (virtual) commitments, commitment randomness (to make the opening zero-knowledge), Q and $\mathcal{H}$. In the process, the prover computes a randomizer for each linear combination of (virtual) commitments checked in Q and a challenge for each element of those linear combinations.

6 Verifier algorithm

The AHP (i. e. computing challenges, randomizers and batch combiners)

The verifier:

1. adds the first round's commitments $[\hat{w}_{i,j}(X)]$ and $[m(X)]$ to the transcript and computes the batch combiners $\{\nu_i\}$ and $\{\tau_{i,j}\}$.
2. adds the second round's commitment $[h_0(X)]$ to the transcript and computes the challenge α.
3. adds the third round's values $\{\sigma_{M,i,j}\}$ to the transcript and computes the batch combiners $\eta_A, \eta_B, \eta_C, \{\mu_i\}$ and $\{\rho_{i,j}\}$.
4. adds the fourth round's commitments $[g_1(X)], [h_1(X)]$ to the transcript and computes the challenge β.
5. adds the fifth round's commitments $\{[g_M(X)]\}$ and claimed sums $\{\sigma_M\}$ to the transcript and computes the challenges $\{\delta_M\}$.
6. adds the sixth round's commitment $[h_2(X)]$ to the transcript and computes the challenge γ.

Completing the verification

After checking degree bounds on the verifying key, the verifier will verify the proof using a polynomial commitment scheme.

The verifier then runs **PC.Verify**, using i.a. the (virtual) commitments, commitment randomness, Q and $\mathcal{H}$. In the process, the verifier computes a randomizer for each linear combination of (virtual) commitments checked in Q and a challenge for each element of those linear combinations.

Compared to [Gab19], we use inverses to efficiently check the equality

$$e([-W]_1, [x]_2)e([-(v - W^z)]_1, [1]_2) \prod_{i\in[4]} e([\mathsf{cm}_i]_1, [x^{d_i-d}]_2) \stackrel{?}{=} 1,$$

where:

- cm_i is a linear combination of the (virtual) commitments which the verifier queries
- W is a linear combination of the proof's commitments to witness polynomials
- v is a linear combination of the random commitment polynomials at Q
- z is a linear combination of Q
- the degree bound d_i is:
  - d for $\mathsf{cm}_{mat}, \mathsf{cm}_{lin}, \mathsf{cm}_{row}$.
  - $|C| - 2$ for cm_{g_1}
  - $|K_M| - 2$ for cm_{g_M}

7 Analysis

The following was adapted from Marlin [CHM+20] section 5.3.3.

7.1 Soundness

For the single circuit case, we argue that the soundness error is at most: $\{\frac{2|R|}{|\mathbb{F}\backslash R|} + \frac{2|C|}{|\mathbb{F}\backslash C|} + \frac{3|K|}{\mathbb{F}}\}$.

Suppose that a given $\hat{w}$ does not contain an encoding of a valid witness w for a given index. We know that either $\hat{z}_A \cdot \hat{z}_B \neq \hat{z}_C$ or one of σ_M does not actually sum to the correct linear combination of $z \cdot M$.

In the first case, the polynomial identity $\hat{z}_A \cdot \hat{z}_B - \hat{z}_C = h_0(X)v_R(X)$ does not hold, so given Schwartz–Zippel the probability that the verifier still accepts is at most $\frac{2|R|}{|\mathbb{F}\backslash R|}$.

In the second case, we have to account for the soundness errors of the outer sumcheck and inner rational sumchecks, which are again bounded by the maximum degree in the respective polynomial equation divided by the size of the set from which the test point is chosen. The outer sumcheck has soundness error at most $\frac{2|C|}{|\mathbb{F}\backslash C|}$. The inner rational sumchecks have soundness error of at most $\frac{3|K|}{|F|}$.

Soundness for the multi-circuit sumcheck follows from creating random linear combinations of single-circuit sumchecks, as also noted in section 3.1 of [CBBZ23].

7.2 Zero-Knowledge

Just like the Marlin paper, we only sketch the intuition because a full proof (which includes constructing a simulator) is similar to the non-holographic setting described in [BCR+19].

The first message of the prover includes an encoding of the witness, which is protected against up to b queries outside of R because the encodings are b-wise independent over $\mathbb{F} \setminus R$. The second message includes the polynomial $h_0(X)$, which in fact is b-wise independent everywhere on $\mathbb{F}$, noting any witness-dependent polynomial requires a hiding bound. Subsequent commitments from the prover do not reveal any further information because they are produced for a sumcheck instance that is shifted by a random polynomial (the polynomial $m(X)$).

However, we still have to prevent the summed (randomly evaluated) assigned witness $\sigma_{i,j,M}$ from leaking information. We can be sure of this because during round 1 the prover added 3 random elements to the witness, one for each matrix M.

This leads to (perfect) zero knowledge with query bound b and a query checker C that rejects any query to $\hat{w}$ that lies in R.

7.3 Efficiency

Proof size

- Marlin: $13\mathbb{G}_1$+$8\mathbb{F}$
- Varuna: $9\mathbb{G}_1$+$10\mathbb{F}$
- Varuna batch i circuits and j instances: $(5 + \text{j} + 3\text{i})\mathbb{G}_1 + (1+9\text{i})\mathbb{F}$

Prover time Because the inner sumchecks are amortized across all instances, asymptotically, addition gates of additional instances become almost free, nearing the prover time of [Gro16].

Verifier time

- Marlin: 2 pairings + $\text{O}(|x| + \log m)$
- Varuna: 2 pairings + $\text{O}(|x| + \log m)$
- Varuna batch i circuits and j instances: 2 pairings + $O(\sum_{i,j}|x_{i,j}| + \log R_{max})$

8 Diagram including multi-circuit batching

$\mathcal{P}(\mathbb{F}, R, C, K, \{A_i\}, \{B_i\}, \{C_i\}, \{x_{i,j}\}, \{w_{i,j}\})$ $\qquad$ $\mathcal{V}^{\{\mathsf{row}_M, \mathsf{col}_M, \mathsf{rowcol}_M, \mathsf{rowcolval}_M\}_{M \in \{A_i, B_i, C_i\}}}(\mathbb{F}, R, C, K, \{x_{i,j}\})$

$z_{i,j} := (x_{i,j}, w_{i,j}), z_{M,i,j} := M_i z_{i,j}$
sample $\hat{w}_{i,j}(X) \in \mathbb{F}^{<|w_{i,j}|+\mathsf{b}}[X]$ and $\hat{z}_{M,i,j}(X) \in \mathbb{F}^{<|R_i|+\mathsf{b}}[X]$
sample mask polynomial $m(X) \in \mathbb{F}^{<2|C|+2\mathsf{b}-2}[X]$ such that $\sum_{\kappa \in C} m(\kappa) = 0$

commitments $\{\mathsf{cm}_{\hat{w},i,j}\}, \mathsf{cm}_m$ →
$\{\nu_i\}, \{\tau_{i,j}\} \leftarrow \mathbb{F}$
← $\{\nu_i\}, \{\tau_{i,j}\}$

BatchZeroCheck for $\sum_i \nu_i s_{R,R_i} \sum_j \tau_{i,j}(\hat{z}_{A,i,j} \cdot \hat{z}_{B,i,j} - \hat{z}_{C,i,j}) = 0$ **over** R

find $h_0(X)$ such that
$\sum_i \nu_i s_{R,R_i}(X) \sum_j \tau_{i,j}(\hat{z}_{A,i,j}(X) \cdot \hat{z}_{B,i,j}(X) - \hat{z}_{C.i,j}(X)) = h_0(X) v_R(X)$ $\quad (*)$

commitments cm_{h_0} →
$\alpha \leftarrow \mathbb{F} \setminus R$
← α
claimed sums $\{\sigma_{i,j,M} = \sum_{c \in C_i} \hat{M}_i(\alpha, c)\hat{z}_{i,j}(c)\}$ →
← $\eta_A, \eta_B, \eta_C, \{\mu_i\}, \{\rho_{i,j}\}$

BatchUniSumcheck for $m(X) + \sum_i \mu_i s_{C,C_i}(X) \sum_j \rho_{i,j} \sum_M \eta_M \hat{M}_i(\alpha, X)\hat{z}_{i,j}(X) - \sigma/|C| = 0$ **over** C

find $g_1(X) \in \mathbb{F}^{|C|-1}[X]$ and $h_1(X)$ such that
$m(X) + \sum_i \mu_i s_{C,C_i}(X) \sum_j \rho_{i,j} \sum_M \eta_M \hat{M}_i(\alpha, X)\hat{z}_{i,j}(X) = h_1(X) v_C(X) + X g_1(X) + \sigma/|C|$ $\quad (**)$

commitments $\mathsf{cm}_{g_1}, \mathsf{cm}_{h_1}$ →
$\beta \leftarrow \mathbb{F} \setminus C$
← $\beta \in \mathbb{F}$

for each $M \in \mathcal{M}$, **RatSumcheck for** $\frac{V_{R_M}(\alpha) V_{C_M}(\beta) \mathsf{rowcolval}_M(\kappa)}{|R_M||C_M|(\alpha - \mathsf{row}_M(\kappa))(\beta - \mathsf{col}_M(\kappa))} = \sigma_M$ **over** K_M

let $a_M(X) := v_{R_M}(\alpha) v_{C_M}(\beta) \mathsf{rowcolval}_M(X)$
let $b_M(X) := |R_M||C_M|(\beta - \mathsf{col}_M(X))(\alpha - \mathsf{row}_M(X))$
$= |R_M||C_M|(\alpha\beta - \alpha\mathsf{col}_M(X) - \beta\mathsf{row}_M(X) + \mathsf{rowcol}_M(X))$ (over K_M)

find $g_M(X), h_M(X) \in \mathbb{F}^{|K_M|-1}[X]$ and $\sigma_M \in \mathbb{F}$ such that
$a_M(X) - b_M(X)(X \cdot g_M(X) + \sigma_M/|K_M|) = h_M(X) v_{K_M}(X)$ $\quad (***)$

commitments $\{\mathsf{cm}_{g_M}\}$, and claimed sums $\{\sigma_M\}$ →
← $\{\delta_M\} \leftarrow \mathbb{F}$

let $h_2(X) := \sum_{M \in \mathcal{M}} \delta_M h_M(X)|K_M|/|K|$

commitment cm_{h_2} →
← $\gamma \leftarrow \mathbb{F}$

To verify $(***)$, $\mathcal{V}$ will need to check the following:

$$\underbrace{\sum_{M \in \mathcal{M}} \delta_M s_{K,K_M}(\gamma)(a_M(\gamma) - b_M(\gamma)(\gamma g_M(\gamma) + \sigma_M/|K_M|)) \; - \; h_2(\gamma) v_K(\gamma) \stackrel{?}{=} 0}_{\mathsf{mat}(\gamma)}$$

Compute $\hat{x}_{i,j}(X) \in \mathbb{F}^{<|x_{i,j}|}[X]$ from input $x_{i,j}$

To verify $(**)$, $\mathcal{V}$ will need to compute $\sigma = \sum_i \mu_i \sum_j \rho_{i,j} \sum_M \eta_M \sigma_{i,j,M}$ and check the following:

$$\underbrace{m(\beta) + \sum_{i \in [|\mathcal{D}|]} \mu_i s_{C,C_i}(\beta) \sum_{j \in [\mathcal{D}_i]} \rho_{i,j} \sum_{M \in A,B,C} \eta_M \sigma_{i,M}(\hat{x}_{i,j}(\beta) + v_{X_{i,j}}(\beta)\hat{w}_{i,j}(\beta)) - h_1(\beta) v_C(\beta) - \beta g_1(\beta) - \sigma/|C| \stackrel{?}{=} 0}_{\mathsf{lin}(\beta)}$$

To verify $(*)$, $\mathcal{V}$ will need to check the following:

$$\underbrace{\sum_i \nu_i s_{R,R_i}(\alpha) \sum_j \tau_{i,j}(\sigma_{i,j,A} \cdot \sigma_{i,j,B} - \sigma_{i,j,C}) - h_0(\alpha) v_R(\alpha) \stackrel{?}{=} 0}_{rowcheck(\alpha)}$$

$\{v_{g_M} := g_M(\gamma)\}, v_{g_1} := g_1(\beta)$

$\{v_{g_M}\}, v_{g_1}$ →
$\xi_1, \ldots, \xi_{|Q|} \leftarrow F$
← $\xi_1, \ldots, \xi_{|Q|}$

use PC.Open with randomness $\xi_1, \ldots, \xi_{|Q|}$ to
construct a batch opening proof π of the following:
$(\mathsf{cm}_{g_M}, \mathsf{vcm}_{\mathsf{mat}})$ at γ evaluate to $(v_{g_M}, 0)$ $\quad (***)$
$(\mathsf{cm}_{g_1}, \mathsf{vcm}_{\mathsf{lin}})$ at β evaluate to $(v_{g_1}, 0)$ $\quad (**)$
$(\mathsf{vcm}_{rowcheck})$ at α evaluates to (0) $\quad (*)$

π →

verify π with PC.Verify evaluations $\{v_{g_M}\}, v_{g_1}$ and
commitments $\{\mathsf{cm}_{g_M}\}, \mathsf{vcm}_{\mathsf{mat}}, \mathsf{cm}_{g_1}, \mathsf{vcm}_{\mathsf{lin}}, \mathsf{vcm}_{rowcheck}$

References

[BCMS20] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data from accumulation schemes. Cryptology ePrint Archive, Paper 2020/499, 2020. https://eprint.iacr.org/2020/499. URL: https://eprint.iacr.org/2020/499.

[BCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In *Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, EUROCRYPT '19, pages 103–128, 2019. URL: https://eprint.iacr.org/2018/828.

[CBBZ23] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. Hyperplonk: Plonk with linear-time prover and high-degree custom gates. In *Advances in Cryptology–EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23–27, 2023, Proceedings, Part II*, pages 499–530. Springer, 2023.

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*, pages 738–768. Springer, 2020.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*, pages 769–793. Springer, 2020.

[Gab19] Ariel Gabizon. Auroralight: Improved prover efficiency and srs size in a sonic-like system. Cryptology ePrint Archive, Paper 2019/601, 2019.

[Gab20] Ariel Gabizon. Lineval, 2020. URL: https://hackmd.io/aWXth2dASPaGVrXiGg1Cmg?view.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. 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 35*, pages 305–326. Springer, 2016.

[Inb23] Karthik Inbasekar. Marlin and me, 2023. URL: https://github.com/ingonyama-zk/papers/blob/main/Marlin_and_me.pdf.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian 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*, 2010.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. Cryptology ePrint Archive, Paper 2019/099, 2019. https://eprint.iacr.org/2019/099. URL: https://eprint.iacr.org/2019/099.