

# In-EVM Mina State Verification

## Technical Reference

Alisa Cherniaeva

[a.cherniaeva@nil.foundation](mailto:a.cherniaeva@nil.foundation)

=nil; Crypto3 (<https://crypto3.nil.foundation>)

Ilia Shirobokov

[i.shirobokov@nil.foundation](mailto:i.shirobokov@nil.foundation)

=nil; Crypto3 (<https://crypto3.nil.foundation>)

Mikhail Komarov

[nemo@nil.foundation](mailto:nemo@nil.foundation)

=nil; Foundation (<https://nil.foundation>)

January 5, 2022

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Overview . . . . .	2
<b>2</b>	<b>State Proof Generator</b>	<b>3</b>
2.1	Introduction . . . . .	3
2.2	Optimizations . . . . .	3
2.2.1	Batched FRI . . . . .	3
2.2.2	Hash By Column . . . . .	3
2.2.3	Hash By Subset . . . . .	4
2.3	RedShift Protocol . . . . .	4
2.3.1	Prover View . . . . .	4
2.3.2	Verifier View . . . . .	6
2.4	Introduction . . . . .	6
2.5	Preliminaries . . . . .	7
2.5.1	Pasta Curves . . . . .	7
2.5.2	Verification Algorithm . . . . .	7
2.6	Elliptic Curve Arithmetic . . . . .	10
2.6.1	Incomplete Addition . . . . .	10
2.6.2	Doubling and Tripling . . . . .	11
2.6.3	Unified Incomplete Addition and Squaring . . . . .	11
2.6.4	Variable Base Scalar Multiplication . . . . .	12
2.6.5	Variable Base Endo-Scalar Multiplication . . . . .	13
2.6.6	Fixed-base scalar multiplication circuit . . . . .	13
2.7	Multi-Scalar Multiplication Circuit . . . . .	14
2.7.1	Naive Algorithm . . . . .	14
2.7.2	Simultaneous Doubling . . . . .	15
2.8	Poseidon Circuit . . . . .	16
2.9	Other Circuits . . . . .	16
2.9.1	Combined Inner Product . . . . .	17
<b>3</b>	<b>In-EVM State Proof Verifier</b>	<b>18</b>
3.1	Verification Logic Architecture . . . . .	18
3.2	Verification Logic API Reference . . . . .	18
3.3	Input Data Structures . . . . .	18
	<b>Bibliography</b>	<b>18</b>

# Chapter 1

## Introduction

This document is a technical reference to the in-EVM Mina state verification project.

### 1.1 Overview

The project's purpose is to provide Ethereum users with reliable Mina Protocol's state proof. The project UX consists of several steps:

1. Retrieve Mina Protocol's state proof.
2. Preprocess it by generating an auxiliary proof.
3. Submit the preprocessed proof to EVM-enabled cluster.
4. Verify the proof with EVM.

Such a UX defines projects parts:

1. Mina Protocol's state retriever (O1Labs' or Chainsafe's protocol implementation).
2. State proof generator.
3. Ethereum RPC proof submitter.
4. EVM-based proof verifier.

The overall architecture diagram is as follows:  
Each of these parts will be considered independently.

## Chapter 2

# State Proof Generator

This introduces a description for Mina Protocol’s state auxiliary proof generator. Crucial components which define this part design and performance are:

1. Input data format (Pickles proof data structure: [2.5.2](#))
2. Proof system used for the proof generation.
3. Circuit definition used for the proof system.

## 2.1 Introduction

### WIP

To prove Mina blockchain’s state on the Ethereum Virtual Machine, we use Redshift SNARK[[1](#)]. RedShift is a transparent SNARK that uses PLONK[[2](#)] proof system but replaces the commitment scheme. The authors utilize FRI[[3](#)] protocol to obtain transparency for the PLONK system.

However, FRI cannot be straightforwardly used with the PLONK system. To achieve the required security level without huge overheads, the authors introduce *list polynomial commitment* scheme as a part of the protocol. For more details, we refer the reader to [[1](#)].

The original RedShift protocol utilizes the classic PLONK[[2](#)] system. To provide better performance, we generalize the original protocol for use with PLONK with custom gates [[4](#)], [[5](#)] and lookup arguments [[6](#)], [[7](#)].

## 2.2 Optimizations

### WIP

### 2.2.1 Batched FRI

Instead of checking each commitment individually, it is possible to aggregate them for FRI. For polynomials  $f_0, \dots, f_k$ :

1. Get  $\theta$  from transcript
2.  $f = f_0 \cdot \theta^{k-1} + \dots + f_k$
3. Run FRI over  $f$ , using oracles to  $f_0, \dots, f_k$

Thus, we can run only one FRI instance for all committed polynomials. See [[1](#)] for details.

### 2.2.2 Hash By Column

Instead of committing each of the polynomials, it is possible to use the same Merkle tree for several polynomials. This leads to the decrease of the number of Merkle tree paths which are required to be provided by the prover.

See [[8](#)], [[1](#)] for details.

### 2.2.3 Hash By Subset

Each  $i + 1$  FRI round supposes the prover to send all elements from a coset  $H \in D^{(i)}$ . Each Merkle leaf is able to contain the whole coset instead of separate values.

See [8] for details. Similar approach is described in [1]. However, the authors of [1] use more values per leaf, that leads to better performance.

## 2.3 RedShift Protocol

### WIP

Notations:

$N_{\text{wires}}$	Number of wires ('advice columns')
$N_{\text{perm}}$	Number of wires that are included in the permutation argument
$N_{\text{sel}}$	Number of selectors used in the circuit
$N_{\text{const}}$	Number of constant columns
$\mathbf{f}_i$	Witness polynomials, $0 \leq i < N_{\text{wires}}$
$\mathbf{f}_{c_i}$	Constant-related polynomials, $0 \leq i < N_{\text{const}}$
$\text{gate}_i$	Gate polynomials, $0 \leq i < N_{\text{sel}}$
$\sigma(\text{col} : i, \text{row} : j) = (\text{col} : i', \text{row} : j')$	Permutation over the table

For details on polynomial commitment scheme and polynomial evaluation scheme, we refer the reader to [1].

- 
1.  $\mathcal{L}' = (\mathbf{q}_0, \dots, \mathbf{q}_{N_{\text{sel}}})$
  2. Let  $\omega$  be a  $2^k$  root of unity
  3. Let  $\delta$  be a  $T$  root of unity, where  $T \cdot 2^S + 1 = p$  with  $T$  odd and  $k \leq S$
  4. Compute  $N_{\text{perm}}$  permutation polynomials  $S_{\sigma_i}(X)$  such that  $S_{\sigma_i}(\omega^j) = \delta^{i'} \cdot \omega^{j'}$
  5. Compute  $N_{\text{perm}}$  identity permutation polynomials:  $S_{id_i}(X)$  such that  $S_{id_i}(\omega^j) = \delta^i \cdot \omega^j$
  6. Let  $H = \{\omega^0, \dots, \omega^n\}$  be a cyclic subgroup of  $\mathbb{F}^*$
  7. Let  $Z(X) = \prod a \in H^*(X - a)$
- 

### Preprocessing:

#### 2.3.1 Prover View

1. Choose masking polynomials:

$$h_i(X) \leftarrow \mathbb{F}_{<k}[X] \text{ for } 0 \leq i < N_{\text{wires}}$$

**Remark:** For details on choice of  $k$ , we refer the reader to [1].

2. Define new witness polynomials:

$$f_i(X) = \mathbf{f}_i(X) + h_i(X)Z(X) \text{ for } 0 \leq i < N_{\text{wires}}$$

3. Add commitments to  $f_i$  to transcript
4. Get  $\beta, \gamma \in \mathbb{F}$  from  $\text{hash}(\text{transcript})$
5. For  $0 \leq i < N_{\text{perm}}$

$$\begin{aligned} p_i &= f_i + \beta \cdot S_{id_i} + \gamma \\ q_i &= f_i + \beta \cdot S_{\sigma_i} + \gamma \end{aligned}$$

6. Define:

$$\begin{aligned} p'(X) &= \prod_{0 \leq i < N_{\text{perm}}} p_i(X) \in \mathbb{F}_{<N_{\text{perm}} \cdot n}[X] \\ q'(X) &= \prod_{0 \leq i < N_{\text{perm}}} q_i(X) \in \mathbb{F}_{<N_{\text{perm}} \cdot n}[X] \end{aligned}$$

7. Compute  $P(X), Q(X) \in \mathbb{F}_{<n+1}[X]$ , such that:

$$\begin{aligned} P(\omega) &= Q(\omega) = 1 \\ P(\omega^i) &= \prod_{1 \leq j < i} p'(\omega^j) \text{ for } i \in 2, \dots, n+1 \\ Q(\omega^i) &= \prod_{1 \leq j < i} q'(\omega^j) \text{ for } i \in 2, \dots, n+1 \end{aligned}$$

8. Compute commitments to  $P, Q$  and add them to transcript.

9. Get  $\alpha_0, \dots, \alpha_5 \in \mathbb{F}$  from  $\text{hash}(\text{transcript})$

10. Get  $\tau$  from  $\text{hash}(\text{transcript})$

11. Define polynomials ( $F_0, \dots, F_4$  - copy-satisfiability,  $\mathbf{gate}_0$  is  $PI$ -constraining gate):

$$\begin{aligned} F_0(X) &= L_1(X)(P(X) - 1) \\ F_1(X) &= L_1(X)(Q(X) - 1) \\ F_2(X) &= P(X)p'(X) - P(X\omega) \\ F_3(X) &= Q(X)q'(X) - Q(X\omega) \\ F_4(X) &= L_n(X)(P(X\omega) - Q(X\omega)) \\ F_5(X) &= \sum_{0 \leq i < N_{\text{sel}}} (\tau^i \cdot \mathbf{q}_i(X) \cdot \mathbf{gate}_i(X)) + PI(X) \end{aligned}$$

12. Compute:

$$\begin{aligned} F(X) &= \sum_{i=0}^5 \alpha_i F_i(X) \\ T(X) &= \frac{F(X)}{Z(X)} \end{aligned}$$

13.  $N_T := \max(N_{\text{perm}}, \deg_{\text{gates}} - 1)$ , where  $\deg_{\text{gates}}$  is the highest degree of the degrees of gate polynomials.

14. Split  $T(X)$  into separate polynomials  $T_0(X), \dots, T_{N_T-1}(X)$ <sup>1</sup>

15. Add commitments to  $T_0(X), \dots, T_{N_T-1}(X)$  to transcript.

16. Get  $y \in \mathbb{F}/H$  from  $\text{hash}(\text{transcript})$

17. Run evaluation scheme with the committed polynomials and  $y$ .

**Remark:** Depending on the circuit, evaluation can be done also on  $y\omega, y\omega^{-1}$ .

18. The proof is  $\pi_{\text{comm}}$  and  $\pi_{\text{eval}}$ , where:

- $\pi_{\text{comm}} = \{f_{0,\text{comm}}, \dots, f_{N_{\text{wires}}-1,\text{comm}}, P_{\text{comm}}, Q_{\text{comm}}, T_{0,\text{comm}}, \dots, T_{N_T-1,\text{comm}}\}$
- $\pi_{\text{eval}}$  is evaluation proofs for  $f_0(y), \dots, f_{N_{\text{wires}}-1}(y), P(y), P(y\omega), Q(y), Q(y\omega), T_0(y), \dots, T_{N_T-1}(y)$

---

<sup>1</sup>Commit scheme supposes that polynomials should be degree  $\leq n$

### 2.3.2 Verifier View

1. Let  $f_{0,\text{comm}}, \dots, f_{N_{\text{wires}}-1,\text{comm}}$  be commitments to  $f_0(X), \dots, f_{N_{\text{wires}}-1}(X)$
2.  $\text{transcript} = \text{setup\_values} || f_{0,\text{comm}} || \dots || f_{N_{\text{wires}}-1,\text{comm}}$
3.  $\beta, \gamma = \text{hash}(\text{transcript})$
4. Let  $P_{\text{comm}}, Q_{\text{comm}}$  be commitments to  $P(X), Q(X)$
5.  $\text{transcript} = \text{transcript} || P_{\text{comm}} || Q_{\text{comm}}$
6.  $\alpha_0, \dots, \alpha_5 = \text{hash}(\text{transcript})$
7.  $\tau = \text{hash}(\text{transcript})$
8.  $N_T := \max(N_{\text{perm}}, \text{deg}_{\text{gates}} - 1)$ , where  $\text{deg}_{\text{gates}}$  is the highest degree of the degrees of gate polynomials.
9. Let  $T_{0,\text{comm}}, \dots, T_{N_T-1,\text{comm}}$  be commitments to  $T_0(X), \dots, T_{N_T-1}(X)$
10.  $\text{transcript} = \text{transcript} || T_{0,\text{comm}} || \dots || T_{N_T-1,\text{comm}}$
11.  $y = \text{hash}_{\mathbb{F}/H}(\text{transcript})$
12. Run evaluation scheme verification with the committed polynomials and  $y$  to check values  $f_i(y), P(y), P(y\omega), Q(y), Q(y\omega), T_j(y)$ .  
**Remark:** Depending on the circuit, evaluation can be done also on  $f_i(y\omega), f_i(y\omega^{-1})$  for some  $i$ .
13. Calculate:

$$\begin{aligned}
F_0(y) &= L_1(y)(P(y) - 1) \\
F_1(y) &= L_1(y)(Q(y) - 1) \\
p'(y) &= \prod p_i(y) = \prod f_i(y) + \beta \cdot S_{id_i}(y) + \gamma \\
F_2(y) &= P(y)p'(y) - P(y\omega) \\
q'(y) &= \prod q_i(y) = \prod f_i(y) + \beta \cdot S_{\sigma_i}(y) + \gamma \\
F_3(y) &= Q(y)q'(y) - Q(y\omega) \\
F_4(y) &= L_n(y)(P(y\omega) - Q(y\omega)) \\
F_5(y) &= \sum_{0 \leq i < N_{\text{sel}}} (\tau^i \cdot \mathbf{q}_i(y) \cdot \text{gate}_i(y)) + PI(y) \\
T(y) &= \sum_{0 \leq j < N_T} y^{n \cdot j} T_j(y)
\end{aligned}$$

14. Check the identity:

$$\sum_{i=0}^5 \alpha_i F_i(y) = Z(y)T(y)$$

## 2.4 Introduction

### WIP

High level description according to RfP<sup>2</sup>

1. Computing several hash values from the data of the proof. This involves using the Poseidon hash function with 55 full rounds both over  $\mathbb{F}_p$  and  $\mathbb{F}_q$  with round constants and MDS matrix specified for  $\mathbb{F}_p$ <sup>3</sup> and for  $\mathbb{F}_q$ <sup>4</sup>.
2. Checking arithmetic equations.
3. Performing one multi-scalar multiplication (MSM) of size  $2n_2 + 4 + (2 + 25) = 63$ , for which some of the bases are fixed and some are variable.
4. For each  $i \in \{1, 2\}$ , performing a multi-scalar multiplication over  $\mathbb{G}_i$  of size  $2^{n_i}$  with a fixed array of bases, and with scalars that can be very efficiently computed from the proof.

<sup>2</sup>[https://hackmd.io/u\\_2Ygx8XS5Ss1a0bgOFjkA](https://hackmd.io/u_2Ygx8XS5Ss1a0bgOFjkA)

<sup>3</sup><https://github.com/o1-labs/proof-systems/blob/master/oracle/src/pasta/fp.rs>

<sup>4</sup><https://github.com/o1-labs/proof-systems/blob/master/oracle/src/pasta/fq.rs>

Note that for MSM in Step 4:

$$\sum_{i=0}^{2^{n_k}-1} s_i \cdot G_i = H$$

$$s_i := \prod_{\substack{0 \leq j \leq n_k \\ \text{bits}(i)[j]=1}} \phi(c_j),$$

where:

- $\phi: \{0, 1\}^{128} \rightarrow \mathbb{F}$  is defined as `to_field` in the implementation<sup>5</sup>.
- Given an integer  $i < 2^{n_k}$ ,  $\text{bits}(i)$  is defined as the little-endian bit array of length  $n$  representing the binary expansion of  $i$ .
- $G_0, \dots, G_{2^{n_k}-1} \in \mathbb{G}_k$  is a fixed sequence of group elements<sup>6</sup>.
- $c_0, \dots, c_{n_k-1} \in \{0, 1\}^{128}$  is a sequence of challenges.

We use the same 15-wires PLONK circuits that are designed for Mina.<sup>7</sup>

## 2.5 Preliminaries

### WIP

#### 2.5.1 Pasta Curves

Let  $n_1 = 17$ ,  $n_2 = 16$ . Pasta curves parameters:

- $p = 2^{254} + 45560315531419706090280762371685220353$
- $q = 2^{254} + 45560315531506369815346746415080538113$
- Pallas:

$$\mathbb{G}_1 = \{(x, y) \in \mathbb{F}_p | y^2 = x^3 + 5\}$$

$$|\mathbb{G}_1| = q$$

- Vesta:

$$\mathbb{G}_2 = \{(x, y) \in \mathbb{F}_q | y^2 = x^3 + 5\}$$

$$|\mathbb{G}_2| = p$$

#### 2.5.2 Verification Algorithm

##### Notations

$N_{\text{wires}}$	Number of wires (‘advice columns’)
$N_{\text{perm}}$	Number of wires that are included in the permutation argument
$N_{\text{prev}}$	Number of previous challenges
$S_{\sigma_i}(X)$	Permutation polynomials for $0 \leq i < N_{\text{perm}}$
$\text{pub}(X)$	Public input polynomial
$w_i(X)$	Witness polynomials for $0 \leq i < N_{\text{wires}}$
$\eta_i(X)$	Previous challenges polynomials for $0 \leq i < N_{\text{prev}}$
$\omega$	$n$ -th root of unity

Denote multi-scalar multiplication  $\sum_{s_i \in \mathbf{s}, G_i \in \mathbf{G}} [s_i]G_i$  by  $\text{MSM}(\mathbf{s}, \mathbf{G})$  for  $l_{\mathbf{s}} = l_{\mathbf{G}}$  where  $l_{\mathbf{s}} = |\mathbf{s}|$ ,  $l_{\mathbf{G}} = |\mathbf{G}|$ . If  $l_{\mathbf{s}} < l_{\mathbf{G}}$ , then we use only first  $l_{\mathbf{s}}$  elements of  $\mathbf{G}$

<sup>5</sup><https://github.com/o1-labs/proof-systems/blob/49f81edc9c86e5907d26ea791fa083640ad0ef3e/oracle/src/sponge.rs#L33>

<sup>6</sup><https://github.com/o1-labs/proof-systems/blob/master/dlog/commitment/src/srs.rs#L70>

<sup>7</sup>[https://o1-labs.github.io/mina-book/specs/15\\_wires/15\\_wires.html](https://o1-labs.github.io/mina-book/specs/15_wires/15_wires.html)



**Proof**  $\pi$  contains (here  $\mathbb{F}_r$  is a scalar field of  $\mathbb{G}$ ):

- Commitments:
  - Witness polynomials:  $w_{0,\text{comm}}, \dots, w_{N_{\text{wires}},\text{comm}} \in \mathbb{G}$
  - Permutation polynomial:  $z_{\text{comm}} \in \mathbb{G}$
  - Quotient polynomial:  $t_{\text{comm}} = (t_{1,\text{comm}}, t_{2,\text{comm}}, \dots, t_{N_{\text{perm}},\text{comm}}) \in (\mathbb{G}^{N_{\text{perm}}} \times \mathbb{G})$
- Evaluations:
  - $w_0(\zeta), \dots, w_{N_{\text{wires}}}(\zeta) \in \mathbb{F}_r$
  - $w_0(\zeta\omega), \dots, w_{N_{\text{wires}}}(\zeta\omega) \in \mathbb{F}_r$
  - $z(\zeta), z(\zeta\omega) \in \mathbb{F}_r$
  - $S_{\sigma_0}(\zeta), \dots, S_{\sigma_{N_{\text{perm}}}}(\zeta) \in \mathbb{F}_r$
  - $S_{\sigma_0}(\zeta\omega), \dots, S_{\sigma_{N_{\text{perm}}}}(\zeta\omega) \in \mathbb{F}_r$
  - $\bar{L}(\zeta\omega) \in \mathbb{F}_r$ <sup>8</sup>
- Opening proof  $o_\pi$  for inner product argument:
  - $(L_i, R_i) \in \mathbb{G} \times \mathbb{G}$  for  $0 \leq i < \text{lr\_rounds}$
  - $\delta, \hat{G} \in \mathbb{G}$
  - $z_1, z_2 \in \mathbb{F}_r$
- previous challenges:
  - $\{\eta_i(\xi_j)\}_j, \eta_{i,\text{comm}}$ , for  $0 \leq i < \text{prev}$

**Remark:** For simplicity, we do not use distinct proofs index  $i$  for each element in the algorithm below. For instance, we write  $pub_{\text{comm}}$  instead of  $pub_{i,\text{comm}}$ .

---

<sup>8</sup>See [https://o1-labs.github.io/mina-book/crypto/plonk/maller\\_15.html](https://o1-labs.github.io/mina-book/crypto/plonk/maller_15.html)

---

**Algorithm 1** Verification

---

**Input:**  $\pi_0, \dots, \pi_{\text{batch\_size}}$  (see 2.5.2)**Output:** acc or rej

1. for each  $\pi_i$ :
    - 1.1  $\text{pub}_{\text{comm}} = \text{MSM}(\mathbf{L}, \text{pub}) \in \mathbb{G}$ , where  $\mathbf{L}$  is Lagrange bases vector
    - 1.2 **random\_oracle**( $p_{\text{comm}}, \pi_i$ ):
      - 1.2.1  $H_{\mathbb{F}_q}.\text{absorb}(\text{pub}_{\text{comm}} || w_{0,\text{comm}} || \dots || w_{N_{\text{wires}},\text{comm}})$
      - 1.2.2  $\beta, \gamma = H_{\mathbb{F}_q}.\text{squeeze}()$
      - 1.2.3  $H_{\mathbb{F}_q}.\text{absorb}(z_{\text{comm}})$
      - 1.2.4  $\alpha = \phi(H_{\mathbb{F}_q}.\text{squeeze}())$
      - 1.2.5  $H_{\mathbb{F}_q}.\text{absorb}(t_{1,\text{comm}} || \dots || t_{N_{\text{perm}},\text{comm}} || \dots || \infty ||)$
      - 1.2.6  $\zeta = \phi(H_{\mathbb{F}_q}.\text{squeeze}())$
      - 1.2.7 Transform  $H_{\mathbb{F}_q}$  to  $H_{\mathbb{F}_r}$
      - 1.2.8  $H_{\mathbb{F}_r}.\text{absorb}(\text{pub}(\zeta) || w_0(\zeta) || \dots || w_{N_{\text{wires}}}(\zeta) || S_0(\zeta) || \dots || S_{N_{\text{perm}}}(\zeta))$
      - 1.2.9  $H_{\mathbb{F}_r}.\text{absorb}(\text{pub}(\zeta\omega) || w_0(\zeta\omega) || \dots || w_{N_{\text{wires}}}(\zeta\omega) || S_0(\zeta\omega) || \dots || S_{N_{\text{perm}}}(\zeta\omega))$
      - 1.2.10  $H_{\mathbb{F}_r}.\text{absorb}(\bar{L}(\zeta\omega))$
      - 1.2.11  $v = \phi(H_{\mathbb{F}_r}.\text{squeeze}())$
      - 1.2.12  $u = \phi(H_{\mathbb{F}_r}.\text{squeeze}())$
      - 1.2.13 Compute evaluation of  $\eta_i(\zeta), \eta_i(\zeta\omega)$  for  $0 \leq i < N_{\text{prev}}$
      - 1.2.14 Compute evaluation of  $\bar{L}(\zeta)$
    - 1.3  $\mathbf{f}_{\text{base}} := \{S_{\sigma_{N_{\text{perm}}-1},\text{comm}}, \text{gate}_{\text{mult},\text{comm}}, w_{0,\text{comm}}, w_{1,\text{comm}}, w_{2,\text{comm}}, q_{\text{const},\text{comm}}, \text{gate}_{\text{psdn},\text{comm}}, \text{gate}_{\text{rc},\text{comm}}, \text{gate}_{\text{ec\_add},\text{comm}}, \text{gate}_{\text{ec\_dbl},\text{comm}}, \text{gate}_{\text{ec\_endo},\text{comm}}, \text{gate}_{\text{ec\_vbase},\text{comm}}\}$
    - 1.4  $s_{\text{perm}} := (w_0(\zeta) + \gamma + \beta \cdot S_{\sigma_0}(\zeta)) \cdot \dots \cdot (w_5(\zeta) + \gamma + \beta \cdot S_{\sigma_{N_{\text{perm}}}}(\zeta))$
    - 1.5  $\mathbf{f}_{\text{scalars}} := \{-z(\zeta\omega) \cdot \beta \cdot \alpha_0 \cdot zkp(\zeta) \cdot s_{\text{perm}}, w_0(\zeta) \cdot w_1(\zeta), w_0(\zeta), w_1(\zeta), 1, s_{\text{psdn}}, s_{\text{rc}}, s_{\text{ec\_add}}, s_{\text{ec\_dbl}}, s_{\text{ec\_endo}}, s_{\text{ec\_vbase}}\}$
    - 1.6  $f_{\text{comm}} = \text{MSM}(\mathbf{f}_{\text{base}}, \mathbf{f}_{\text{scalars}})$
    - 1.7  $\bar{L}_{\text{comm}} = f_{\text{comm}} - t_{\text{comm}} \cdot (\zeta^n - 1)$
    - 1.8 **PE** is a set of elements of the form  $(f_{\text{comm}}, f(\zeta), f(\zeta\omega))$  for the following polynomials:  
 $\eta_0, \dots, \eta_{N_{\text{prev}}}, \text{pub}, w_0, \dots, w_{N_{\text{wires}}}, z, S_{\sigma_0}, \dots, S_{\sigma_{N_{\text{perm}}}}, \bar{L}$
    - 1.9  $\mathcal{P}_i = \{H_{\mathbb{F}_q}, \zeta, v, u, \mathbf{PE}, o_{\pi_i}\}$
  2. **final\_check**( $\mathcal{P}_0, \dots, \mathcal{P}_{\text{batch\_size}}$ )
-

---

**Algorithm 2** Final Check

---

**Input:**  $\pi_0, \dots, \pi_{\text{batch\_size}}$ , where  $\pi_i = \{H_{i, \mathbb{F}_q}, \zeta_i, \zeta_i \omega, v_i, u_i, \mathbf{PE}_i, o_{\pi_i}\}$

**Output:** acc or rej

1.  $\rho_1 \rightarrow \mathbb{F}_r$
  2.  $\rho_2 \rightarrow \mathbb{F}_r$
  3.  $r_0 = r'_0 = 1$
  4. for  $0 \leq i < \text{batch\_size}$ :
    - 4.1  $\text{cip}_i = \text{combined\_inner\_product}(\zeta_i, \zeta_i \omega, v_i, u_i, \mathbf{PE}_i)$
    - 4.2  $H_{i, \mathbb{F}_q}.\text{absorb}(\text{cip}_i - 2^{255})$
    - 4.3  $U_i = (H_{i, \mathbb{F}_q}.\text{squeeze}()).\text{to\_group}()$
    - 4.4 Calculate opening challenges  $\xi_{i,j}$  from  $o_{\pi_i}$
    - 4.5  $h_i(X) := \prod_{k=0}^{\log(d+1)-1} (1 + \xi_{\log(d+1)-k} X^{2^k})$ , where  $d = \text{lr\_rounds}$
    - 4.6  $b_i = h_i(\zeta) + u_i \cdot h_i(\zeta \omega)$
    - 4.7  $C_i = \sum_j v_i^j (\sum_k r_i^k f_{j, \text{comm}})$ , where  $f_{j, \text{comm}}$  from  $\mathbf{PE}_i$ .
    - 4.8  $Q_i = \sum (\xi_{i,j} \cdot L_{i,j} + \xi_{i,j}^{-1} \cdot R_j) + \text{cip}_i \cdot U_i + C_i$
    - 4.9  $c_i = \phi(H_{i, \mathbb{F}_q}.\text{squeeze}())$
    - 4.10  $r_i = r_{i-1} \cdot \rho_1$
    - 4.11  $r'_i = r'_{i-1} \cdot \rho_2$
    - 4.12 Check  $\hat{G}_i = \langle s, G \rangle$ , where  $s$  is set of  $h(X)$  coefficients.  
**Remark:** This check can be done inside the MSM below using  $r'_i$ .
  5.  $\text{res} = \sum_i r^i (c_i Q_i + \text{delta}_i - (z_{i,1}(\hat{G}_i + b_i U_i) + z_{i,2} H))$
  6. return  $\text{res} == 0$
- 

---

**Algorithm 3** Combined Inner Product

---

**Input:**  $\xi, r, f_0(\zeta_1), \dots, f_k(\zeta_1), f_0(\zeta_2), \dots, f_k(\zeta_2)$

**Output:**  $s$

1.  $s = \sum_{i=0}^k \xi^i \cdot (f_i(\zeta_1) + r \cdot f_i(\zeta_2))$
- 

## 2.6 Elliptic Curve Arithmetic

WIP

### 2.6.1 Incomplete Addition

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$i$	$x_1$	$y_1$	$x_2$	$y_2$	$x_3$	$y_3$	$r$	...	...	...	...	...	...	...	...

Constraints:

- $(x_2 - x_1) \cdot (y_3 + y_1) - (y_1 - y_2) \cdot (x_1 - x_3) = 0$
- $(x_1 + x_2 + x_3) \cdot (x_1 - x_3) \cdot (x_1 - x_3) - (y_3 + y_1) \cdot (y_3 + y_1) = 0$
- $(x_2 - x_1) \cdot r = 1$

### 2.6.2 Doubling and Tripling

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$i$	$x_1$	$y_1$	$x_2$	$y_2$	$x_3$	$y_3$	$r_1$	$r_2$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

Constraints:

- Doubling:
  - $4 \cdot y_1^2 \cdot (x_2 + 2 \cdot x_1) = 9 \cdot x_1^4$
  - $2 \cdot y_1 \cdot (y_2 + y_1) = (3 \cdot x_1^2) \cdot (x_1 - x_2)$
  - $y_1 \cdot r_1 = 1$
- Addition (for tripling):
  - $(x_2 - x_1) \cdot (y_3 + y_1) - (y_1 - y_2) \cdot (x_1 - x_3) = 0$
  - $(x_1 + x_2 + x_3) \cdot (x_1 - x_3) \cdot (x_1 - x_3) - (y_3 + y_1) \cdot (y_3 + y_1) = 0$
  - $(x_2 - x_1) \cdot r_2 = 1$

### 2.6.3 Unified Incomplete Addition and Squaring

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$i$	$x_1$	$y_1$	$x_2$	$y_2$	$x_3$	$y_3$	<b>inf</b>	<b>same_x</b>	$s$	<b>inv<sub>y</sub></b>	<b>inv<sub>x</sub></b>	$\dots$	$\dots$	$\dots$	$\dots$

Constraints (**max degree** = 3):

1.  $w_7 \cdot (w_2 - w_0) = 0$
2.  $(w_2 - w_0) \cdot w_{10} - (1 - w_7) = 0$
3.  $w_7 \cdot (2w_8 \cdot w_1 - 3w_0^2) + (1 - w_7) \cdot (w_2 - w_0 \cdot w_8 - (w_3 - w_1))$
4.  $w_8^2 = w_0 + w_2 + w_4$
5.  $w_5 = w_8 \cdot (w_0 - w_4) - w_1$
6.  $(w_3 - w_1) \cdot (w_7 - w_6) = 0$
7.  $(w_3 - w_1) \cdot w_9 - w_6 = 0$

Copy constraints:

1.  $w_6 = 0$

**Details.** The gate uses basic group law formulae. Let  $P = (x_1, y_1)$ ,  $Q = (x_2, y_2)$ ,  $R = (x_3, y_3)$  and  $R = P + Q$ . Then:

- $(x_2 - x_1) \cdot s = y_2 - y_1$
- $s^2 = x_1 + x_2 + x_3$
- $y_3 = s \cdot (x_1 - x_3) - y_1$

For point doubling  $R = P + P = 2P$ :

- $2s \cdot y_1 = 3x_1^2$
- $s^2 = 2x_1 + x_3$
- $y_3 = s \cdot (x_1 - x_3) - y_1$

The gate does not handle cases  $\mathcal{O} + P$  or  $\mathcal{O} + \mathcal{O}$ . To ensure that operations with point-at-infinity are not included in the circuit's trace, copy constraint  $w_6 = 0$  (**inf** = 0) was introduced.

Constraints details:

- $x_2 - x_1$  zero check:
  1.  $w_7 \cdot (w_2 - w_0) = 0 \longleftrightarrow \text{same\_x} \cdot (x_2 - x_1)$   
If  $x_1 \neq x_2$ , then  $\text{same\_x} = 0$
  2.  $(w_2 - w_0) \cdot w_{10} - (1 - w_7) = 0 \longleftrightarrow (x_2 - x_1) \cdot \text{inv}_x - (1 - \text{same\_x})$   
If  $x_1 \neq x_2$ , then  $\text{inv}_x = (x_2 - x_1)^{-1}$
- Group law constraints:
  1.  $w_7 \cdot (2w_8 \cdot w_1 - 3w_0^2) + (1 - w_7) \cdot ((w_2 - w_0 \cdot w_8 - (w_3 - w_1)) \longleftrightarrow \text{same\_x} \cdot (2s \cdot y_1 - 3x_1^2) + (1 - \text{same\_x}) \cdot (x_2 - x_1 \cdot s - (y_2 - y_1)))$   
If  $x_1 = x_2$  then use doubling  $2s \cdot y_1 = 3x_1^2$ . Otherwise use addition  $(x_2 - x_1) \cdot s = y_2 - y_1$ .
  2.  $w_8^2 = w_0 + w_2 + w_4 \longleftrightarrow s^2 = x_1 + x_2 + x_3$   
Constrains  $x_3$ . It does not depend on  $x_1, x_2$  equality.
  3.  $w_5 = w_8 \cdot (w_0 - w_4) - w_1 \longleftrightarrow y_3 = s \cdot (x_1 - x_3) - y_1$   
Constrains  $y_3$ . It does not depend on  $x_1, x_2$  equality.
- $P + (-P)$  constraints:
  1.  $(w_3 - w_1) \cdot (w_7 - w_6) = 0 \longleftrightarrow (y_2 - y_1) \cdot (\text{same\_x} - \text{inf}) = 0$   
We can get infinity point iff  $x_1 = x_2$  and  $y_1 \neq y_2$ .  
If  $y_1 \neq y_2$  then  $\text{inf} = \text{same\_x}$ .
  2.  $(w_3 - w_1) \cdot w_9 - w_6 = 0 \longleftrightarrow (y_2 - y_1) \cdot \text{inv}_y - \text{inf}$   
The prover sets  $\text{inv}_y = 0$  for  $y_1 = y_2$ .  
If  $y_1 \neq y_2$  then  $\text{inv}_y = (y_2 - y_1)^{-1}$

## 2.6.4 Variable Base Scalar Multiplication

For  $S = [r]T$ , where  $r = 2^n + k$  and  $k = [k_n \dots k_0]$ ,  $k_i \in \{0, 1\}$ :<sup>9</sup>

1.  $S = [2]T$
2. for  $i$  from  $n - 1$  to 0:
  - 2.1  $Q = k_{i+1} ? T : -T$
  - 2.2  $R = S + Q$
  - 2.3  $S = R + S$
3.  $S = k_0 ? S - T : S$

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$i$	$x_T$	$y_T$	$x_S$	$y_S$	$x_P$	$y_P$	$n = 0$	$x_R$	$y_R$	$s_1$	$s_2$	$b_1$	$s_3$	$s_4$	$b_2$
$i + 1$	$s_5$	$b_3$	$x_S$	$y_S$	$x_P$	$y_P$	$n$	$x_R$	$y_R$	$x_V$	$y_V$	$s_1$	$b_1$	$s_3$	$b_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i + 100$	$x_T$	$y_T$	$x_S$	$y_S$	$x_P$	$y_P$	$n$	$x_R$	$y_R$	$s_1$	$s_2$	$b_1$	$s_3$	$s_4$	$b_2$
$i + 101$	$s_5$	$b_3$	$x_S$	$y_S$	$x_P$	$y_P$	$n$	$x_R$	$y_R$	$x_V$	$y_V$	$s_1$	$b_1$	$s_3$	$b_2$

Constraints for  $i + z$ , where  $z \bmod 2 = 0$ :

- $b_1 \cdot (b_1 - 1) = 0$
- $b_2 \cdot (b_2 - 1) = 0$
- $(x_P - x_T) \cdot s_1 = y_P - (2b_1 - 1) \cdot y_T$
- $s_1^2 - s_2^2 = x_T - x_R$
- $(2 \cdot x_P + x_T - s_1^2) \cdot (s_1 + s_2) = 2y_P$
- $(x_P - x_R) \cdot s_2 = y_R + y_P$
- $(x_R - x_T) \cdot s_3 = y_R - (2b_2 - 1) \cdot y_T$
- $s_3^2 - s_4^2 = x_T - x_S$

<sup>9</sup>Using the results from <https://arxiv.org/pdf/math/0208038.pdf>

- $(2 \cdot x_R + x_T - s_3^2) \cdot (s_3 + s_4) = 2 \cdot y_R$
- $(x_R - x_S) \cdot s_4 = y_S + y_R$
- $n = 32 \cdot \text{next}(n) + 16 \cdot b_1 + 8 \cdot b_2 + 4 \cdot \text{next}(b_1) + 2 \cdot \text{next}(b_2) + \text{next}(b_3)$

Constraints for  $i + z$ , where  $z \bmod 2 = 1$ :

- $b_1 \cdot (b_1 - 1) = 0$
- $b_2 \cdot (b_2 - 1) = 0$
- $b_3 \cdot (b_3 - 1) = 0$
- $(x_P - x_T) \cdot s_1 = y_P - (2b_1 - 1) \cdot y_T$
- $(2 \cdot x_P + x_T - s_1^2) \cdot ((x_P - x_R) \cdot s_1 + y_R + y_P) = (x_P - x_R) \cdot 2y_P$
- $(y_R + y_P)^2 = (x_P - x_R)^2 \cdot (s_1^2 - x_T + x_R)$
- $(x_T - x_R) \cdot s_3 = (2b_2 - 1) \cdot y_T - y_R$
- $(2x_R - s_3^2 + x_T) \cdot ((x_R - x_V) \cdot s_3 + y_V + y_R) = (x_R - x_V) \cdot 2y_R$
- $(y_V + y_R)^2 = (x_R - x_V)^2 \cdot (s_3^2 - x_T + x_V)$
- $(x_T - x_V) \cdot s_5 = (2b_3 - 1) \cdot y_T - y_V$
- $(2x_V - s_5^2 + x_T) \cdot ((x_V - x_S) \cdot s_5 + y_S + y_V) = (x_V - x_S) \cdot 2y_V$
- $(y_S + y_V)^2 = (x_V - x_S)^2 \cdot (s_5^2 - x_T + x_S)$

## 2.6.5 Variable Base Endo-Scalar Multiplication

For  $S = [r]T$ , where  $r = [r_n \dots r_0]$  and  $r_i \in \{0, 1\}$ :<sup>10</sup>

1.  $S = [2](\phi(T) + T)$
2. for  $i$  from  $\frac{\lambda}{2} - 1$  to 0:
  - 2.1  $Q = r_{2i+1} ? \phi([2r_{2i} - 1]T) : [2r_{2i} - 1]T$
  - 2.2  $R = S + Q$
  - 2.3  $S = R + S$

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$i$	$x_T$	$y_T$	$x_S$	$y_S$	$x_P$	$y_P$	$n$	$x_R$	$y_R$	$s_1$	$s_3$	$b_1$	$b_2$	$b_3$	$b_4$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i + 63$	$x_T$	$y_T$	$x_S$	$y_S$	$x_P$	$y_P$	$n$	$x_R$	$y_R$	$s_1$	$s_3$	$b_1$	$b_2$	$b_3$	$b_4$

Constraints:

- $b_1 \cdot (b_1 - 1) = 0$
- $b_2 \cdot (b_2 - 1) = 0$
- $b_3 \cdot (b_3 - 1) = 0$
- $b_4 \cdot (b_4 - 1) = 0$
- $((1 + (\text{endo} - 1) \cdot b_2) \cdot x_T - x_P) \cdot s_1 = (2 \cdot b_1 - 1) \cdot y_T - y_P$
- $(2 \cdot x_P - s_1^2 + (1 + (\text{endo} - 1) \cdot b_2) \cdot x_T) \cdot ((x_P - x_R) \cdot s_1 + y_R + y_P) = (x_P - x_R) \cdot 2 \cdot y_P$
- $(y_R + y_P)^2 = (x_P - x_R)^2 \cdot (s_1^2 - (1 + (\text{endo} - 1) \cdot b_2) \cdot x_T + x_R)$
- $((1 + (\text{endo} - 1) \cdot b_2) \cdot x_T - x_R) \cdot s_3 = (2 \cdot b_3 - 1) \cdot y_T - y_R$
- $(2 \cdot x_R - s_3^2 + (1 + (\text{endo} - 1) \cdot b_4) \cdot x_T) \cdot ((x_R - x_S) \cdot s_3 + y_S + y_R) = (x_R - x_S) \cdot 2 \cdot y_R$
- $(y_S + y_R)^2 = (x_R - x_S)^2 \cdot (s_3^2 - (1 + (\text{endo} - 1) \cdot b_4) \cdot x_T + x_S)$
- $n = 16 \cdot \text{next}(n) + 8 \cdot b_1 + 4 \cdot b_2 + 2 \cdot b_3 + b_4$

## 2.6.6 Fixed-base scalar multiplication circuit

We precompute all values  $w(B, s, k) = (k_i + 2) \cdot 8^s B$ , where  $k_i \in \{0, \dots, 7\}$ ,  $s \in \{0, \dots, 83\}$  and  $w(B, s, k) = (k_i \cdot 8^s - \sum_{j=0}^{84} 8^{j+1}) \cdot B$ , where  $k_i \in \{0, \dots, 7\}$ ,  $s = 84$ .

<sup>10</sup>Using the results from <https://eprint.iacr.org/2019/1021.pdf>

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$i$	$b_0$	$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$u_0$	$u_1$	$v_0$	$v_1$	$x_1$	$y_1$	$x_2$	$y_2$	$acc$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i + 42$	$b_0$	$b_1$	$b_2$	$u_0$	$v_0$	$x_w$	$y_w$	$\alpha$	$\beta$	$\gamma$	$\delta$	$\lambda$	$-$	$-$	$b$

Define the following functions:

1.  $\phi_1 : (x_1, x_2, x_3, x_4) \mapsto$   
 $x_3 \cdot (-u'_0 \cdot x_2 \cdot x_1 + u'_0 \cdot x_1 + u'_0 \cdot x_2 - u'_0 + u'_2 \cdot x_1 \cdot x_2 - u'_2 \cdot x_2 + u'_4 \cdot x_1 \cdot x_2 - u'_4 \cdot x_2 - u'_6 \cdot x_1 \cdot x_2 +$   
 $u'_1 \cdot x_2 \cdot x_1 - u'_1 \cdot x_1 - u'_1 \cdot x_2 + u'_1 - u'_3 \cdot x_1 \cdot x_2 + u'_3 \cdot x_2 - u'_5 \cdot x_1 \cdot x_2 + u'_5 \cdot x_2 + u'_7 \cdot x_1 \cdot x_2) - (x_4 -$   
 $u'_0 \cdot x_2 \cdot x_1 + u'_0 \cdot x_1 + u'_0 \cdot x_2 - u'_0 + u'_2 \cdot x_1 \cdot x_2 - u'_2 \cdot x_2 + u'_4 \cdot x_1 \cdot x_2 - u'_4 \cdot x_2 - u'_6 \cdot x_1 \cdot x_2)$
2.  $\phi_2 : (x_1, x_2, x_3, x_4) \mapsto$   
 $x_3 \cdot (-v'_0 \cdot x_2 \cdot x_1 + v'_0 \cdot x_1 + v'_0 \cdot x_2 - v'_0 + v'_2 \cdot x_1 \cdot x_2 - v'_2 \cdot x_2 + v'_4 \cdot x_1 \cdot x_2 - v'_4 \cdot x_2 - v'_6 \cdot x_1 \cdot x_2 + v'_1 \cdot$   
 $x_2 \cdot x_1 - v'_1 \cdot x_1 - v'_1 \cdot x_2 + v'_1 - v'_3 \cdot x_1 \cdot x_2 + v'_3 \cdot x_2 - v'_5 \cdot x_1 \cdot x_2 + v'_5 \cdot x_2 + v'_7 \cdot x_1 \cdot x_2) - (x_4 - v'_0 \cdot$   
 $x_2 \cdot x_1 + v'_0 \cdot x_1 + v'_0 \cdot x_2 - v'_0 + v'_2 \cdot x_1 \cdot x_2 - v'_2 \cdot x_2 + v'_4 \cdot x_1 \cdot x_2 - v'_4 \cdot x_2 - v'_6 \cdot x_1 \cdot x_2)$

Constraints:

- For  $i + 0$ :
  - $b_i \cdot (b_i - 1) = 0$ , where  $i \in \{0, \dots, 5\}$
  - $\phi_1(b_0, b_1, b_2, u_0) = 0$ , where  $(u'_i, v'_i) = w(B, 0, i)$
  - $\phi_1(b_3, b_4, b_5, u_1) = 0$ , where  $(u'_i, v'_i) = w(B, 1, i)$
  - $\phi_2(b_0, b_1, b_2, v_0) = 0$ , where  $(u'_i, v'_i) = w(B, 0, i)$
  - $\phi_2(b_3, b_4, b_5, v_1) = 0$ , where  $(u'_i, v'_i) = w(B, 1, i)$
  - $acc = b_0 + b_1 \cdot 2 + b_2 \cdot 2^2 + b_3 \cdot 2^3 + b_4 \cdot 2^4 + b_5 \cdot 2^5$
  - $(x_1, y_1) = (u_0, v_0)$
  - $(x_2, y_2) = (x_1, y_1) + (u_1, v_1)$  incomplete addition, where  $x_1 \neq u_1$
- For  $i + z$ ,  $z \in \{1, \dots, 41\}$ :
  - $b_i \cdot (b_i - 1) = 0$ , where  $i \in \{0, \dots, 5\}$
  - $\phi_1(b_0, b_1, b_2, u_0) = 0$ , where  $(u'_i, v'_i) = w(B, z \cdot 2, i)$
  - $\phi_1(b_3, b_4, b_5, u_1) = 0$ , where  $(u'_i, v'_i) = w(B, z \cdot 2 + 1, i)$
  - $\phi_2(b_0, b_1, b_2, v_0) = 0$ , where  $(u'_i, v'_i) = w(B, z \cdot 2, i)$
  - $\phi_2(b_3, b_4, b_5, v_1) = 0$ , where  $(u'_i, v'_i) = w(B, z \cdot 2 + 1, i)$
  - $acc = b_0 + b_1 \cdot 2 + b_2 \cdot 2^2 + b_3 \cdot 2^3 + b_4 \cdot 2^4 + b_5 \cdot 2^5 + acc_{prev} \cdot 2^6$
  - $(x_1, y_1) = (u_0, v_0) + (x_2, y_2)_{prev}$  incomplete addition, where  $u_0 \neq x_2$
  - $(x_2, y_2) = (x_1, y_1) + (u_1, v_1)$  incomplete addition, where  $x_1 \neq u_1$
- For  $i + 42$ :
  - $b_i \cdot (b_i - 1) = 0$ , where  $i \in \{0, \dots, 2\}$
  - $\phi_1(b_0, b_1, b_2, u_0) = 0$ , where  $(u'_i, v'_i) = w(B, 84, i)$
  - $\phi_2(b_0, b_1, b_2, v_0) = 0$ , where  $(u'_i, v'_i) = w(B, 84, i)$
  - $b = b_0 + b_1 \cdot 2 + b_2 \cdot 2^2 + acc_{prev} \cdot 2^3$
  - $(x_w, y_w) = (u_0, v_0) + (x_2, y_2)_{prev}$  complete addition from [Orchard](#)

## 2.7 Multi-Scalar Multiplication Circuit

WIP

Input:  $G_0, \dots, G_{k-1} \in \mathbb{G}, s_0, \dots, s_{k-1} \in \mathbb{F}_r$ , where  $\mathbb{F}_r$  is scalar field of  $\mathbb{G}$ .

Output:  $S = \sum_{i=0}^k s_i \cdot G_i$

### 2.7.1 Naive Algorithm

Using endomorphism:

1.  $A = \infty$

2. for  $j$  from 0 to  $k - 1$ :

2.1  $r := s_j, T := G_j$

2.2  $S = [2](\phi(T) + T)$

2.3 for  $i$  from  $\frac{\lambda}{2} - 1$  to 0:

2.3.1  $Q = r_{2i+1} ? \phi([2r_{2i} - 1]T) : [2r_{2i} - 1]T$

2.3.2  $R = S + Q$

2.3.3  $S = R + S$

2.4  $A = A + S$

$$\text{rows} \approx k \cdot (\text{sm\_rows} + 1 + 2) \approx 67k,$$

where **sm\_rows** is the number of rows in the scalar multiplication circuit.

**Without endomorphism:**

1.  $A = \infty$

2. for  $j$  from 0 to  $k - 1$ :

2.1  $r := s_j, T := G_j$

2.2  $S = [2]T$

2.3 for  $i$  from  $n - 1$  to 0:

2.3.1  $Q = k_{i+1} ? T : -T$

2.3.2  $R = S + Q$

2.3.3  $S = R + S$

2.4  $S = k_0 ? S - T : S$

2.5  $A = A + S$

$$\text{rows} \approx k \cdot (\text{sm\_rows} + 1 + 1) \approx 105k,$$

where **sm\_rows** is the number of rows in the scalar multiplication circuit.

## 2.7.2 Simultaneous Doubling

**Remark:** Simultaneous doubling incurs a negligible completeness error for independently chosen random terms of the sum.

**Using endomorphism:**

1.  $A = \sum_{j=0}^k [2](\phi(G_j) + G_j)$

2. for  $i$  from  $\frac{\lambda}{2} - 1$  to 0:

2.1 for  $j$  from 0 to  $k - 1$ :

2.1.1  $r := s_j, T := G_j$

2.1.2  $Q = r_{2i+1} ? \phi([2r_{2i} - 1]T) : [2r_{2i} - 1]T$

2.1.3  $A = A + Q$

2.2 if  $i \neq 0$ :

2.2.1  $A = 2 \cdot A$

$$\text{rows} \approx \frac{\lambda}{2} \cdot (k \cdot \text{add\_rows} + \text{dbl\_rows}) + 2k \approx 64 \cdot (k + 1) \approx 66k + 64,$$

where

- **add\_rows** is the number of rows in the addition circuit.
- **dbl\_rows** is the number of rows in the doubling circuit.



**Without endomorphism:**

1.  $A = \sum_{j=0}^k [2]G_j$
2. for  $i$  from  $n - 1$  to 0:
  - 2.1 for  $j$  from 0 to  $k - 1$ :
    - 2.1.1  $r := s_j, T := G_j$
    - 2.1.2  $Q = k_{i+1} ? T : -T$
    - 2.1.3  $A = A + Q$
  - 2.2 if  $i \neq 0$ :
    - 2.2.1  $A = 2 \cdot A$
3.  $A = A + \sum_{j=0}^k [1 - s_{j,0}]G_j$

$$\text{rows} \approx \frac{2}{3}n \cdot (k \cdot \text{add\_rows} + \text{dbl\_rows}) + k \approx 103 \cdot (k + 1) + 2k \approx 104k + 103,$$

where

- **add\_rows** is the number of rows in the addition circuit.
- **dbl\_rows** is the number of rows in the doubling circuit.

## 2.8 Poseidon Circuit

**WIP**

Mina uses Poseidon hash with width = 3. Therefore, each permutation state is represented by 3 elements and each row contains 5 states.

Denote  $i$ -th permutation state by  $T_i = (T_{i,0}, T_{i,1}, T_{i,2})$ .

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$i$	$T_{0,0}$	$T_{0,1}$	$T_{0,2}$	$T_{4,0}$	$T_{4,1}$	$T_{4,2}$	$T_{1,0}$	$T_{1,1}$	$T_{1,2}$	$T_{2,0}$	$T_{2,1}$	$T_{2,2}$	$T_{3,0}$	$T_{3,1}$	$T_{3,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i + 10$	$T_{50,0}$	$T_{50,1}$	$T_{50,2}$	$T_{54,0}$	$T_{54,1}$	$T_{54,2}$	$T_{51,0}$	$T_{51,1}$	$T_{51,2}$	$T_{52,0}$	$T_{52,1}$	$T_{52,2}$	$T_{53,0}$	$T_{53,1}$	$T_{53,2}$
$i + 11$	$T_{55,0}$	$T_{55,1}$	$T_{55,2}$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

State change constraints:

$$\text{STATE}(i + 1) = \text{STATE}(i)^\alpha \cdot \text{MDS} + \text{RC}$$

Denote the index of the first state in the row by **start** (e.g. **start** = 50 for 10-th row). We can expand the previous formula to:

- For  $i$  from **start** to **start** + 5:
  - $T_{i+1,0} = T_{i,0}^5 \cdot \text{MDS}[0][0] + T_{i,1}^5 \cdot \text{MDS}[0][1] + T_{i,2}^5 \cdot \text{MDS}[0][2] + \text{RC}_{i+1,0}$
  - $T_{i+1,1} = T_{i,0}^5 \cdot \text{MDS}[1][0] + T_{i,1}^5 \cdot \text{MDS}[1][1] + T_{i,2}^5 \cdot \text{MDS}[1][2] + \text{RC}_{i+1,1}$
  - $T_{i+1,2} = T_{i,0}^5 \cdot \text{MDS}[2][0] + T_{i,1}^5 \cdot \text{MDS}[2][1] + T_{i,2}^5 \cdot \text{MDS}[2][2] + \text{RC}_{i+1,2}$

Notice that the constraints above include the state from the next row (**start** + 5).

## 2.9 Other Circuits

**WIP**

### 2.9.1 Combined Inner Product

$$\sum_{i=0}^k \xi^i \cdot (f_i(\zeta_1) + r \cdot f_i(\zeta_2))$$

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$i + 0$	$f_1$	$f'_1$	$f_2$	$f'_2$	<b>acc</b>	$\xi$	$\xi_{\text{acc}}$	$s_1$	$s_2$	$\xi'_{\text{acc}}$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$i + 1$	$f_3$	$f'_3$	$f_4$	$f'_4$	<b>acc</b>	$r$	$\xi_{\text{acc}}$	$s_1$	$s_2$	$\xi'_{\text{acc}}$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i + \lceil \frac{k}{2} \rceil - 1$	$f_{k-3}$	$f'_{k-3}$	$f_{k-2}$	$f'_{k-2}$	<b>acc</b>	$\xi$	$\xi_{\text{acc}}$	$s_1$	$s_2$	$\xi'_{\text{acc}}$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$i + \lceil \frac{k}{2} \rceil$	$f_{k-1}$	$f'_{k-1}$	$f_k$	$f'_k$	<b>acc</b>	$r$	$\xi_{\text{acc}}$	$s_1$	$s_2$	$\xi'_{\text{acc}}$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

Constraints for  $i + z$ , where  $z \bmod 2 = 0$ :

- $(w_0 + w_1 \cdot \text{next}(w_5)) \cdot w_6 = w_7$
- $(w_2 + w_3 \cdot \text{next}(w_5)) \cdot w_9 = w_8$
- $w_5 \cdot w_6 = w_9$
- $w_5 \cdot w_9 = \text{next}(w_9)$
- $w_5 \cdot \text{next}(w_9) = \text{next}(w_5)$
- $w_4 + w_7 + w_8 + \text{next}(w_7) + \text{next}(w_8) = \text{next}(w_4)$

Constraints for  $i + z$ , where  $z \bmod 2 = 1$ :

- $(w_0 + w_1 \cdot w_5) \cdot w_9 = w_7$
- $(w_2 + w_3 \cdot w_5) \cdot w_6 = w_8$

## Chapter 3

# In-EVM State Proof Verifier

This introduces a description for in-EVM Mina Protocol state proof verification mechanism. Crucial components which define this part design are:

1. Verification architecture description.
2. Verification logic API reference.
3. Input data structures description.

### 3.1 Verification Logic Architecture

### 3.2 Verification Logic API Reference

### 3.3 Input Data Structures

# Bibliography

1. Kattis A., Panarin K., Vlasov A. RedShift: Transparent SNARKs from List Polynomial Commitment IOPs. Cryptology ePrint Archive, Report 2019/1400. 2019. <https://ia.cr/2019/1400>.
2. Gabizon A., Williamson Z. J., Ciobotaru O. PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge. Cryptology ePrint Archive, Report 2019/953. 2019. <https://ia.cr/2019/953>.
3. Fast Reed-Solomon interactive oracle proofs of proximity / E. Ben-Sasson, I. Bentov, Y. Horesh et al. // 45th international colloquium on automata, languages, and programming (icalp 2018) / Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2018.
4. Gabizon A., Williamson Z. J. Proposal: The Turbo-PLONK program syntax for specifying SNARK programs. [https://docs.zkproof.org/pages/standards/accepted-workshop3/proposal-turbo\\_plonk.pdf](https://docs.zkproof.org/pages/standards/accepted-workshop3/proposal-turbo_plonk.pdf).
5. PLONKish Arithmetization - The halo2 book. <https://zcash.github.io/halo2/concepts/arithmetization.html>.
6. Gabizon A., Williamson Z. J. plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Report 2020/315. 2020. <https://ia.cr/2020/315>.
7. Lookup argument - The halo2 book. <https://zcash.github.io/halo2/design/proving-system/lookup.html>.
8. Chiesa A., Ojha D., Spooner N. Fractal: Post-Quantum and Transparent Recursive Proofs from Holography. Cryptology ePrint Archive, Report 2019/1076. 2019. <https://ia.cr/2019/1076>.