

# 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>)

May 10, 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	Mina Verification Algorithm . . . . .	3
2.2.1	Pasta Curves . . . . .	3
2.2.2	Verification Algorithm . . . . .	4
2.3	Elliptic Curve Arithmetic . . . . .	6
2.3.1	Unified Incomplete Addition and Doubling . . . . .	6
2.3.2	Variable Base Scalar Multiplication . . . . .	8
2.3.3	Variable Base Endo-Scalar Multiplication . . . . .	9
2.4	Multi-Scalar Multiplication Circuit . . . . .	10
2.5	Poseidon Circuit . . . . .	11
2.6	Other Circuits . . . . .	11
2.6.1	Endo-Scalar Computation . . . . .	11
2.7	Proof Verification Component . . . . .	12
<b>3</b>	<b>In-EVM State Proof Verifier</b>	<b>17</b>
3.1	Verification Logic Architecture . . . . .	17
3.2	Verification Logic API Reference . . . . .	17
3.3	Input Data Structures . . . . .	17
<b>4</b>	<b>Appendix A. In-EVM Mina State</b>	<b>18</b>
4.1	Overview . . . . .	18
4.1.1	Purpose . . . . .	19
	<b>Bibliography</b>	<b>20</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 (O(1) Labs' 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: [2.2.2](#))
2. Proof system used for the proof generation.
3. Circuit definition used for the proof system.

### 2.1 Introduction

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].<sup>1</sup>

### 2.2 Mina Verification Algorithm

WIP

#### 2.2.1 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:

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

---

<sup>1</sup>Details on the proof system: [https://github.com/NilFoundation/evm-mina-verification/tree/master/docs/proof\\_system](https://github.com/NilFoundation/evm-mina-verification/tree/master/docs/proof_system)

## 2.2.2 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  $\mathbf{G}$

**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>2</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  $\text{pub}_{\text{comm}}$  instead of  $\text{pub}_{i,\text{comm}}$ .

<sup>2</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 2.2.2)**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.  $s = \sum_{i=0}^k \xi^i \cdot (f_i(\zeta_1) + r \cdot f_i(\zeta_2))$
- 

We use the same 15-wires PLONK circuits that are designed for Mina.<sup>3</sup>

## 2.3 Elliptic Curve Arithmetic

### 2.3.1 Unified Incomplete Addition and Doubling

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$i$	$x_1$	$y_1$	$x_2$	$y_2$	$x_3$	$y_3$	inf	same_x	$s$	inv $_y$	inv $_x$	...	...	...	...

Evaluations:

- Addition case:

---

<sup>3</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)

- $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$
- $\mathbf{inf} = 1$  if  $(x_3, y_3)$  is a point-at-infinity,  $\mathbf{inf} = 0$  otherwise
- $\mathbf{same\_x} = 1$  if  $x_1 = x_2$ ,  $\mathbf{same\_x} = 0$  otherwise
- $s = \frac{y_1 - y_2}{x_1 - x_2}$  if  $x_1 \neq x_2$ ,  $s = 0$  otherwise
- $\mathbf{inv}_y = \frac{1}{y_2 - y_1}$  if  $y_2 \neq y_1$ ,  $\mathbf{inv}_y = 0$  otherwise
- $\mathbf{inv}_x = \frac{1}{x_2 - x_1}$  if  $x_2 \neq x_1$ ,  $\mathbf{inv}_x = 0$  otherwise
- Doubling case:
  - $(x_3, y_3) = 2(x_1, y_1)$
  - $x_2 = x_1, y_2 = y_1$
  - $\mathbf{inf} = 1$  if  $(x_3, y_3)$  is a point-at-infinity,  $\mathbf{inf} = 0$  otherwise
  - $\mathbf{same\_x} = 1$
  - $s = \frac{3x_1^2}{2y_1}$  if  $y_1 \neq 0$ ,  $s = 0$  otherwise
  - $\mathbf{inv}_y = 0$
  - $\mathbf{inv}_x = 0$

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  $w_6 = 0$  ( $\mathbf{inf} = 0$ ) was introduced.

Constraints details:

- $x_2 - x_1$  zero check:
  1.  $w_7 \cdot (w_2 - w_0) = 0 \longleftrightarrow \mathbf{same\_x} \cdot (x_2 - x_1)$   
If  $x_1 \neq x_2$ , then  $\mathbf{same\_x} = 0$
  2.  $(w_2 - w_0) \cdot w_{10} - (1 - w_7) = 0 \longleftrightarrow (x_2 - x_1) \cdot \mathbf{inv}_x - (1 - \mathbf{same\_x})$   
If  $x_1 \neq x_2$ , then  $\mathbf{inv}_x = (x_2 - x_1)^{-1}$
- Group law constraints:
  1.  $w_7 \cdot (2w_8 \cdot w_1 - 3w_0^2) + (1 - w_7) \cdot ((w_2 - w_0) \cdot w_8 - (w_3 - w_1)) \longleftrightarrow \mathbf{same\_x} \cdot (2s \cdot y_1 - 3x_1^2) + (1 - \mathbf{same\_x}) \cdot (x_2 - x_1 \cdot s - (y_2 - y_1))$   
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  $\text{inv}_y = (y_2 - y_1)^{-1}$

### 2.3.2 Variable Base Scalar Multiplication

For  $R = [r]T$ ,  $k = \frac{x-2^{255}}{2}$ : <sup>4</sup>

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

The first and last steps of the algorithm are verified by the unified addition and doubling circuit.

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$i$	$x_T$	$y_T$	$x_0$	$y_0$	$n = 0$	$n'$	—	$x_1$	$y_1$	$x_2$	$y_2$	$x_3$	$y_3$	$x_4$	$y_4$
$i + 1$	$x_5$	$y_5$	$b_0$	$b_1$	$b_2$	$b_3$	$b_4$	$s_0$	$s_1$	$s_2$	$s_3$	$s_4$	—	—	—
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i + 100$	$x_T$	$y_T$	$x_0$	$y_0$	$n$	$n'$	—	$x_1$	$y_1$	$x_2$	$y_2$	$x_3$	$y_3$	$x_4$	$y_4$
$i + 101$	$x_5$	$y_5$	$b_0$	$b_1$	$b_2$	$b_3$	$b_4$	$s_0$	$s_1$	$s_2$	$s_3$	$s_4$	—	—	—

Evaluations:

- $b_i$  are bits of the  $k$ , first  $b_1$  is the most significant bit of  $k$ ,  $n$  is an accumulator of  $b_i$ .
- $(x_1, y_1) - (x_0, y_0) = (x_0, y_0) + (x_T, (2b_1 - 1)y_T)$
- $(x_2, y_2) - (x_1, y_1) = (x_1, y_1) + (x_T, (2b_1 - 1)y_T)$
- $(x_3, y_3) - (x_2, y_2) = (x_2, y_2) + (x_T, (2b_1 - 1)y_T)$
- $(x_4, y_4) - (x_3, y_3) = (x_3, y_3) + (x_T, (2b_1 - 1)y_T)$
- $(x_5, y_5) - (x_4, y_4) = (x_4, y_4) + (x_T, (2b_1 - 1)y_T)$
- $s_0 = \frac{y_0 - (2b_0 - 1) \cdot y_T}{x_0 - x_T}$
- $s_1 = \frac{y_1 - (2b_1 - 1) \cdot y_T}{x_1 - x_T}$
- $s_2 = \frac{y_2 - (2b_2 - 1) \cdot y_T}{x_2 - x_T}$
- $s_3 = \frac{y_3 - (2b_3 - 1) \cdot y_T}{x_3 - x_T}$
- $s_4 = \frac{y_4 - (2b_4 - 1) \cdot y_T}{x_4 - x_T}$

Constraints:

- $\text{next}(w_2) \cdot (w_2 - 1) = 0$
- $\text{next}(w_3) \cdot (w_3 - 1) = 0$
- $\text{next}(w_4) \cdot (w_4 - 1) = 0$
- $\text{next}(w_5) \cdot (w_5 - 1) = 0$
- $\text{next}(w_6) \cdot (w_6 - 1) = 0$

<sup>4</sup>Using the results from <https://arxiv.org/pdf/math/0208038.pdf>

- $(w_2 - w_0) \cdot \text{next}(w_7) = w_3 - (2\text{next}(w_2) - 1) \cdot w_1$
- $(w_7 - w_0) \cdot \text{next}(w_8) = w_8 - (2\text{next}(w_3) - 1) \cdot w_1$
- $(w_{10} - w_0) \cdot \text{next}(w_9) = w_{11} - (2\text{next}(w_4) - 1) \cdot w_1$
- $(w_{12} - w_0) \cdot \text{next}(w_{10}) = w_{13} - (2\text{next}(w_5) - 1) \cdot w_1$
- $(\text{next}(w_0) - w_0) \cdot \text{next}(w_{11}) = \text{next}(w_1) - (2\text{next}(w_6) - 1) \cdot w_1$
- $(2 \cdot w_3 - \text{next}(w_7) \cdot (2 \cdot w_2 - \text{next}(w_7)^2 + w_0))^2 = (2 \cdot w_2 - \text{next}(w_7)^2 + w_0)^2 \cdot (w_7 - w_0 + \text{next}(w_7)^2)$
- $(2 \cdot w_8 - \text{next}(w_8) \cdot (2 \cdot w_7 - \text{next}(w_8)^2 + w_0))^2 = (2 \cdot w_7 - \text{next}(w_8)^2 + w_0)^2 \cdot (w_9 - w_0 + \text{next}(w_8)^2)$
- $(2 \cdot w_{10} - \text{next}(w_9) \cdot (2 \cdot w_9 - \text{next}(w_9)^2 + w_0))^2 = (2 \cdot w_9 - \text{next}(w_9)^2 + w_0)^2 \cdot (w_{11} - w_0 + \text{next}(w_9)^2)$
- $(2 \cdot w_{12} - \text{next}(w_{10}) \cdot (2 \cdot w_{11} - \text{next}(w_{10})^2 + w_0))^2 = (2 \cdot w_{11} - \text{next}(w_{10})^2 + w_0)^2 \cdot (w_{13} - w_0 + \text{next}(w_{10})^2)$
- $(2 \cdot w_{14} - \text{next}(w_{11}) \cdot (2 \cdot w_{13} - \text{next}(w_{11})^2 + w_0))^2 = (2 \cdot w_{13} - \text{next}(w_{11})^2 + w_0)^2 \cdot (\text{next}(w_0) - w_0 + \text{next}(w_{11})^2)$
- $(w_8 + w_3) \cdot (2 \cdot w_2 - \text{next}(w_7)^2 + w_0) = (w_2 - w_7) \cdot (2 \cdot w_3 - \text{next}(w_7) \cdot (2 \cdot w_2 - \text{next}(w_7)^2 + w_0))$
- $(w_{10} + w_8) \cdot (2 \cdot w_7 - \text{next}(w_8)^2 + w_0) = (w_7 - w_9) \cdot (2 \cdot w_8 - \text{next}(w_8) \cdot (2 \cdot w_7 - \text{next}(w_8)^2 + w_0))$
- $(w_{12} + w_{10}) \cdot (2 \cdot w_9 - \text{next}(w_9)^2 + w_0) = (w_9 - w_{11}) \cdot (2 \cdot w_{10} - \text{next}(w_9) \cdot (2 \cdot w_9 - \text{next}(w_9)^2 + w_0))$
- $(w_{14} + w_{10}) \cdot (2 \cdot w_{11} - \text{next}(w_{10})^2 + w_0) = (w_{11} - w_{13}) \cdot (2 \cdot w_{12} - \text{next}(w_{10}) \cdot (2 \cdot w_{11} - \text{next}(w_{10})^2 + w_0))$
- $(\text{next}(w_1) + w_{14}) \cdot (2 \cdot w_{13} - \text{next}(w_{11})^2 + w_0) = (w_{13} - \text{next}(w_0) \cdot (2 \cdot w_{14} - \text{next}(w_{11}) \cdot (2 \cdot w_{13} - \text{next}(w_{11})^2 + w_0)))$
- $w_5 = 32 \cdot (w_4) + 16 \cdot \text{next}(w_2) + 8 \cdot \text{next}(w_3) + 4 \cdot \text{next}(w_4) + 2 \cdot \text{next}(w_5) + \text{next}(w_6)$

Copy constraints:

- $(x_T, y_T)$  in row  $j$  are copy constrained with  $(x_T, y_T)$  in row  $j + 2$
- $(x_0, y_0)$  in row  $i$  are copy constrained with values from the first doubling circuit
- $(x_0, y_0)$  in row  $j, j \neq i$  are copy constrained with  $(x_5, y_5)$  in row  $j - 1$
- $n = 0$  in row  $i$  and  $n$  in the row  $j, j \neq i$  is copy constrained with  $n'$  in the row  $j - 2$

### 2.3.3 Variable Base Endo-Scalar Multiplication

For  $R = [b]T$ , where  $b = [b_n \dots b_0]$  and  $b_i \in \{0, 1\}$ :<sup>5</sup>

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

The first step of the algorithm are verified by the doubling and unified addition circuit.

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$i$	$x_T$	$y_T$	--	--	$x_P$	$y_P$	$n = 0$	$x_R$	$y_R$	$s_1$	$s_3$	$b_1$	$b_2$	$b_3$	$b_4$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i + 63$	$x_T$	$y_T$	--	--	$x_P$	$y_P$	$n$	$x_R$	$y_R$	$s_1$	$s_3$	$b_1$	$b_2$	$b_3$	$b_4$
$i + 64$	--	--	--	--	$x_P$	$y_P$	$n$	--	--	--	--	--	--	--	--

Evaluations:

- The first  $x_P, y_P$  are equal to  $2 \cdot ((x_T, y_T) + ((\text{endo}) \cdot x_T, y_T))$
- $b_i$  are bits of the  $k$ , first  $b_1$  is the most significant bit of  $k$ ,  $n$  is an accumulator of  $b_i$ .
- $(x_R, y_R) - (x_P, y_P) = (x_P, y_P) + (1 + (\text{endo} - 1) \cdot b_2)x_T, (2b_1 - 1)y_T$
- $(\text{next}(x_P), \text{next}(y_P)) - (x_R, y_R) = (x_R, y_R) + ((\text{endo} - 1) \cdot b_2)x_T, (2b_1 - 1)y_T$
- $s_1 = \frac{(2b_1 - 1) \cdot y_T - y_P}{(1 + (\text{endo} - 1) \cdot b_2)x_T - x_P}$
- $s_3 = \frac{(2b_1 - 1) \cdot y_T - y_R}{(1 + (\text{endo} - 1) \cdot b_2)x_T - x_R}$

<sup>5</sup>Using the results from <https://eprint.iacr.org/2019/1021.pdf>

Constraints:

- $w_{11} \cdot (w_{11} - 1) = 0$
- $w_{12} \cdot (w_{12} - 1) = 0$
- $w_{13} \cdot (w_{13} - 1) = 0$
- $w_{14} \cdot (w_{14} - 1) = 0$
- $((1 + (\text{endo} - 1) \cdot w_{12}) \cdot w_0 - w_4) \cdot w_9 = (2 \cdot w_{11} - 1) \cdot w_1 - w_5$
- $(2 \cdot w_4 - w_9^2 + (1 + (\text{endo} - 1) \cdot w_{12}) \cdot w_0) \cdot ((w_4 - w_7) \cdot w_9 + w_8 + w_5) = (w_4 - w_7) \cdot 2 \cdot w_5$
- $(w_8 + w_5)^2 = (w_4 - w_7)^2 \cdot (w_9^2 - (1 + (\text{endo} - 1) \cdot w_{12}) \cdot w_0 + w_7)$
- $((1 + (\text{endo} - 1) \cdot w_{12}) \cdot w_0 - w_7) \cdot w_{10} = (2 \cdot w_{13} - 1) \cdot w_1 - w_8$
- $(2 \cdot w_7 - w_{10}^2 + (1 + (\text{endo} - 1) \cdot w_{14}) \cdot w_0) \cdot ((w_7 - \text{next}(w_4)) \cdot w_{10} + \text{next}(w_5) + w_8) = (w_7 - \text{next}(w_4)) \cdot 2 \cdot w_8$
- $(\text{next}(w_4) + w_8)^2 = (w_7 - \text{next}(w_4))^2 \cdot (w_{10}^2 - (1 + (\text{endo} - 1) \cdot w_{14}) \cdot w_0 + \text{next}(w_4))$
- $\text{next}(w_6) = 16 \cdot w_6 + 8 \cdot w_{11} + 4 \cdot w_{12} + 2 \cdot w_{13} + w_{14}$

Copy constraints:

- $(x_T, y_T)$  in row  $j$  are copy constrained with  $(x_T, y_T)$  in row  $j + 1$
- $(x_P, y_P)$  in row  $i$  are copy constrained with values from the first doubling circuit

## 2.4 Multi-Scalar Multiplication Circuit

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

**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  $A = A + S$

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

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

## 2.5 Poseidon Circuit

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).

## 2.6 Other Circuits

### 2.6.1 Endo-Scalar Computation

Let  $\alpha$  be equals to  $\phi(b)$ , where  $b \in 0, 1^\lambda$ .

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$i$	$n_0$	$n_8$	$a_0$	$b_0$	$a_8$	$b_8$	—	$x_0$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i + 7$	$n_0$	$n_8$	$a_0$	$b_0$	$a_8$	$b_8$	$res$	$x_0$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$

Evaluations:

- In the first row  $n_0 = 0$ ,  $a_0 = 2$ ,  $b_0 = 2$ .
- $x_i$  are 2-bits chunks of the  $b$ , first  $x_0$  is the most significant bit of  $b$ ,  $n$  is an accumulator of  $x_i$ .
- The values  $(a_8, b_8)$  are 8 iterations of the following computations:

$$(a_i, b_i) = (2 \cdot a_{i-1} + c_f(x_{i-1}), 2 \cdot b_{i-1} + d_f(x_{i-1})), \text{ where } c_f(x) = 2/3 \cdot x^3 - 5/2 \cdot x^2 + 11/6 \cdot x \text{ and } d_f(x) = 2/3 \cdot x^3 - 7/2 \cdot x^2 + 29/6 \cdot x - 1.$$

Constraints:

- $w_7 \cdot (w_7 - 1) \cdot (w_7 - 2) \cdot (w_7 - 3) = 0$
- $w_8 \cdot (w_8 - 1) \cdot (w_8 - 2) \cdot (w_8 - 3) = 0$
- $w_9 \cdot (w_9 - 1) \cdot (w_9 - 2) \cdot (w_9 - 3) = 0$
- $w_{10} \cdot (w_{10} - 1) \cdot (w_{10} - 2) \cdot (w_{10} - 3) = 0$
- $w_{11} \cdot (w_{11} - 1) \cdot (w_{11} - 2) \cdot (w_{11} - 3) = 0$
- $w_{12} \cdot (w_{12} - 1) \cdot (w_{12} - 2) \cdot (w_{12} - 3) = 0$
- $w_{13} \cdot (w_{13} - 1) \cdot (w_{13} - 2) \cdot (w_{13} - 3) = 0$
- $w_{14} \cdot (w_{14} - 1) \cdot (w_{14} - 2) \cdot (w_{14} - 3) = 0$

- $w_4 = 256 \cdot w_2 + 128 \cdot c_f(w_6) + 64 \cdot c_f(w_7) + 32 \cdot c_f(w_8) + 16 \cdot c_f(w_9) + 8 \cdot c_f(w_{10}) + 4 \cdot c_f(w_{11}) + 2 \cdot c_f(w_{12}) + c_f(w_{13})$
- $w_5 = 256 \cdot w_3 + 128 \cdot d_f(w_6) + 64 \cdot d_f(w_7) + 32 \cdot d_f(w_8) + 16 \cdot d_f(w_9) + 8 \cdot d_f(w_{10}) + 4 \cdot d_f(w_{11}) + 2 \cdot d_f(w_{12}) + d_f(w_{13})$
- $w_1 = 2^{16} \cdot w_0 + 2^{14} \cdot w_6 + 2^{12} \cdot w_7 + 2^{10} \cdot w_8 + 2^8 \cdot w_9 + 2^6 \cdot w_{10} + 2^4 \cdot w_{11} + 2^2 \cdot w_{12} + w_{13}$
- for  $i + 7$ :

1.  $w_6 = \mathbf{endo} \cdot w_4 + w_5$

Copy constraints:

- $n_0, a_0, b_0$  in row  $j + 1$  are copy constrained with  $(n_8, a_8, b_8)$  in row  $j$

## 2.7 Proof Verification Component

Let  $\mathbf{G}$  be a group of points on the elliptic curve  $E(\mathbb{F}_p)$ ,  $|\mathbf{G}| = r$ .

Kimchi verification procedure includes operations over two groups:  $\mathbf{G}$  and scalars of  $\mathbf{G}$ . Thus, the verification circuit has to handle operations over two fields:  $\mathbb{F}_p$  and  $\mathbb{F}_r$ . This could be achieved either with non-native arithmetic circuits<sup>6</sup> or via splitting the verification into two proofs over different fields. Here we use the second option.

---

<sup>6</sup>For instance, see <https://www.plonk.cafe/t/non-native-field-arithmetic-with-turboplonek-plookup-etc/90>

1. for each  $\pi_i$ :
    - 1.1 **random\_oracle**( $p_{\text{comm}}, \pi_i$ ):
      - 1.1.1 Copy **joint\_combiner** from PI
      - 1.1.2 Copy  $\beta, \gamma$  from PI
      - 1.1.3 Copy  $\alpha_c$  from PI
      - 1.1.4  $\alpha = \phi(\alpha_c)$
      - 1.1.5 Copy  $\zeta_c$  from PI
      - 1.1.6  $\zeta = \phi(\zeta_c)$
      - 1.1.7 Initialize  $H_{\mathbb{F}_r}$
      - 1.1.8 Copy  $H_{\mathbb{F}_q}.\text{digest}$  from PI
      - 1.1.9  $H_{\mathbb{F}_r}.\text{absorb}(H_{\mathbb{F}_q}.\text{digest})$
      - 1.1.10  $\zeta_1 = \zeta^n$  for  $n = |\text{domain}|$
      - 1.1.11  $\zeta_w = \zeta \cdot \omega$
      - 1.1.12 **all\_alphas** =  $[1, \alpha, \dots, \alpha^{\text{next\_power}}]$
      - 1.1.13 **lagrange** =  $[\zeta - \text{domain}.w, \dots, \zeta_w - \text{domain}.w]$  L195
      - 1.1.14 **lagrange** =  $[1/\text{lagrange}[0], \dots]$
      - 1.1.15 **p\_eval**[0] =  $(\sum(\text{pub}[i] \cdot \text{domain}[i] \cdot (-\text{lagrange}[i])) \cdot (\zeta_1 - 1) \cdot \frac{1}{|\text{domain}|})$
      - 1.1.16 **p\_eval**[1] =  $(\sum(\text{pub}[i] \cdot \text{domain}[i] \cdot (-\text{lagrange}[\text{pub}.len + i])) \cdot (\zeta_w^n - 1) \cdot \frac{1}{|\text{domain}|})$
      - 1.1.17  $H_{\mathbb{F}_r}.\text{absorb}(\text{p\_eval}[0])$
      - 1.1.18  $H_{\mathbb{F}_r}.\text{absorb}(\text{evals}[0])$  <- PI src -> plonk\_sponge.rs L41
      - 1.1.19  $H_{\mathbb{F}_r}.\text{absorb}(\text{p\_eval}[1])$
      - 1.1.20  $H_{\mathbb{F}_r}.\text{absorb}(\text{evals}[1])$  <- PI
      - 1.1.21 Copy  $\bar{L}(\zeta\omega)$  from PI
      - 1.1.22  $H_{\mathbb{F}_r}.\text{absorb}(\bar{L}(\zeta\omega))$
      - 1.1.23  $v = \phi(H_{\mathbb{F}_r}.\text{squeeze}())$
      - 1.1.24  $u = \phi(H_{\mathbb{F}_r}.\text{squeeze}())$
      - 1.1.25 Compute evaluation of  $\eta_i(\zeta), \eta_i(\zeta\omega)$  for  $0 \leq i < N_{\text{prev}}$ :
        - 1.1.25.1 **powers\_of\_evals** =  $[\zeta^{\text{max\_poly\_size}}, \zeta_w^{\text{max\_poly\_size}}]$
        - 1.1.25.2 ...
      - 1.1.26 Compute evaluation of  $\bar{L}(\zeta)$ :
        - 1.1.26.1 ...
    - 1.2 Combine evals (ploynomial evals) L412
    - 1.3 **f\_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\_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 **f\_scalars** :=  $\{-z(\zeta\omega) \cdot \beta \cdot \alpha_0 \cdot \text{zkp}(\zeta) \cdot s_{\text{perm}}, w_0(\zeta) \cdot w_1(\zeta), w_0(\zeta), w_1(\zeta), 1, s_{\text{psdn}}, s_{\text{rc}}, s_{\text{ec\_add}}, s_{\text{ec\_dbl}}, s_{\text{ec\_endo}}, s_{\text{ec\_vbase}}\}$
    - 1.6 **PE** is a set of elements of the form  $(f_{\text{comm}}, f(\zeta), f(\zeta\omega))$  for the following polynomials:  
 $\eta_0, \dots, \eta_{N_{\text{prev}}}, \text{pub}, w_0, \dots, w_{N_{\text{wires}}}, z, S_{\sigma_0}, \dots, S_{\sigma_{N_{\text{perm}}}}, \bar{L}$
    - 1.7  $\mathcal{P}_i = \{H_{\mathbb{F}_q}, \zeta, v, u, \mathbf{PE}, o_{\pi_i}\}$
  2. **final\_check\_scalar\_field**( $\mathcal{P}_0, \dots, \mathcal{P}_{\text{batch\_size}}$ )
-

1. for each  $\pi_i$ :
    - 1.1  $pub_{\text{comm}} = -\text{MSM}(\mathbf{L}, \text{pub}) \in \mathbb{G}$ , where  $\mathbf{L}$  is Lagrange bases vector
    - 1.2  $\text{random\_oracle}(p_{\text{comm}}, \pi_i)$ :
      - 1.2.1  $H_{\mathbb{F}_q}.\text{absorb}(pub_{\text{comm}} || w_{0,\text{comm}} || \dots || w_{N_{\text{wires}},\text{comm}})$
      - 1.2.2  $\text{joint\_combiner} = H_{\mathbb{F}_q}.\text{squeeze}() <- \text{PI check}$
      - 1.2.3  $H_{\mathbb{F}_q}.\text{absorb}(\text{LOOKUP})$  L146, commitments sorted
      - 1.2.4  $\beta, \gamma = H_{\mathbb{F}_q}.\text{squeeze}() <- \text{PI check}$
      - 1.2.5  $H_{\mathbb{F}_q}.\text{absorb}(\text{LOOKUP2})$  L156m commitments aggregated
      - 1.2.6  $H_{\mathbb{F}_q}.\text{absorb}(z_{\text{comm}})$
      - 1.2.7  $\alpha = H_{\mathbb{F}_q}.\text{squeeze}() <- \text{PI check}$
      - 1.2.8  $H_{\mathbb{F}_q}.\text{absorb}(t_{1,\text{comm}} || \dots || t_{N_{\text{perm}},\text{comm}} || \dots || \infty ||)$
      - 1.2.9  $\zeta = H_{\mathbb{F}_q}.\text{squeeze}() <- \text{PI check}$
      - 1.2.10 Get digest from  $H_{\mathbb{F}_q} <- \text{PI check}$
    - 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  $\mathbf{f}_{\text{scalars}} := \{-z(\zeta\omega) \cdot \beta \cdot \alpha_0 \cdot \text{zkp}(\zeta) \cdot s_{\text{perm}}, w_0(\zeta) \cdot w_1(\zeta), w_0(\zeta), w_1(\zeta), 1, s_{\text{psdn}}, s_{\text{rc}}, s_{\text{ec\_add}}, s_{\text{ec\_dbl}}, s_{\text{ec\_endo}}, s_{\text{ec\_vbase}}\}$
    - 1.6  $f_{\text{comm}} = \text{MSM}(\mathbf{f}_{\text{base}}, \mathbf{f}_{\text{scalars}})$
    - 1.7 Copy from PI  $(\zeta^n - 1)$
    - 1.8  $\bar{L}_{\text{comm}} = f_{\text{comm}} - t_{\text{comm}} \cdot (\zeta^n - 1)$
    - 1.9  $\mathbf{PE}$  is a set of elements of the form  $(f_{\text{comm}}, f(\zeta), f(\zeta\omega))$  for the following polynomials:  
 $\eta_0, \dots, \eta_{N_{\text{prev}}}, \text{pub}, w_0, \dots, w_{N_{\text{wires}}}, z, S_{\sigma_0}, \dots, S_{\sigma_{N_{\text{perm}}}}, \bar{L}$
    - 1.10  $\mathcal{P}_i = \{H_{\mathbb{F}_q}, \zeta, v, u, \mathbf{PE}, o_{\pi_i}\}$
  2.  $\text{final\_check\_base\_field}(\mathcal{P}_0, \dots, \mathcal{P}_{\text{batch\_size}})$
-

---

**Algorithm 6** Final Check - Scalar Field

---

**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.  $\rho_1 \rightarrow \mathbb{F}_r \leftarrow$  should be calculated as poseidon from  $H_{i, \mathbb{F}_q}$  state
  2.  $\rho_2 \rightarrow \mathbb{F}_r$
  3.  $r_0 = r'_0 = 1$
  4. for  $0 \leq i < \text{batch\_size}$ :
    - 4.1  $cip_i = \text{combined\_inner\_product}(\zeta_i, \zeta_i \omega, v_i, u_i, \mathbf{PE}_i) \leftarrow$  PI check
    - 4.2 Calculate opening challenges  $\xi_{i,j}$  from limbs in  $o_{\pi_i} \leftarrow$  PI?
    - 4.3 Calculate inversion from  $\xi_{i,j}$
    - 4.4 Copy limbs  $c_i\_limbs$  from PI
    - 4.5  $c_i = \phi(c_i\_limbs)$
    - 4.6  $h_i(X) := \prod_{k=0}^{\log(d+1)-1} (1 + \xi_{\log(d+1)-k} X^{2^k})$ , where  $d = \text{lr\_rounds}$
    - 4.7  $b_i = h_i(\zeta) + u_i \cdot h_i(\zeta \omega)$
    - 4.8  $sg = -r_i \cdot \text{opening.z1} - r'_i$
    - 4.9  $r_i = r_{i-1} \cdot \rho_1$
    - 4.10  $r'_i = r'_{i-1} \cdot \rho_2$
- 

---

**Algorithm 7** Final Check - Base Field

---

**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. for  $0 \leq i < \text{batch\_size}$ :
    - 1.1 Get limbs  $cip_i$  from PI
    - 1.2  $H_{i, \mathbb{F}_q}.\text{absorb}(cip_i - 2^{255})$
    - 1.3  $U_i = (H_{i, \mathbb{F}_q}.\text{squeeze}()).\text{to\_group}()$
    - 1.4 Calculate opening challenges  $\xi_{i,j}$  from  $o_{\pi_i} \leftarrow$  PI output as limbs:
      - 1.4.1 ?????
    - 1.5  $H_{i, \mathbb{F}_q}.\text{absorb}(\text{openings}.\delta)$  L791
    - 1.6  $h_i(X) := \prod_{k=0}^{\log(d+1)-1} (1 + \xi_{\log(d+1)-k} X^{2^k})$ , where  $d = \text{lr\_rounds}$
    - 1.7  $C_i = \sum_j v_i^j (\sum_k r_i^k f_{j, \text{comm}})$ , where  $f_{j, \text{comm}}$  from  $\mathbf{PE}_i$ .
    - 1.8  $Q_i = \sum (\xi_{i,j} \cdot L_{i,j} + \xi_{i,j}^{-1} \cdot R_j) + cip_i \cdot U_i + C_i$
    - 1.9  $c_i = H_{i, \mathbb{F}_q}.\text{squeeze}() \leftarrow$  PI
    - 1.10 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  $r'_i$ .
  2. Fq:  $\text{res} = \sum_i r^i (c_i Q_i + \text{delta}_i - (z_{i,1}(\hat{G}_i + b_i U_i) + z_{i,2} H))$
  3. Fq: return  $\text{res} == 0$
-



---

**Algorithm 8** 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. Fr:  $s = \sum_{i=0}^k \xi^i \cdot (f_i(\zeta_1) + r \cdot f_i(\zeta_2))$
-

## 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

The verification logic is split to several parts:

1. Verification Key Definition
2. LPC/FRI auxiliary proof deserialization

### 3.2 Verification Logic API Reference

### 3.3 Input Data Structures

## Chapter 4

# Appendix A. In-EVM Mina State

This introduces a description for in-EVM Mina Protocol state handling mechanism which is supposed to provide a bridge user with the way to verify plaintext transactions coming from Mina database commit log on EVM.

### 4.1 Overview

The protocol described literally replicates Mina's commit log construction protocol on EVM. The overall process description is as follows:

---

**Algorithm 9** Commit Log Construction Overview

---

1. A user retrieves a replication packet  $B_n$  containing some transaction  $T$  from Mina's commit log.
  2. A user submits the replication packet  $B_n$  to the in-EVM piece of logic.
  3. The in-EVM piece of logic emplaces the replication packet  $B_n$  into the backwards-linked list  $C$ .
  4. The in-EVM piece of logic computes a Poseidon hash  $H_{B_n}$  of a replication packet  $B_n$  and inserts such one in a Merkle Tree  $T$ .
  5. The in-EVM piece of logic uses a Merkle Tree's hash  $H_{B_n}$  of a particular replication packet  $B_n$  as an input to the state proof verification mechanism, taking the state proof from the original Mina's cluster in the same time, corresponding to the replication packet  $B_n$  sequo.
  6. In case the verification of a state proof corresponding to the replication packet  $B_n$  was completed successfully, such a replication packet  $B_n$  can be considered valid and appended to the backwards-linked list, representing in-EVM Mina's commit log.
  7. In case the verification of a state proof corresponding to the replication packet  $B_n$  wasn't completed successfully, then a replication packet  $B_n$  gets rejected by the in-EVM piece of logic.
  8. In case there are more than a single replication packet  $B_n$  (e.g.  $B_{n_1}$  and  $B_{n_2}$ ) and each of them is being considered valid, the backward-linked list used to store such replication packets turns into the tree containing several branches of backward-linked lists  $C_1, \dots, C_M$ .
  9. In case several branches  $C_1, \dots, C_M$  are introduced, the Mina's Ouroboros modification chain selection rule applies to pick the same branch the original Mina's cluster chain selection rule picked.
- 

$T_{n_1, n_2}$  allows to provide a successful transaction from  $\{B_{n_1}, \dots, B_{n_2}\}$  to the Ethereum-based proof verifier later.

Ouroboros' consensus protocol chain selection rule which is supposed to handle potentially incorrect replication packet data submitted by the user (and to keep the in-EVM commit log consistent with the actual Mina's one) is defined as follows:

Here,  $C_{loc}$  is the local commit log sequence,  $N = C_1, \dots, C_M$  is the list of potential commit log sequences to choose from. The function  $getMinDen(C)$  outputs the minimum of all the window densities observed thus far in  $C$ .

---

**Algorithm 10**  $\text{getMinDen}(C)$ 

---

Let  $B_{last}$  be the last block in  $C$ .

1. if  $B_{last} = G$  then // i.e., if  $B_{last}$  is the genesis block
  2. return 0
  3. else
  4. Parse  $B_{last}$  to obtain the parameter  $\text{minDen}$ .
  5. return  $\text{minDen}$
- 

The function  $\text{isShortRange}(C, C')$  outputs whether or not the chains fork in the “short range” or not.

---

**Algorithm 11**  $\text{isShortRange}(C1, C2)$ 

---

1. Let  $\text{prevLockcp}$  and  $\text{prevLockcp}$  be the  $\text{prevLockcp}$  components in the 12 last blocks of  $C1$ ,  $C2$ , respectively.
  2. if  $\text{prevLockcp} = \text{prevLockcp}$  then
  3. return  $\top$
  4. else
  5. return  $\perp$
- 

---

**Algorithm 12**  $\text{maxvalid-sc}(C_{loc}, N = C_1, \dots, C_M, k)$ 

---

1. Set  $C_{max} \Leftarrow C_{loc}$  // Compare  $C_{loc}$  with each candidate chain in  $N$
  2. for  $i = 1, \dots, M$  do if  $\text{isShortRange}(C_i, C_{max})$  then // Short-range fork  
if  $|C_i| > |C_{max}|$  then  
Set  $C_{max} \Leftarrow C_i$   
end if  
else //Long-range fork  
if  $\text{getMinDen}(C_i) > \text{getMinDen}(C_{max})$  then  
Set  $C_{max} \Leftarrow C_i$   
end if  
end if  
end for
  3. return  $C_{max}$
- 

#### 4.1.1 Purpose

The protocol is supposed to make it possible for the users to prove a particular transaction to the in-EVM Mina’s commit log replica to be able to prove it actually belongs to Mina’s commit log.

The overview of such a mechanism is as follows:

---

**Algorithm 13** Transaction Plaintext Data Proving Approach

---

1. A user retrieves the transaction  $T$  from Mina's database commit log
  2. A user compares the transaction  $T$  with the contents of the in-EVM Mina's commit log representation.
  3. If a trivial comparison results in a match, Mina's data from the transaction  $T$  can be considered valid for the in-EVM usage.
  4. Otherwise, the transaction is supposed to be rejected.
-

# 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>.