In-EVM Mina State Verification

Technical Reference

Alisa Cherniaeva
a.cherniaeva@nil.foundation
`=nil;` Crypto3 (`https://crypto3.nil.foundation`)

Ilia Shirobokov
i.shirobokov@nil.foundation
`=nil;` Crypto3 (`https://crypto3.nil.foundation`)

Mikhail Komarov
nemo@nil.foundation
`=nil;` Foundation (`https://nil.foundation`)

May 10, 2022

Contents

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 verificator.

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 generilize the original protocol for use with PLONK with custom gates [4], [5] and lookup arguments [6], [7].[1]

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:

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

[1] Details on the proof system: https://github.com/NilFoundation/evm-mina-verification/tree/master/docs/proof_system

2.2.2 Verification Algorithm

Notations

$N_{\texttt{wires}}$	Number of wires ('advice columns')
$N_{\texttt{perm}}$	Number of wires that are included in the permutation argument
$N_{\texttt{prev}}$	Number of previous challenges
$S_{\sigma_i}(\mathrm{X})$	Permutation polynomials for $0 \leq i < N_{\texttt{perm}}$
$pub(X)$	Public input polynomial
$w_i(X)$	Witness polynomials for $0 \leq i < N_{\texttt{wires}}$
$\eta_i(X)$	Previous challenges polynomials for $0 \leq i < N_{\texttt{prev}}$
ω	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 $\texttt{MSM}(\mathbf{s}, \mathbf{G})$ 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 π constains (here $\mathbb{F}_r$ is a scalar field of $\mathbb{G}$):

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

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

[2] See https://o1-labs.github.io/mina-book/crypto/plonk/maller_15.html

Algorithm 1 Verification

Input: $\pi_0, \ldots, \pi_{\texttt{batch\_size}}$ (see 2.2.2)
Output: `acc` or `rej`

1. for each π_i:

   1.1 $pub_{\texttt{comm}} = \texttt{MSM}(\mathbf{L}, \texttt{pub}) \in \mathbb{G}$, where $\mathbf{L}$ is Lagrange bases vector

   1.2 `random_oracle`$(p_{\texttt{comm}}, \pi_i)$:

      1.2.1 $H_{\mathbb{F}_q}.\texttt{absorb}(pub_{\texttt{comm}}||w_{0,\texttt{comm}}||...||w_{N_{\text{wires}},\texttt{comm}})$

      1.2.2 $\beta, \gamma = H_{\mathbb{F}_q}.\texttt{squeeze}()$

      1.2.3 $H_{\mathbb{F}_q}.\texttt{absorb}(z_{\texttt{comm}})$

      1.2.4 $\alpha = \phi(H_{\mathbb{F}_q}.\texttt{squeeze}())$

      1.2.5 $H_{\mathbb{F}_q}.\texttt{absorb}(t_{1,\texttt{comm}}||...||t_{N_{\text{perm}},\texttt{comm}}||...||\infty||)$

      1.2.6 $\zeta = \phi(H_{\mathbb{F}_q}.\texttt{squeeze}())$

      1.2.7 Transfrorm $H_{\mathbb{F}_q}$ to $H_{\mathbb{F}_r}$

      1.2.8 $H_{\mathbb{F}_r}.\texttt{absorb}(pub(\zeta)||w_0(\zeta)||...||w_{N_{\text{wires}}}(\zeta)||S_0(\zeta)||...||S_{N_{\text{perm}}}(\zeta))$

      1.2.9 $H_{\mathbb{F}_r}.\texttt{absorb}(pub(\zeta\omega)||w_0(\zeta\omega)||...||w_{N_{\text{wires}}}(\zeta\omega)||S_0(\zeta\omega)||...||S_{N_{\text{perm}}}(\zeta\omega))$

      1.2.10 $H_{\mathbb{F}_r}.\texttt{absorb}(\bar{L}(\zeta\omega))$

      1.2.11 $v = \phi(H_{\mathbb{F}_r}.\texttt{squeeze}())$

      1.2.12 $u = \phi(H_{\mathbb{F}_r}.\texttt{squeeze}())$

      1.2.13 Compute evaluation of $\eta_i(\zeta), \eta_i(\zeta\omega)$ for $0 \leq i < N_{\texttt{prev}}$

      1.2.14 Compute evaluation of $\bar{L}(\zeta)$

   1.3 $\mathbf{f}_{\text{base}} := \{S_{\sigma_{N_{\text{perm}}-1},\texttt{comm}}, \texttt{gate}_{\text{mult},\texttt{comm}}, w_{0,\texttt{comm}}, w_{1,\texttt{comm}}, w_{2,\texttt{comm}}, q_{\text{const},\texttt{comm}}, \texttt{gate}_{\text{psdn},\texttt{comm}}, \texttt{gate}_{\text{rc},\texttt{comm}}, \texttt{gate}_{\text{ec\_add},\texttt{comm}}, \texttt{gate}_{\text{ec\_dbl},\texttt{comm}}, \texttt{gate}_{\text{ec\_endo},\texttt{comm}}, \texttt{gate}_{\text{ec\_vbase},\texttt{comm}}\}$

   1.4 $s_{\text{perm}} := (w_0(\zeta) + \gamma + \beta \cdot S_{\sigma_0}(\zeta)) \cdot ... \cdot (w_5(\zeta) + \gamma + \beta \cdot S_{\sigma_{N_{\text{perm}}}}(\zeta))$

   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 $f_{\texttt{comm}} = \text{MSM}(\mathbf{f}_{\text{base}}, \mathbf{f}_{\text{scalars}})$

   1.7 $\bar{L}_{\texttt{comm}} = f_{\texttt{comm}} - t_{\texttt{comm}} \cdot (\zeta^n - 1)$

   1.8 **PE** is a set of elements of the form $(f_{\texttt{comm}}, f(\zeta), f(\zeta\omega))$ for the following polynomials: $\eta_0, \ldots, \eta_{N_{\texttt{prev}}}, pub, w_0, \ldots, w_{N_{\texttt{wires}}}, z, S_{\sigma_0}, \ldots, S_{\sigma_{N_{\texttt{perm}}}}, \bar{L}$

   1.9 $\mathcal{P}_i = \{H_{\mathbb{F}_q}, \zeta, v, u, \mathbf{PE}, o_{\pi_i}\}$

2. `final_check`$(\mathcal{P}_0, \ldots, \mathcal{P}_{\texttt{batch\_size}})$

Algorithm 2 Final Check

Input: $\pi_0, \dots, \pi_{\texttt{batch\_size}}$, 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. $\rho_1 \to \mathbb{F}_r$
2. $\rho_2 \to \mathbb{F}_r$
3. $r_0 = r'_0 = 1$
4. for $0 \le i < \texttt{batch\_size}$:
   4.1 $cip_i = \texttt{combined\_inner\_product}(\zeta_i, \zeta_i\omega, v_i, u_i, \mathbf{PE}_i)$
   4.2 $H_{i,\mathbb{F}_q}.\texttt{absorb}(cip_i - 2^{255})$
   4.3 $U_i = (H_{i,\mathbb{F}_q}.\texttt{squeeze}()).\texttt{to\_group}()$
   4.4 Calculate opening challenges $\xi_{i,j}$ from o_{π_i}
   4.5 $h_i(X) \coloneqq \prod_{k=0}^{\log(d+1)-1}(1 + \xi_{\log(d+1)-k}X^{2^k})$, where $d = \texttt{lr\_rounds}$
   4.6 $b_i = h_i(\zeta) + u_i \cdot h_i(\zeta\omega)$
   4.7 $C_i = \sum\limits_j v_i^j(\sum\limits_k r_i^k f_{j,\texttt{comm}})$, where $f_{j,\texttt{comm}}$ from $\mathbf{PE}_i$.
   4.8 $Q_i = \sum(\xi_{i,j} \cdot L_{i,j} + \xi_{i,j}^{-1} \cdot R_j) + cip_i \cdot U_i + C_i$
   4.9 $c_i = \phi(H_{i,\mathbb{F}_q}.\texttt{squeeze}())$
   4.10 $r_i = r_{i-1} \cdot \rho_1$
   4.11 $r'_i = r'_{i-1} \cdot \rho_2$
   4.12 Check $\hat{G}_i =< s, G >$, where s is set of $h(X)$ coefficients.
   Remark: This check can be done inside the `MSM` below using r'_i.
5. $\texttt{res} = \sum\limits_i r^i(c_iQ_i + delta_i - (z_{i,1}(\hat{G}_i + b_iU_i) + z_{i,2}H))$
6. `return 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. $s = \sum\limits_{i=0}^{k} \xi^i \cdot (f_i(\zeta_1) + r \cdot f_i(\zeta_2))$

We use the same 15-wires PLONK circuits that are designed for Mina.[3]

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	$\texttt{inv}_y$	$\texttt{inv}_x$	...	...	...	...

Evaluations:

- Addition case:

[3] https://o1-labs.github.io/mina-book/specs/15_wires/15_wires.html

- $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$
- `inf` $= 1$ if (x_3, y_3) is a point-at-infinity, `inf` $= 0$ otherwise
- `same_x` $= 1$ if $x_1 = x_2$, `same_x` $= 0$ otherwise
- $s = \frac{y_1 - y_2}{x_1 - x_2}$ if $x_1 \neq x_2$, $s = 0$ otherwise
- $\texttt{inv}_y = \frac{1}{y_2 - y_1}$ if $y_2 \neq y_1$, $\texttt{inv}_y = 0$ otherwise
- $\texttt{inv}_x = \frac{1}{x_2 - x_1}$ if $x_2 \neq x_1$, $\texttt{inv}_x = 0$ otherwise

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

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 $w_6 = 0$ (`inf` $= 0$) was introduced.

Constraints details:

- $x_2 - x_1$ zero check:
  1. $w_7 \cdot (w_2 - w_0) = 0 \longleftrightarrow \texttt{same\_x} \cdot (x_2 - x_1)$
     If $x_1 \neq x_2$, then `same_x` $= 0$
  2. $(w_2 - w_0) \cdot w_{10} - (1 - w_7) = 0 \longleftrightarrow (x_2 - x_1) \cdot \texttt{inv}_x - (1 - \texttt{same\_x})$
     If $x_1 \neq x_2$, then $\texttt{inv}_x = (x_2 - x_1)^{-1}$
- Group law constraints:
  1. $w_7 \cdot (2w_8 \cdot w_1 - 3w_0^2) + (1 - w_7) \cdot ((w_2 - w_0 \cdot w_8 - (w_3 - w_1)) \longleftrightarrow$
     $\texttt{same\_x} \cdot (2s \cdot y_1 - 3x_1^2) + (1 - \texttt{same\_x}) \cdot (x_2 - x_1 \cdot s - (y_2 - y_1))$
     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. $(w_3 - w_1) \cdot (w_7 - w_6) = 0 \longleftrightarrow (y_2 - y_1) \cdot (\texttt{same\_x} - \texttt{inf}) = 0$
     We can get inifinity point iff $x_1 = x_2$ and $y_1 \neq y_2$.
     If $y_1 \neq y_2$ then $\texttt{inf} = \texttt{same\_x}$.
  2. $(w_3 - w_1) \cdot w_9 - w_6 = 0 \longleftrightarrow (y_2 - y_1) \cdot \texttt{inv}_y - \texttt{inf}$
     The prover sets $\texttt{inv}_y = 0$ for $y_1 = y_2$.
     If $y_1 \neq y_2$ then $\texttt{inv}_y = (y_2 - y_1)^{-1}$

2.3.2 Variable Base Scalar Multiplication

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

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 alforithm 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:

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

[4] Using the results from https://arxiv.org/pdf/math/0208038.pdf

- $(w_2 - w_0) \cdot \texttt{next}(w_7) = w_3 - (2\texttt{next}(w_2) - 1) \cdot w_1$
- $(w_7 - w_0) \cdot \texttt{next}(w_8) = w_8 - (2\texttt{next}(w_3) - 1) \cdot w_1$
- $(w_{10} - w_0) \cdot \texttt{next}(w_9) = w_{11} - (2\texttt{next}(w_4) - 1) \cdot w_1$
- $(w_{12} - w_0) \cdot \texttt{next}(w_{10}) = w_{13} - (2\texttt{next}(w_5) - 1) \cdot w_1$
- $(\texttt{next}(w_0) - w_0) \cdot \texttt{next}(w_{11}) = \texttt{next}(w_1) - (2\texttt{next}(w_6) - 1) \cdot w_1$

- $(2 \cdot w_3 - \texttt{next}(w_7) \cdot (2 \cdot w_2 - \texttt{next}(w_7)^2 + w_0))^2 = (2 \cdot w_2 - \texttt{next}(w_7)^2 + w_0)^2 \cdot (w_7 - w_0 + \texttt{next}(w_7)^2)$
- $(2 \cdot w_8 - \texttt{next}(w_8) \cdot (2 \cdot w_7 - \texttt{next}(w_8)^2 + w_0))^2 = (2 \cdot w_7 - \texttt{next}(w_8)^2 + w_0)^2 \cdot (w_9 - w_0 + \texttt{next}(w_8)^2)$
- $(2 \cdot w_{10} - \texttt{next}(w_9) \cdot (2 \cdot w_9 - \texttt{next}(w_9)^2 + w_0))^2 = (2 \cdot w_9 - \texttt{next}(w_9)^2 + w_0)^2 \cdot (w_{11} - w_0 + \texttt{next}(w_9)^2)$
- $(2 \cdot w_{12} - \texttt{next}(w_{10}) \cdot (2 \cdot w_{11} - \texttt{next}(w_{10})^2 + w_0))^2 = (2 \cdot w_{11} - \texttt{next}(w_{10})^2 + w_0)^2 \cdot (w_{13} - w_0 + \texttt{next}(w_{10})^2)$
- $(2 \cdot w_{14} - \texttt{next}(w_{11}) \cdot (2 \cdot w_{13} - \texttt{next}(w_{11})^2 + w_0))^2 = (2 \cdot w_{13} - \texttt{next}(w_{11})^2 + w_0)^2 \cdot (\texttt{next}(w_0) - w_0 + \texttt{next}(w_{11})^2)$

- $(w_8 + w_3) \cdot (2 \cdot w_2 - \texttt{next}(w_7)^2 + w_0) = (w_2 - w_7) \cdot (2 \cdot w_3 - \texttt{next}(w_7) \cdot (2 \cdot w_2 - \texttt{next}(w_7)^2 + w_0))$
- $(w_{10} + w_8) \cdot (2 \cdot w_7 - \texttt{next}(w_8)^2 + w_0) = (w_7 - w_9) \cdot (2 \cdot w_8 - \texttt{next}(w_8) \cdot (2 \cdot w_7 - \texttt{next}(w_8)^2 + w_0))$
- $(w_{12} + w_{10}) \cdot (2 \cdot w_9 - \texttt{next}(w_9)^2 + w_0) = (w_9 - w_{11}) \cdot (2 \cdot w_{10} - \texttt{next}(w_9) \cdot (2 \cdot w_9 - \texttt{next}(w_9)^2 + w_0))$
- $(w_{14} + w_{10}) \cdot (2 \cdot w_{11} - \texttt{next}(w_{10})^2 + w_0) = (w_{11} - w_{13}) \cdot (2 \cdot w_{12} - \texttt{next}(w_{10}) \cdot (2 \cdot w_{11} - \texttt{next}(w_{10})^2 + w_0))$
- $(\texttt{next}(w_1) + w_{14}) \cdot (2 \cdot w_{13} - \texttt{next}(w_{11})^2 + w_0) = (w_{13} - \texttt{next}(w_0) \cdot (2 \cdot w_{14} - \texttt{next}(w_{11}) \cdot (2 \cdot w_{13} - \texttt{next}(w_{11})^2 + w_0))$

- $w_5 = 32 \cdot (w_4) + 16 \cdot \texttt{next}(w_2) + 8 \cdot \texttt{next}(w_3) + 4 \cdot \texttt{next}(w_4) + 2 \cdot \texttt{next}(w_5) + \texttt{next}(w_6)$

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 contrained with n' in the row $j-2$

2.3.3 Variable Base Endo-Scalar Multiplication

For $R = [b]T$, where $b = [b_n ... b_0]$ and $b_i \in \{0, 1\}$: [5]

1. $P = [2](\phi(T) + T)$
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 $R - P = P + Q$

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

[5] Using the results from https://eprint.iacr.org/2019/1021.pdf

Constraints:

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

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: $G_0, ..., G_{k-1} \in \mathbb{G}, s_0, ..., s_{k-1} \in \mathbb{F}_r$, 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 $r \coloneqq s_j$, $T \coloneqq G_j$

   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 $A = A + S$

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

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 $r \coloneqq s_j$, $T \coloneqq G_j$

   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 $A = A + S$

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

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:

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

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:
  - $T_{i+1,0} = T_{i,0}^5 \cdot \texttt{MDS}[0][0] + T_{i,1}^5 \cdot \texttt{MDS}[0][1] + T_{i,2}^5 \cdot \texttt{MDS}[0][2] + \texttt{RC}_{i+1,0}$
  - $T_{i+1,1} = T_{i,0}^5 \cdot \texttt{MDS}[1][0] + T_{i,1}^5 \cdot \texttt{MDS}[1][1] + T_{i,2}^5 \cdot \texttt{MDS}[1][2] + \texttt{RC}_{i+1,1}$
  - $T_{i+1,2} = T_{i,0}^5 \cdot \texttt{MDS}[2][0] + T_{i,1}^5 \cdot \texttt{MDS}[2][1] + T_{i,2}^5 \cdot \texttt{MDS}[2][2] + \texttt{RC}_{i+1,2}$

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 α 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:

  $(a_i, b_i) = (2 \cdot a_{i-1} + c_f(x_{i-1}), 2 \cdot b_{i-1} + d_f(x_{i-1}))$, where $c_f(x) = 2/3 \cdot x^3 - 5/2 \cdot x^2 + 11/6 \cdot x$ 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. $w_6 = \texttt{endo} \cdot w_4 + \cdot w_5$

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[6] or via splitting the verification into two proofs over different fields. Here we use the second option.

[6]For instance, see https://www.plonk.cafe/t/non-native-field-arithmetic-with-turboplonk-plookup-etc/90

Algorithm 4 Verifier.Scalar_Field

1. for each π_i:

   1.1 $\texttt{random\_oracle}(p_{\texttt{comm}}, \pi_i)$:

   1.1.1 Copy `joint_combiner` from PI

   1.1.2 Copy β, γ from PI

   1.1.3 Copy α_c from PI

   1.1.4 $\alpha = \phi(\alpha_c)$

   1.1.5 Copy ζ_c from PI

   1.1.6 $\zeta = \phi(\zeta_c)$

   1.1.7 Initialize $H_{\mathbb{F}_r}$

   1.1.8 Copy $H_{\mathbb{F}_q}.\texttt{digest}$ from PI

   1.1.9 $H_{\mathbb{F}_r}.\texttt{absorb}(H_{\mathbb{F}_q}.\texttt{digest})$

   1.1.10 $\zeta_1 = \zeta^n$ for $n = |domain|$

   1.1.11 $\zeta_w = \zeta \cdot \omega$

   1.1.12 $\texttt{all\_alphas} = [1, \alpha, \ldots, \alpha^{next\_power}]$

   1.1.13 $\texttt{lagrange} = [\zeta - domain.w, \ldots, \zeta_w - domain.w]$ L195

   1.1.14 $\texttt{lagrange} = [1/\texttt{lagrange}[0], \ldots]$

   1.1.15 $\texttt{p\_eval}[0] = (\sum(pub[i] \cdot domain[i] \cdot (-lagrange[i])) \cdot (\zeta_1 - 1) \cdot \frac{1}{|domain|}$

   1.1.16 $\texttt{p\_eval}[1] = (\sum(pub[i] \cdot domain[i] \cdot (-lagrange[pub.len + i])) \cdot (\zeta_w^n - 1) \cdot \frac{1}{|domain|}$

   1.1.17 $H_{\mathbb{F}_r}.\texttt{absorb}(p\_eval[0])$

   1.1.18 $H_{\mathbb{F}_r}.\texttt{absorb}(evals[0])$ <- PI src -> plonk_sponge.rs L41

   1.1.19 $H_{\mathbb{F}_r}.\texttt{absorb}(p\_eval[1])$

   1.1.20 $H_{\mathbb{F}_r}.\texttt{absorb}(evals[1])$ <- PI

   1.1.21 $Copy\bar{L}(\zeta\omega)$ from PI

   1.1.22 $H_{\mathbb{F}_r}.\texttt{absorb}(\bar{L}(\zeta\omega))$

   1.1.23 $v = \phi(H_{\mathbb{F}_r}.\texttt{squeeze}())$

   1.1.24 $u = \phi(H_{\mathbb{F}_r}.\texttt{squeeze}())$

   1.1.25 $powers\_of\_evals = [\zeta^{max\_poly\_size}, \zeta_w^{max\_poly\_size}]$

   1.1.26 Compute evaluation of $\eta_i(\zeta), \eta_i(\zeta\omega)$ for $0 \le i < N_{\texttt{prev}}$
   $\eta_i(X) = \sum_{j=0}^{k=\log d+1-1} 1 + \xi_{i,j} \cdot X^{2^j}$

   1.1.27 Compute evaluation of $\bar{L}(\zeta)$[7]

   1.2 Combine (multiply) proof evaluations over the polynomials with $\zeta, \zeta\omega$

   1.3 $\mathbf{f}_{\text{base}} := \{S_{\sigma_{N_{\texttt{perm}}-1},\texttt{comm}}, \texttt{gate}_{\text{mult},\texttt{comm}}, w_{0,\texttt{comm}}, w_{1,\texttt{comm}}, w_{2,\texttt{comm}}, q_{\text{const},\texttt{comm}}, \texttt{gate}_{\text{psdn},\texttt{comm}}, \texttt{gate}_{\text{rc},\texttt{comm}}, \texttt{gate}_{\text{ec\_add},\texttt{comm}}, \texttt{gate}_{\text{ec\_dbl},\texttt{comm}}, \texttt{gate}_{\text{ec\_endo},\texttt{comm}}, \texttt{gate}_{\text{ec\_vbase},\texttt{comm}}\}$

   1.4 $s_{\text{perm}} := (w_0(\zeta) + \gamma + \beta \cdot S_{\sigma_0}(\zeta)) \cdot \ldots \cdot (w_5(\zeta) + \gamma + \beta \cdot S_{\sigma_{N_{\texttt{perm}}}}(\zeta))$

   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 **PE** is a set of elements of the form $(f_{\texttt{comm}}, f(\zeta), f(\zeta\omega))$ for the following polynomials: $\eta_0, \ldots, \eta_{N_{\texttt{prev}}}, pub, w_0, \ldots, w_{N_{\texttt{wires}}}, z, S_{\sigma_0}, \ldots, S_{\sigma_{N_{\texttt{perm}}}}, \bar{L}$

   1.7 $\mathcal{P}_i = \{H_{\mathbb{F}_q}, \zeta, v, u, \mathbf{PE}, o_{\pi_i}\}$

2. $\texttt{batch\_verify\_scalar\_field}(\mathcal{P}_0, \ldots, \mathcal{P}_{\texttt{batch\_size}})$

[7] Details: https://o1-labs.github.io/proof-systems/plonk/maller_15.html

Algorithm 5 Verifier.Base_Field

1. for each π_i:

   1.1 $pub_{\texttt{comm}} = -\texttt{MSM}(\mathbf{L}, \texttt{pub}) \in \mathbb{G}$, where $\mathbf{L}$ is Lagrange bases vector

   1.2 $\texttt{random\_oracle}(p_{\texttt{comm}}, \pi_i)$:

   1.2.1 $H_{\mathbb{F}_q}.\texttt{absorb}(pub_{\texttt{comm}}||w_{0,\texttt{comm}}||...||w_{N_{\text{wires}},\texttt{comm}})$

   1.2.2 $\texttt{joint\_combiner} = H_{\mathbb{F}_q}.\texttt{squeeze}()$ <- PI check

   1.2.3 $H_{\mathbb{F}_q}.\texttt{absorb}(LOOKUP)$ L146, commitments sorted

   1.2.4 $\beta, \gamma = H_{\mathbb{F}_q}.\texttt{squeeze}()$ <- PI check

   1.2.5 $H_{\mathbb{F}_q}.\texttt{absorb}(LOOKUP2)$ L156m commitments aggregated

   1.2.6 $H_{\mathbb{F}_q}.\texttt{absorb}(z_{\texttt{comm}})$

   1.2.7 $\alpha = H_{\mathbb{F}_q}.\texttt{squeeze}()$ <- PI check

   1.2.8 $H_{\mathbb{F}_q}.\texttt{absorb}(t_{1,\texttt{comm}}||...||t_{N_{\texttt{perm}},\texttt{comm}}||...||\infty||)$

   1.2.9 $\zeta = H_{\mathbb{F}_q}.\texttt{squeeze}()$ <- PI check

   1.2.10 Get digest from $H_{\mathbb{F}_q}$ <- PI check

   1.3 $\mathbf{f}_{\text{base}} \coloneqq \{S_{\sigma_{N_{\texttt{perm}}-1},\texttt{comm}}, \texttt{gate}_{\text{mult},\texttt{comm}}, w_{0,\texttt{comm}}, w_{1,\texttt{comm}}, w_{2,\texttt{comm}}, q_{\text{const},\texttt{comm}}, \texttt{gate}_{\text{psdn},\texttt{comm}}, \texttt{gate}_{\text{rc},\texttt{comm}}, \texttt{gate}_{\text{ec\_add},\texttt{comm}}, \texttt{gate}_{\text{ec\_dbl},\texttt{comm}}, \texttt{gate}_{\text{ec\_endo},\texttt{comm}}, \texttt{gate}_{\text{ec\_vbase},\texttt{comm}}\}$

   1.4 $s_{\text{perm}} \coloneqq (w_0(\zeta) + \gamma + \beta \cdot S_{\sigma_0}(\zeta)) \cdot ... \cdot (w_5(\zeta) + \gamma + \beta \cdot S_{\sigma_{N_{\texttt{perm}}}}(\zeta))$

   1.5 $\mathbf{f}_{\text{scalars}} \coloneqq \{-z(\zeta\omega) \cdot \beta \cdot \alpha_0 \cdot zkp(\zeta) \cdot s_{\text{perm}}, w_0(\zeta) \cdot w_1(\zeta), w_0(\zeta), w_1(\zeta), 1$ $s_{\text{psdn}}, s_{\text{rc}}, s_{\text{ec\_add}}, s_{\text{ec\_dbl}}, s_{\text{ec\_endo}}, s_{\text{ec\_vbase}}\}$

   1.6 $f_{\texttt{comm}} = \text{MSM}(\mathbf{f}_{\text{base}}, \mathbf{f}_{\text{scalars}})$

   1.7 Copy from PI $(\zeta^n - 1)$

   1.8 $\bar{L}_{\texttt{comm}} = f_{\texttt{comm}} - t_{\texttt{comm}} \cdot (\zeta^n - 1)$

   1.9 **PE** is a set of elements of the form $(f_{\texttt{comm}}, f(\zeta), f(\zeta\omega))$ for the following polynomials: $\eta_0, \ldots, \eta_{N_{\texttt{prev}}}, pub, w_0, \ldots, w_{N_{\texttt{wires}}}, z, S_{\sigma_0}, \ldots, S_{\sigma_{N_{\texttt{perm}}}}, \bar{L}$

   1.10 $\mathcal{P}_i = \{H_{\mathbb{F}_q}, \zeta, v, u, \mathbf{PE}, o_{\pi_i}\}$

2. $\texttt{batch\_verify\_base\_field}(\mathcal{P}_0, \ldots, \mathcal{P}_{\texttt{batch\_size}})$

Algorithm 6 Batch Verify - Scalar Field

Input: $\pi_0, \ldots, \pi_{\texttt{batch\_size}}$, 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. $\rho_1 \leftarrow \mathbb{F}_r$
2. $\rho_2 \leftarrow \mathbb{F}_r$
3. $r_0 = r'_0 = 1$
4. $\texttt{scalars} = [0, \ldots, 0]$
5. for $0 \leq i < \texttt{batch\_size}$:
   - 5.1 $cip_i = \texttt{combined\_inner\_product}(\zeta_i, \zeta_i\omega, v_i, u_i, \mathbf{PE}_i)$
   - 5.2 Calculate opening challenges $\xi_{i,j}$ from o_{π_i} and copy-constraint them
   - 5.3 Calculate inversion from $\xi_{i,j}$
   - 5.4 Copy c_chal_i from PI
   - 5.5 $c_i = \phi(c\_chal_i)$
   - 5.6 $h_i(X) \coloneqq \prod_{k=0}^{\log(d+1)-1}(1 + \xi_{\log(d+1)-k} X^{2^k})$, where $d = \texttt{lr\_rounds}$
   - 5.7 $b_i = h_i(\zeta) + u_i \cdot h_i(\zeta\omega)$
   - 5.8 $sg_i = -r_i \cdot opening.z1 - r'_i$
   - 5.9 $\texttt{scalars} = \texttt{scalars}||sg_i$
   - 5.10 $\texttt{scalars}[i+1] = \texttt{scalars}[i+1] + \sum_j (s_j \cdot r'_i)$ for s_j from h_i coefficients
   - 5.11 $\texttt{scalars}[0] = \texttt{scalars}[0] - r_i \cdot openings.z2$
   - 5.12 $\texttt{scalars} = \texttt{scalars}||(-r_i \cdot opening.z1 \cdot b0)$
   - 5.13 $r_i = r_{i-1} \cdot \rho_1$
   - 5.14 $r'_i = r'_{i-1} \cdot \rho_2$

Algorithm 7 Batch Verify - Base Field

Input: $\pi_0, \ldots, \pi_{\texttt{batch\_size}}$, 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. $\texttt{bases} = [pk.H, pk.G_0, \ldots, pk.G_n, \mathcal{O}, \ldots, \mathcal{O}]$
2. for $0 \leq i < \texttt{batch\_size}$:
   - 2.1 Get cip_i from PI
   - 2.2 $H_{i,\mathbb{F}_q}.\texttt{absorb}(cip_i - 2^{255})$
   - 2.3 $U_i = (H_{i,\mathbb{F}_q}.\texttt{squeeze}()).\texttt{to\_group}()$
   - 2.4 Calculate opening challenges $\xi_{i,j}$ from o_{π_i}
   - 2.5 $H_{i,\mathbb{F}_q}.\texttt{absorb}(openings.\delta)$
   - 2.6 $h_i(X) \coloneqq \prod_{k=0}^{\log(d+1)-1}(1 + \xi_{\log(d+1)-k} X^{2^k})$, where $d = \texttt{lr\_rounds}$
   - 2.7 $C_i = \sum\limits_j v_i^j (\sum\limits_k r_i^k f_{j,\texttt{comm}})$), where $f_{j,\texttt{comm}}$ from $\mathbf{PE}_i$.
   - 2.8 $Q_i = \sum(\xi_{i,j} \cdot L_{i,j} + \xi_{i,j}^{-1} \cdot R_j) + cip_i \cdot U_i + C_i$
   - 2.9 $c_i = H_{i,\mathbb{F}_q}.\texttt{squeeze}()$ <- PI
   - 2.10 Check $\hat{G}_i = < s, G >$, where s is set of $h(X)$ coefficients.
     Remark: This check can be done inside the `MSM` below using r_i'.
3. Fq: $\texttt{res} = \sum\limits_i r^i (c_i Q_i + delta_i - (z_{i,1}(\hat{G}_i + b_i U_i) + z_{i,2} H))$
4. Fq: `return res` $== 0$

Algorithm 8 Combined Inner Product

Input: $\xi, r, f_0(\zeta_1), \ldots, f_k(\zeta_1), f_0(\zeta_2), \ldots, f_k(\zeta_2)$
Output: s

1. Fr: $s = \sum\limits_{i=0}^{k} \xi^i \cdot (f_i(\zeta_1) + r \cdot f_i(\zeta_2))$

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 constructon 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 trnasaction 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 seqno.
6. In case the verification of a state proof corresponding to the replication packet B_n was completed sucessfully, 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 sucessfully, 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 $C_1, ..., C_M$.
9. In case several branches $C_1, ..., C_M$ 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 successfull transaction from $\{B_{n_1}, ..., B_{n_2}\}$ to the Ethereum-based proof verificator later.

Ouroboros' consensus protocol chain selecton 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, $N = C_1, ..., C_M$ 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 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 $minDen$.
5. return $minDen$

The function $isShortRange(C, C')$ outputs whether or not the chains fork in the "short range" or not.

Algorithm 11 isShortRange(C1, C2)

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

Algorithm 12 maxvalid-sc($C_{loc}, N = C_1, ..., C_M, k$)

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

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.