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)

November 20, 2021

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 (O1Labs' or Chainsafe's protocol implementation).
2. State proof generator.
3. Ethereum RPC proof submitter.
4. EVM-based proof verificator.

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 generilize 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 individualy, it is possible to aggregate them for FRI. For polynomials $f_0, \dots, f_k$:

1. Get θ 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 commited 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:

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

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

---

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

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

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

2. Define new witness polynomials:

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

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

$$p_i = f_i + \beta \cdot S_{id_i} + \gamma$$
$$q_i = f_i + \beta \cdot S_{\sigma_i} + \gamma$$

6. Define:

$$p'(X) = \prod_{0 \le i < N_{\texttt{perm}}} p_i(X) \in \mathbb{F}_{<N_{\texttt{perm}} \cdot n}[X]$$
$$q'(X) = \prod_{0 \le i < N_{\texttt{perm}}} q_i(X) \in \mathbb{F}_{<N_{\texttt{perm}} \cdot n}[X]$$

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

$$P(\omega) = Q(\omega) = 1$$
$$P(\omega^i) = \prod_{1 \le j < i} p'(\omega^i) \text{ for } i \in 2, \ldots, n+1$$
$$Q(\omega^i) = \prod_{1 \le j < i} q'(\omega^i) \text{ for } i \in 2, \ldots, n+1$$

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

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

10. Define polynomials ($F_0, \ldots, F_4$ - copy-satisfability):

$$F_0(X) = L_1(X)(P(X) - 1)$$
$$F_1(X) = L_1(X)(Q(X) - 1)$$
$$F_2(X) = P(X)p'(X) - P(X\omega)$$
$$F_3(X) = Q(X)q'(X) - Q(X\omega)$$
$$F_4(X) = L_n(X)(P(X\omega) - Q(X\omega))$$
$$F_5(X) = \sum_{0 \le i < N_{\texttt{sel}}} (\mathbf{q}_i(X) \cdot \texttt{gate}_i(X)) + \sum_{0 \le i < N_{\texttt{const}}} (\mathbf{f}_{c_i}(X)) + PI(X)$$

11. Compute:

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

12. Split $T(X)$ into separate polynomials $T_0(X), ..., T_{N_{\texttt{perm}}-1}(X)$

13. Add commitments to $T_0(X), ..., T_{N_{\texttt{perm}}-1}(X)$ to transcript.

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

15. Run evaluation scheme with the committed polynomials and y.
Remark: Depending on the circuit, evaluation can be done also on $y\omega, y\omega^{-1}$.

16. The proof is $\pi_{\texttt{comm}}$ and $\pi_{\texttt{eval}}$, where:
    - $\pi_{\texttt{comm}} = \{f_{0,\texttt{comm}}, \ldots, f_{N_{\texttt{wires}}-1,\texttt{comm}}, P_{\texttt{comm}}, Q_{\texttt{comm}}, T_{0,\texttt{comm}}, ..., T_{N_{\texttt{perm}}-1,\texttt{comm}}\}$
    - $\pi_{\texttt{eval}}$ is evaluation proofs for $f_0(y), \ldots, f_{N_{\texttt{wires}}-1}(y), P(y), P(y\omega), Q(y), Q(y\omega), T_0(y), \ldots, T_{N_{\texttt{perm}}-1}(y)$

2.3.2 Verifier View

1. Let $f_{0,\texttt{comm}}, \ldots, f_{N_{\texttt{wires}}-1,\texttt{comm}}$ be commitments to $f_0(X), \ldots, f_{N_{\texttt{wires}}-1}(X)$

2. transcript = setup_values$||f_{0,\texttt{comm}}|| \ldots ||f_{N_{\texttt{wires}}-1,\texttt{comm}}$

3. $\beta, \gamma = hash$(transcript)

4. Let $P_{\texttt{comm}}, Q_{\texttt{comm}}$ be commitments to $P(X), Q(X)$

5. transcript = transcript$||P_{\texttt{comm}}||Q_{\texttt{comm}}$

6. $\alpha_0, \ldots, \alpha_5 = hash$(transcript)

7. Let $T_{0,\mathtt{comm}}, \ldots, T_{N_{\mathtt{perm}}-1,\mathtt{comm}}$ be commitments to $T_0(X), \ldots, T_{N_{\mathtt{perm}}-1}(X)$

8. transcript = transcript$||T_{0,\mathtt{comm}}||\ldots||T_{N_{\mathtt{perm}}-1,\mathtt{comm}}$

9. $y = hash|_{\mathbb{F}/H}(\text{transcript})$

10. 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.

11. Calculate:

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

12. 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[1]

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 $\mathbb{F}_p$[2] and for $\mathbb{F}_q$[3].

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.

Note that for MSM in Step 4:

$$\begin{aligned}
\sum_{i=0}^{2^{n_k}-1} s_i \cdot G_i &= H\\
s_i &\coloneqq \prod_{\substack{0 \le j \le n_k \\ \text{bits}(i)[j]=1}} \phi(c_j),
\end{aligned}$$

where:

- $\phi\colon \{0,1\}^{128} \to \mathbb{F}$ is defined as `to_field` in the implementation[4].
- Given an integer $i < 2^{n_k}$, bits(i) is defined as the little-endian bit array of length n representing the binary expansion of i.
- $G_0, \ldots, G_{2^{n_k-1}} \in \mathbb{G}_k$ is a fixed sequence of group elements[5].
- $c_0, \ldots, c_{n_k-1} \in \{0,1\}^{128}$ is a sequence of challenges.

We use the same 15-wires PLONK circuits that are designed for Mina.[6]

[1] https://hackmd.io/u_2Ygx8XS5Ss1a0bg0FjkA
[2] https://github.com/o1-labs/proof-systems/blob/master/oracle/src/pasta/fp.rs
[3] https://github.com/o1-labs/proof-systems/blob/master/oracle/src/pasta/fq.rs
[4] https://github.com/o1-labs/proof-systems/blob/49f81edc9c86e5907d26ea791fa083640ad0ef3e/oracle/src/sponge.rs#L33
[5] https://github.com/o1-labs/proof-systems/blob/master/dlog/commitment/src/srs.rs#L70
[6] https://o1-labs.github.io/mina-book/specs/15_wires/15_wires.html

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

$N_{\texttt{wires}}$	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: $w_{0,\texttt{comm}}, ..., w_{N_{\texttt{wires}},\texttt{comm}} \in \mathbb{G}$
  - Permutation polynomial: $z_{\texttt{comm}} \in \mathbb{G}$
  - Quotinent polynomial: $t_{\texttt{comm}} = (t_{1,\texttt{comm}}, t_{2,\texttt{comm}}, ..., t_{N_{\texttt{perm}},\texttt{comm}}) \in (\mathbb{G}^{N_{\texttt{perm}}} \times \mathbb{G})$
- Evaluations:
  - $w_0(\zeta), ..., w_{N_{\texttt{wires}}}(\zeta) \in \mathbb{F}_r$
  - $w_0(\zeta\omega), ..., w_{N_{\texttt{wires}}}(\zeta\omega) \in \mathbb{F}_r$
  - $z(\zeta), z(\zeta\omega) \in \mathbb{F}_r$
  - $S_{\sigma_0}(\zeta), ..., S_{\sigma_{N_{\texttt{perm}}}}(\zeta) \in \mathbb{F}_r$
  - $S_{\sigma_0}(\zeta\omega), ..., S_{\sigma_{N_{\texttt{perm}}}}(\zeta\omega) \in \mathbb{F}_r$
  - $\bar{L}(\zeta\omega) \in \mathbb{F}_r$[7]
- 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:

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

- $\{\eta_i(\xi_j)\}_j, \eta_{i,\texttt{comm}}$, for $0 \le i < \texttt{prev}$

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 $pub_{i,\texttt{comm}}$.

Algorithm 1 Verification

Input: $\pi_0, \ldots, \pi_{\texttt{batch\_size}}$ (see 2.5.2)
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 $\texttt{random\_oracle}(p_{\texttt{comm}}, \pi_i)$:

   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 $\beta, \gamma = H_{\mathbb{F}_q}.\texttt{squeeze}()$
   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 $H_{\mathbb{F}_q}.\texttt{absorb}(t_{1,\texttt{comm}}||...||t_{N_{\texttt{perm}},\texttt{comm}}||...||\infty||)$
   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 $0 \le i < N_{\texttt{prev}}$
   1.2.14 Compute evaluation of $\bar{L}(\zeta)$

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

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

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

   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. $\texttt{final\_check}(\mathcal{P}_0, \ldots, \mathcal{P}_{\texttt{batch\_size}})$

Algorithm 2 Final Check

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. $\rho_1 \to \mathbb{F}_r$
2. $\rho_2 \to \mathbb{F}_r$
3. $r_0 = r'_0 = 1$
4. for $0 \leq i <$ `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 $d =$ `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. $\texttt{res} = \sum\limits_i r^i(c_i Q_i + delta_i - (z_{i,1}(\hat{G}_i + b_i U_i) + z_{i,2}H))$
6. `return res` $== 0$

Algorithm 3 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. $s = \sum\limits_{i=0}^{k} \xi^i \cdot (f_i(\zeta_1) + r \cdot f_i(\zeta_2))$

2.6 Elliptic Curve Arithmetic

WIP

2.6.1 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:

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

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	...	...	...	...	...	...	...

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):
  - $(x_2 - x_1) \cdot (y_3 + y_1) - (y_1 - y_2) \cdot (x_1 - x_3)$
  - $(x_1 + x_2 + x_3) \cdot (x_1 - x_3) \cdot (x_1 - x_3) - (y_3 + y_1) \cdot (y_3 + y_1)$
  - $(x_2 - x_1) \cdot r_2 = 1$

2.6.3 Variable Base Scalar Multiplication

For $S = [r]T$, where $r = 2^n + k$ and $k = [k_n...k_0]$, $k_i \in \{0, 1\}$: [8]

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 $z \mod 2 = 0$:

- $b_1 \cdot (b_1 - 1) = 0$
- $b_2 \cdot (b_2 - 1) = 0$
- $(x_P - x_T) \cdot s_1 = y_P - (2b_1 - 1) \cdot y_T$
- $s_1^2 - s_2^2 = x_T - x_R$
- $(2 \cdot x_P + x_T - s_1^2) \cdot (s_1 + s_2) = 2y_P$
- $(x_P - x_R) \cdot s_2 = y_R + y_P$
- $(x_R - x_T) \cdot s_3 = y_R - (2b_2 - 1) \cdot y_T$
- $s_3^2 - s_4^2 = x_T - x_S$
- $(2 \cdot x_R + x_T - s_3^2) \cdot (s_3 + s_4) = 2 \cdot y_R$
- $(x_R - x_S) \cdot s_4 = y_S + y_R$
- $n = 32 \cdot \mathtt{next}(n) + 16 \cdot b_1 + 8 \cdot b_2 + 4 \cdot \mathtt{next}(b_1) + 2 \cdot \mathtt{next}(b_2) + \mathtt{next}(b_3)$

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

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

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

2.6.4 Variable Base Endo-Scalar Multiplication

For $S = [r]T$, where $r = [r_n ... r_0]$ and $r_i \in \{0, 1\}$: [9]

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
$i+1$	s_5	b_3	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
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i+62$	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
$i+63$	s_5	b_3	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:

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

2.6.5 Fixed-base scalar multiplication circuit

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

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	α	β	γ	δ	λ	$-$	$-$	b

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

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 $i \in \{0, .., 5\}$
  - $\phi_1(b_0, b_1, b_2, u_0) = 0$, 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 $i + z$, $z \in 1, ..., 41$:
  - $b_i \cdot (b_i - 1) = 0$, where $i \in \{0, .., 5\}$
  - $\phi_1(b_0, b_1, b_2, u_0) = 0$, 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 $i \in \{0, .., 2\}$
  - $\phi_1(b_0, b_1, b_2, u_0) = 0$, 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: $G_0, ..., G_{k-1} \in \mathbb{G}, s_0, ..., s_{k-1} \in \mathbb{F}_r$, where $\mathbb{F}_r$ is scalar field of $\mathbb{G}$.

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

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 $Q = r_{2i+1} \mathbin{?} \phi([2r_{2i} - 1]T) : [2r_{2i} - 1]T$
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 $Q = k_{i+1} \mathbin{?} T : -T$
        2.3.2 $R = S + Q$
        2.3.3 $S = R + S$

    2.4 $S = k_0 \mathbin{?} S - T : S$

    2.5 $A = A + S$

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

where `sm_rows` is the number of rows in the scalar multiplication circuit.

2.7.2 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 $r \coloneqq s_j$, $T \coloneqq G_j$
        2.1.2 $Q = r_{2i+1} \mathbin{?} \phi([2r_{2i} - 1]T) : [2r_{2i} - 1]T$
        2.1.3 $A = A + Q$

    2.2 if $i \neq 0$:

        2.2.1 $A = 2 \cdot A$

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

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. $A = A + \sum_{j=0}^{k} [1 - s_{j,0}]G_j$

$$\text{rows} \approx \tfrac{2}{5}n \cdot (k \cdot \texttt{add\_rows} + \texttt{dbl\_rows}) + k \approx 103 \cdot (k+1) + 2k \approx 104k + 103,$$

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	$T_{0,0}$	$T_{0,1}$	$T_{0,2}$	$T_{4,0}$	$T_{4,1}$	$T_{4,2}$	$T_{1,0}$	$T_{1,1}$	$T_{1,2}$	$T_{2,0}$	$T_{2,1}$	$T_{2,2}$	$T_{3,0}$	$T_{3,1}$	$T_{3,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i+10$	$T_{50,0}$	$T_{50,1}$	$T_{50,2}$	$T_{54,0}$	$T_{54,1}$	$T_{54,2}$	$T_{51,0}$	$T_{51,1}$	$T_{51,2}$	$T_{52,0}$	$T_{52,1}$	$T_{52,2}$	$T_{53,0}$	$T_{53,1}$	$T_{53,2}$
$i+11$	$T_{55,0}$	$T_{55,1}$	$T_{55,2}$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$

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

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

2.9 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'	$\texttt{acc}$	ξ	$\xi_{\texttt{acc}}$	s_1	s_2	$\xi'_{\texttt{acc}}$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$i+1$	f_3	f_3'	f_4	f_4'	$\texttt{acc}$	r	$\xi_{\texttt{acc}}$	s_1	s_2	$\xi'_{\texttt{acc}}$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i+\lceil \frac{k}{2} \rceil - 1$	f_{k-3}	f_{k-3}'	f_{k-2}	f_{k-2}'	$\texttt{acc}$	ξ	$\xi_{\texttt{acc}}$	s_1	s_2	$\xi'_{\texttt{acc}}$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$i+\lceil \frac{k}{2} \rceil$	f_{k-1}	f_{k-1}'	f_k	f_k'	$\texttt{acc}$	r	$\xi_{\texttt{acc}}$	s_1	s_2	$\xi'_{\texttt{acc}}$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

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

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

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

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

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.

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.