In-EVM Mina State Verification

Circuit Description

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

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

October 19, 2021

1 Introduction

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 63 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:

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

where:

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

WIP

[1] https://hackmd.io/u_2Ygx8XS5Ss1aObgOFjkA
[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

2 Preliminaries

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:

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

2.2 Verification Algorithm

Proof state (here $\mathbb{F}_r$ is a scalar field for $\mathbb{G}$):

- DLog Commitments:
  - $l_{comm}, r_{comm}, o_{comm}, z_{comm} \in \mathbb{G}$ // could each commit contains multiple points?
  - $t_{comm} = (t_{comm,1}, t_{comm,2}) \in (\mathbb{G}^5 \times \mathbb{G})$
- Openings:
  - $(L_i, R_i) \in \mathbb{G} \times \mathbb{G}$ for $0 \leq i <$ `lr_rounds` // vector of rounds of L and R commitments
  - $\delta, SG \in \mathbb{G}$
  - $z_1, z_2 \in \mathbb{F}_r$
- Polynomial Evaluations a, b, for $i = \{1, 2\}$:
  - $l_i, r_i, o_i, z_i, f_i \in \mathbb{F}_r$ // could each eval contains multiple points irl?
  - $t_i \in \mathbb{F}_r^5$
  - $\sigma_{1_i}, \sigma_{2_i} \in \mathbb{F}_r$
- $w \in \mathbb{F}_r^{s_w}$ - witness
- previous challenges:
  - $(c_i, p_i) \in (\mathbb{F}_r \times \mathbb{G})$ for $0 \leq i <$ `prev`

Let g_r, g_q are generators of $\mathbb{F}_r$ and $\mathbb{F}_q$ accordingly.
Verification algorithm:

1. for each $\mathcal{P}$:

   1.1 $p_{comm} = \text{MSM}(\text{lgr\_comm, proof.public}) \in \mathbb{G}$ // public input verification

   1.2 $ORACLES \rightarrow \{\text{digest}, (\beta, \gamma, \alpha', \alpha, \zeta, v, u, \zeta', v', u'), \alpha_2, (pub_1, pub_2), \texttt{evlp}, \texttt{polys}, \zeta_1, \text{combined inner product}\}$:

   1.2.1 $H_{\mathbb{F}_q}.absorb(p_{comm}||l_{comm}||r_{comm}||o_{comm})$

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

   1.2.3 $\gamma = H_{\mathbb{F}_q}.squeeze()$

   1.2.4 $H_{\mathbb{F}_q}.absorb(z_{comm})$

   1.2.5 $\alpha' = H_{\mathbb{F}_q}.squeeze()$

   1.2.6 $\alpha = \phi(\alpha', endo\_r)$

   1.2.7 $H_{\mathbb{F}_q}.absorb(t_{comm,1}||\infty||...||\infty||t_{comm,2})$ // input size?

   1.2.8 $\zeta' = H_{\mathbb{F}_q}.squeeze()$

   1.2.9 $\zeta = \phi(\zeta', endo\_r)$

   1.2.10 digest $= H_{\mathbb{F}_q}.digest()$

   1.2.11 $\zeta_1 = \zeta^n$

   1.2.12 $\zeta_\omega = \zeta * g_r$

1.2.13 $\alpha_2 = [\alpha^2, ..., \alpha^1 9]$

1.2.14 compute Lagrange base evaluation denominators

1.2.15 evaluate public input polynomials (return pub_1, pub_2)

1.2.16 $H_{\mathbb{F}_r}.absorb(pub_1||pub_2)$

1.2.17 $v' = H_{\mathbb{F}_r}.squeeze()$

1.2.18 $v = \phi(v', endo\_r)$

1.2.19 $u' = H_{\mathbb{F}_r}.squeeze()$

1.2.20 $u = \phi(u', endo\_r)$

1.2.21 `elvp` $= \zeta^{\texttt{mpl}}, \zeta_\omega^{\texttt{mpl}}$

1.2.22 `prev_chal_evals`

1.2.23 inner product calculations

1.3 arithmetic operations:

1.3.1 polynomial evaluation over a, b (proof evaluations)

1.3.2 polynomial evaluation over `zkpm` at ζ

1.3.3 `perm_scalars`

1.4 $f_{comm} = \text{MSM}(p, s)$

1.5 linearization polynomial evaluation consistency:

2. srs.verify:

2.1 ...

2.2 MSM:

$$\sum_i r^i(c_i Q_i + delta_i - (z_{1,i}(G_i + b_i U_i) + z_{2,i} H))$$

WIP

3 Multi-Scalar Multiplication Circuit

WIP

4 Poseidon Circuit

4.1 $\mathbb{F}_p$

4.2 $\mathbb{F}_q$

WIP

5 Other Circuits

WIP

6 Bringing it all together

WIP

References