# In-EVM Solana 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 17, 2021

## 1 Introduction

This paper contains a description of the following PLONK-style circuits:

- SHA256 for block headers and transactions hashes verification.
- EdDSA for validators signature verification.
- Poseidon/Reinforce Concrete for state Merkle-tree and proof generation.

## 2 SHA256 Circuit

Suppose that input data in the 32-bits form, which is already padded to the required size. Checking that chunked input data corresponds to the original data out of this circuit. However, we add the boolean check and range proof.

**Range proof that $a < 2^{32}$** Let $a = \{a_0, ..., a_{15}\}$, where $a_i$ is two bits.

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a_{12}$ | $a_{13}$ | $a_{14}$ | $a_{15}$ | acc |
| j + 1 | $a_8$ | $a_9$ | $a_{10}$ | $a_{11}$ | acc |
| j + 2 | $a_4$ | $a_5$ | $a_6$ | $a_7$ | acc |
| j + 3 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | a |

Range gate constraints:

$$w_{1,i}(w_{1,i}-1)(w_{1,i}-2)(w_{1,i}-3) + w_{2,i}(w_{2,i}-1)(w_{2,i}-2)(w_{2,i}-3) + w_{3,i}(w_{3,i}-1)(w_{3,i}-2)(w_{3,i}-3) + w_{4,i}(w_{4,i}-1)(w_{4,i}-2)(w_{4,i}-3)$$
$$w_{o,i} = w_{o,i-1} * 4^4 + w_{4,i} * 4^3 + w_{3,i} * 4^2 + w_{2,i} * 4 + w_{1,i}$$

The range proofs are included for each input data block.

**The function $\sigma_0$** contain sparse mapping subcircuit with base 2. Let $a$ be divided to $a_0, a_1, a_2, a_3$ 8 bits-chunks. The values $a'_0, a'_1, a'_2, a'_3$ are in sparse form, and $a'$ is a sparse $a$. We need the following lookup tables:

1. **SHA-256 NORMALIZE2**: Read $a'_i$ to $a_i$
2. **SHA-256 8ROT3 32**: Read $a'_1$ to $r_1$
3. **SHA-256 8ROT2 32**: Read $a'_4$ to $r_2$
4. **SHA-256 8SHR3 32**: Read $a'_0$ to $r_3$

|  | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | a |
| j + 1 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | acc |
| j + 2 | $r1$ | $r_2$ | $r_3$ |  | $\sigma_0$ |

Sparse map gate constraints:

$$w_{o,j} = w_{1,j} + w_{2,j} * 2^8 + w_{3,j} * 2^{8*2} + w_{4,j} * 2^{8*3}$$
$$w_{o,j+1} = w_{2,j+1} * 4^{8-7} + w_{3,j+1} * 4^{8*2-7} + w_{4,j+1} * 4^{8*3-7} + w_{1,j+1} * 4^{8*2-2} + w_{2,j+1} * 4^{8*3-2} + w_{4,j+1} * 4^{8-2} + w_{2,j+1} * 4^{8-3} + w_{3,j+1} * 4^{8*2-3} + w_{4,j+1} * 4^{8^3-3}$$
$$w_{o,j+2} = w_{0,j+1} + w_{1,j+2} + w_{2,j+2} + w_{3,j+2}$$
7 plookup constraints

**The function $\sigma_1$** contain sparse mapping subcircuit with base 2. Let $a$ be divided to $a_0, a_1, a_2, a_3$ 8 bits-chunks. The values $a'_0, a'_1, a'_2, a'_3$ are in sparse form and $a'$ is a sparse $a$. We need the following lookup tables:

1. **SHA-256 NORMALIZE2**: Read $a_i$ to $a'_i$
2. **SHA-256 8ROT1 32**: Read $a'_2$ to $r_1$
3. **SHA-256 8ROT3 32**: Read $a'_2$ to $r_2$
4. **SHA-256 8ROT2 32**: Read $a'_1$ to $r_3$

|  | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | a |
| j + 1 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | acc |
| j + 2 | $r1$ | $r_2$ | $r_3$ |  | $\sigma_1$ |

Sparse map gate constraints:

$$w_{o,j} = w_{1,j} + w_{2,j} * 2^8 + w_{3,j} * 2^{8*2} + w_{4,j} * 2^{8*3}$$
$$w_{o,j+1} = w_{1,j+1} * 4^{8*2-1} + w_{2,j+1} * 4^{8*3-1} + w_{4,j+1} * 4^{8-1} + w_{1,j+1} * 4^{8*2-3} + w_{2,j+1} * 4^{8*3-3} + w_{4,j+1} * 4^{8-3} + w_{1,j+1} * 4^{8^3-2} + w_{3,j+1} * 4^{8-2} + w_{4,j+1} * 4^{8^2-2}$$
$$w_{o,j+2} = w_{0,j+1} + w_{1,j+2} + w_{2,j+2} + w_{3,j+2}$$
7 plookup constraints

The sparse values $\sigma_0$ and $\sigma_1$ have to be normalized. The final addition requires one add gate. We use **SHA256 NORMALIZE2**

|  | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ |  |
| j + 1 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $\sigma_i$ |

Normalize gate constraints:

$$w_{o,j-1} = w_{4,j} * 4^8 * 3 + w_{3,j} * 4^8 * 2 + w_{2,j} * 4^8 + w_{1,j}$$
$$w_{o,j+1} = w_{4,j+1} * 256^3 + w_{3,j+1} * 256^2 + w_{2,j+1} * 256 + w_{1,j+1}$$ 4 plookup constraints

**The $\Sigma_0$ function** contain sparse mapping subcircuit with base 2. Let $a$ be divided to $a_0, a_1, a_2, a_3$ 8 bits-chunks. The values $a'_0, a'_1, a'_2, a'_3$ are in sparse form, and $a'$ is a sparse $a$. We need the following lookup tables:

1. **SHA-256 NORMALIZE2**: Read $a_i$ to $a'_i$
2. **SHA-256 8ROT2 32**: Read $a'_0$ to $r_1$
3. **SHA-256 8ROT5 32**: Read $a'_1$ to $r_2$
4. **SHA-256 8ROT6 32**: Read $a'_2$ to $r_3$

|  | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | a |
| j + 1 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | a' |
| j + 2 | $r1$ | $r_2$ | $r_3$ |  | $\Sigma_0$ |

Sparse map gate constraints:

$$w_{o,j} = w_{1,j} + w_{2,j} * 2^8 + w_{3,j} * 2^{8*2} + w_{4,j} * 2^{8*3}$$
$$w_{o,j+1} = w_{1,j+1} + w_{2,j+1} * 4^8 + w_{3,j+1} * 4^{8*2} + w_{4,j+1} * 4^{8*3}$$
$$w_{o,j+2} = w_{2,j+1} * 4^{8-2} + w_{3,j+1} * 4^{8*2-2} + w_{4,j+1} * 4^{8*3-2} + w_{1,j+1} * 4^{8*3-5} + w_{3,j+1} * 4^{8-5} + w_{4,j+1} * 4^{8*2-5} + w_{1,j+1} * 4^{8*2-6} + w_{2,j+1} * 4^{8*3-6} + w_{4,j+1} * 4^{8-6} + w_{1,j+2} + w_{2,j+2} + w_{3,j+2}$$

7 plookup constraints

**The $\Sigma_1$ function** contain sparse mapping subcircuit with base 2. Let $a$ be divided to $a_0, a_1, a_2, a_3$ 8 bits-chunks. The values $a'_0, a'_1, a'_2, a'_3$ are in sparse form, and $a'$ is a sparse $a$. We need the following lookup tables:

1. **SHA-256 NORMALIZE7**: Read $a_i$ to $a'_i$
2. **SHA-256 8ROT6 32**: Read $a'_0$ to $r_1$
3. **SHA-256 8ROT3 32**: Read $a'_1$ to $r_2$
4. **SHA-256 8ROT1 32**: Read $a'_3$ to $r_3$

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | a |
| j + 1 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | a' |
| j + 2 | $r1$ | $r_2$ | $r_3$ | | $\Sigma_1$ |

Sparse map gate constraints:

$$w_{o,j} = w_{1,j} + w_{2,j} * 2^8 + w_{3,j} * 2^{8*2} + w_{4,j} * 2^{8*3}$$
$$w_{o,j+1} = w_{1,j+1} + w_{2,j+1} * 7^8 + w_{3,j+1} * 7^{8*2} + w_{4,j+1} * 7^{8*3}$$
$$w_{o,j+2} = w_{2,j+1} * 7^{8-6} + w_{3,j+1} * 7^{8*2-6} + w_{7,j+1} * 4^{8*3-6} + w_{1,j+1} * 7^{8*3-3} + w_{3,j+1} * 7^{8-3} + w_{4,j+1} * 7^{8*2-3} + w_{1,j+1} * 7^{8-1} + w_{2,j+1} * 7^{8*2-1} + w_{3,j+1} * 7^{8*3-1} + w_{1,j+2} + w_{2,j+2} + w_{3,j+2}$$

7 plookup constraints

The sparse values $\Sigma_0$ and $\Sigma_1$ have to be normalized. We use **SHA256 NORMALIZE7**

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | |
| j + 1 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $\Sigma_i$ |

Normalize gate constraints:

$$w_{o,j-1} = w_{4,j} * 4^8 * 3 + w_{3,j} * 4^8 * 2 + w_{2,j} * 4^8 + w_{1,j} \text{ for } \Sigma_1 \text{ replace 4 with 7}$$
$$w_{o,i} = w_{4,i} * 256^3 + w_{3,i} * 256^2 + w_{2,i} * 256 + w_{1,i}$$

7 plookup constraints

**The Maj function** contain sparse mapping subcircuit with base 2 for $a, b, c$. Let $a; b; c$ be divided to $a_0, a_1, a_2, a_3; b_0, b_1, b_2, b_3; c_0, c_1, c_2, c_3$ 8 bits-chunks. The values $a'_0, a'_1, a'_2, a'_3$ are in sparse form, and $a'$ is a sparse $a$. Similarly for b and c. Note, that $a$ we already have in the sparse from $\Sigma_0$ in the circuit. The variables $b$ and $c$ were represented in sparse form in the previous rounds or it is public inputs.

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j - k | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | a' |
| ... | | | | | |
| j - l | $b'_0$ | $b'_1$ | $b'_2$ | $b'_3$ b' | |
| ... | | | | | |
| j - t | $c'_0$ | $c'_1$ | $c'_2$ | $c'_3$ | c' |
| ... | | | | | |
| j + 0 | a' | b' | c' | | maj |

Sparse map gate constraints:

$$w_{o,j} = w_{1,j} + w_{2,j} + w_{3,j}$$

The sparse values $maj$ have to be normalized. We use **SHA256 MAJ NORMALIZE2**

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | |
| j + 1 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $maj$ |

Normalize gate constraints:

$$w_{o,i} = w_{4,i} * 256^3 + w_{3,i} * 256^2 + w_{2,i} * 8 + w_{1,i}$$

The final addition requires one add gate.

**The Ch function** contain sparse mapping subcircuit with base 2 for $e, f, g$. Let $e; f; g$ be divided to $e_0, e_1, e_2, e_3; f_0, f_1, f_2, f_3; g_0, g_1, g_2, g_3$ 8 bits-chunks. The values $e'_0, e'_1, e'_2, e'_3$ are in sparse form, and $e'$ is a sparse $e$. Similarly for b and c. Note, that $e$ we already have in the sparse from $\Sigma_0$ in the circuit. The variables $f$ and $g$ were represented in sparse form in the previous rounds or it is public inputs.

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j - k | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | a' |
| ... | | | | | |
| j - l | $b'_0$ | $b'_1$ | $b'_2$ | $b'_3$ b' | |
| ... | | | | | |
| j - t | $c'_0$ | $c'_1$ | $c'_2$ | $c'_3$ | c' |
| ... | | | | | |
| j + 0 | a' | b' | c' | | ch |

Sparse map gate constraints:

$$w_{o,j} = w_{1,j} + 2 * w_{2,j} + 3 * w_{3,j}$$

The sparse values $ch$ have to be normalized. We use **SHA256 CH NORMALIZE7**

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | |
| j + 1 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $ch$ |

Normalize gate constraints:

$$w_{o,i} = w_{4,i} * 256^3 + w_{3,i} * 256^2 + w_{2,i} * 8 + w_{1,i}$$

The final addition requires one add gate.
The updating of variables for new rounds costs 10 add gates.
Producing the final hash value costs two add gates.

## 2.1 SHA-512

SHA-512 uses the similar logical functions as in refsha256 which operates on 64-bits words. Thus each input uses the same range proof which extended to 64-bits.

**Range proof that $a < 2^{64}$** Let $a = \{a_0, ..., a_{32}\}$, where $a_i$ is two bits.

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a_{29}$ | $a_{30}$ | $a_{31}$ | $a_{32}$ | acc |
| j + 1 | $a_{25}$ | $a_{26}$ | $a_{27}$ | $a_{28}$ | acc |
| ... | | | | | |
| j + 6 | $a_4$ | $a_5$ | $a_6$ | $a_7$ | acc |
| j + 7 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | a |

Range gate constraints:

$$w_{1,i}(w_{1,i} - 1)(w_{1,i} - 2)(w_{1,i} - 3) + w_{2,i}(w_{2,i} - 1)(w_{2,i} - 2)(w_{2,i} - 3) + w_{3,i}(w_{3,i} - 1)(w_{3,i} - 2)(w_{3,i} - 3) + w_{4,i}(w_{4,i} - 1)(w_{4,i} - 2)(w_{4,i} - 3)$$

$$w_{o,i} = w_{o,i-1} * 4^4 + w_{4,i} * 4^3 + w_{3,i} * 4^2 + w_{2,i} * 4 + w_{1,i}$$

The range proofs are included for each input data block.

**The function $\sigma_0$** contain sparse mapping subcircuit with base 2. Let $a$ be divided to $a_0, a_1, a_2, ..., a_7$ 8 bits-chunks. The values $a'_0, a'_1, a'_2, ..., a'_7$ are in sparse form, and $a'$ is a sparse $a$. We need the following lookup tables:

1. **SHA-256 NORMALIZE2**: Read $a_i$ to $a'_i$
2. **SHA-256 8ROT1 64**: Read $a'_0$ to $r_1$
3. **SHA-256 8SHR7 64**: Read $a'_0$ to $r_3$

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
| j + 1 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | a |
| j + 2 | $a_5$ | $a_6$ | $a_7$ | $a'_4$ | $\sigma_0$ |
| j + 3 | $a'_5$ | $a'_6$ | $a'_7$ | $r_1$ | $r_2$ |

Sparse map gate constraints:

$$w_{o,j+1} = w_{1,j} + w_{2,j} * 2^8 + w_{3,j} * 2^{8*2} + w_{4,j} * 2^{8*3} + w_{o,j} * 2^{8*4} + w_{1,j+2} * 2^{8*5} + w_{2,j+2} * 2^{8*6} + w_{3,j+2} * 2^{8*7}$$
$$w_{o,j+2} = w_{2,j+1} * 4^{8-1} + w_{3,j+1} * 4^{8*2-1} + w_{4,j+1} * 4^{8*3-1} + w_{4,j+2} * 4^{8*4-1} + w_{1,j+3} * 4^{8*5-1} + w_{2,j+3} * 4^{8*6-1} + w_{3,j+3} * 4^{8*7-1} + w_{1,j+1} * 4^{8*7} + w_{2,j+1} + w_{3,j+1} * 4^{8} + w_{4,j+1} * 4^{8*2} + w_{4,j+2} * 4^{8*3} + w_{1,j+3} * 4^{8*4} + w_{2,j+3} * 4^{8*5} + w_{3,j+3} * 4^{8*6} + w_{2,j+1} * 4^{8-7} + w_{3,j+1} * 4^{8*2-7} + w_{4,j+1} * 4^{8*3-7} + w_{4,j+2} * 4^{8*4-7} + w_{1,j+3} * 4^{8*5-7} + w_{2,j+3} * 4^{8*6-7} + w_{3,j+3} * 4^{8*7-7} + w_{4,j+3} + w_{o,j+3}$$

10 plookup constraints

**The function $\sigma_1$** contain sparse mapping subcircuit with base 2. Let $a$ be divided to $a_0, a_1, a_2, ..., a_7$ 8 bits-chunks. The values $a'_0, a'_1, a'_2, ..., a'_7$ are in sparse form, and $a'$ is a sparse $a$. We need the following lookup tables:

1. **SHA-256 NORMALIZE2**: Read $a_i$ to $a'_i$
2. **SHA-256 8ROT3 64**: Read $a'_2$ to $r_1$
3. **SHA-256 8ROT5 SHR6 64**: Read $a'_7 + a'_0$ to $r_2$

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
| j + 1 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | a |
| j + 2 | $a_5$ | $a_6$ | $a_7$ | $a'_4$ | $\sigma_1$ |
| j + 3 | $a'_5$ | $a'_6$ | $a'_7$ | $r_1$ | $r_2$ |

Sparse map gate constraints:

$$w_{o,j+1} = w_{1,j} + w_{2,j} * 2^8 + w_{3,j} * 2^{8*2} + w_{4,j} * 2^{8*3} + w_{o,j} * 2^{8*4} + w_{1,j+2} * 2^{8*5} + w_{2,j+2} * 2^{8*6} + w_{3,j+2} * 2^{8*7}$$
$$w_{o,j+2} = w_{1,j+1} * 4^{64-19} + w_{2,j+1} * 4^{64+(8-19)} + w_{4,j+1} * 4^{8*3-19} + w_{4,j+2} * 4^{8*4-19} + w_{1,j+3} * 4^{8*5-19} + w_{2,j+3} * 4^{8*6-19} + w_{3,j+3} * 4^{8*7-19} + w_{1,j+1} * 4^{64-61)} + w_{2,j+1} * 4^{64+(8-61)} + w_{3,j+1} * 4^{64+(8*2-61)} + w_{4,j+1} * 4^{64+(8*3-61)} + w_{4,j+2} * 4^{64+(8*4-61)} + w_{1,j+3} * 4^{64+(8*5-61)} + w_{2,j+3} * 4^{64+(8*6-61)} + w_{2,j+1} * 4^{8-6} + w_{3,j+1} * 4^{8*2-6} + w_{4,j+1} * 4^{8*3-6} + w_{4,j+2} * 4^{8*4-6} + w_{1,j+3} * 4^{8*5-6} + w_{2,j+3} * 4^{8*6-6} + w_{3,j+3} * 4^{8*7-6} + w_{4,j+3} + w_{o,j+3}$$

10 plookup constraints

The sparse values $\sigma_0$ and $\sigma_1$ have to be normalized. The final addition requires one add gate. Note, that $a'$ already initialized in the row $j-2$. We use **SHA256 NORMALIZE2**

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | $acc$ |
| j + 1 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | 0 |
| j + 2 | $a'_4$ | $a'_5$ | $a'_6$ | $a'_7$ | $\sigma_i$ |
| j + 3 | $a_4$ | $a_5$ | $a_6$ | $a_7$ | |

Normalize gate constraints:

$$w_{o,j+1} = w_{4,j+1} * 256^3 + w_{3,j+1} * 256^2 + w_{2,j+1} * 256 + w_{1,j+1} + w_{1,j+3} * 256^4 + w_{2,j+3} * 256^5 + w_{3,j+3} * 256^6 + w_{4,j+4} * 256^7$$
$$w_{o,j} = w_{o,j-2} - (w_{4,j} * 256^3 + w_{3,j} * 256^2 + w_{2,j} * 256 + w_{1,j})$$
$$w_{o,j+1} = w_{o,j} - (w_{1,j+3} * 256^4 + w_{2,j+3} * 256^5 + w_{3,j+3} * 256^6 + w_{4,j+4} * 256^7)$$

8 plookup constraints

**The $\Sigma_0$ function** contain sparse mapping subcircuit with base 2. Let $a$ be divided to $a_0, a_1, a_2, a_3$ 7-bits chunks and $a_4, a_5, a_6, a_7$ 9 bits-chunks. The values $a'_0, a'_1, a'_2, ..., a'_7$ are in sparse form, and $a'$ is a sparse $a$. We need the following lookup tables:

1. **SHA-256 9NORMALIZE2**: Read $a_i$ to $a'_i$
2. **SHA-256 7NORMALIZE2**: Read $a_i$ to $a'_i$
3. **SHA-256 9ROT6 32**: Read $a'_4$ to $r_2$
4. **SHA-256 9ROT2 32**: Read $a'_5$ to $r_3$

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
| j + 1 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | a |
| j + 2 | $a_5$ | $a_6$ | $a_7$ | $a'_4$ | $\Sigma_0$ |
| j + 3 | $a'_5$ | $a'_6$ | $a'_7$ | $r_1$ | $r_2$ |

Sparse map gate constraints:

$$w_{o,j+1} =$$
$$w_{1,j} + w_{2,j} * 2^7 + w_{3,j} * 2^{7*2} + w_{4,j} * 2^{7*3} + w_{o,j} * 2^{7*4} + w_{1,j+2} * 2^{7*4+9} + w_{2,j+2} * 2^{7*4+9*2} + w_{3,j+2} * 2^{7*4+9*3}$$
$$w_{o,j+2} = w_{4,j+2} + w_{1,j+3} * 4^9 + w_{2,j+3} * 4^{9*2} + w_{3,j+3} * 4^{9*3} + w_{1,j+1} * 4^{9*4} + w_{2,j+1} * 4^{9*4+7} + w_{3,j+1} * 4^{9*4+7*2} + w_{4,j+1} * 4^{9*4+7*3} + w_{1,j+1} * 4^{64-34)} + w_{2,j+1} * 4^{64+(7-34)} + w_{3,j+1} * 4^{64+(7*2-34)} + w_{4,j+1} * 4^{64+(7*3-34)} + w_{1,j+3} * 4^{64+(7*4+9-34)} + w_{2,j+3} * 4^{64+(7*4+9*2-34)} + w_{3,j+3} * 4^{64+(7*4+9*3-34)} + w_{1,j+1} * 4^{64-39)} + w_{2,j+1} * 4^{64+(7-39)} + w_{3,j+1} * 4^{64+(7*2-39)} + w_{4,j+1} * 4^{64+(7*3-39)} + w_{4,j+2} * 4^{64+(7*4-39)} + w_{2,j+3} * 4^{64+(7*4+9*2-39)} + w_{3,j+3} * 4^{64+(7*4+9*3-39)} + w_{4,j+3} + w_{o,j+3}$$

10 plookup constraints

**The $\Sigma_1$ function** contain sparse mapping subcircuit with base 2. Let $a$ be divided to $a_0, a_1, a_2, a_3$ 7-bits chunks and $a_4, a_5, a_6, a_7$ 9 bits-chunks. The values $a'_0, a'_1, a'_2, ..., a'_7$ are in sparse form, and $a'$ is a sparse $a$. We need the following lookup tables:

1. **SHA-256 9NORMALIZE2**: Read $a_i$ to $a'_i$
2. **SHA-256 7NORMALIZE2**: Read $a_i$ to $a'_i$
3. **SHA-256 7ROT4 32**: Read $a'_2$ to $r_2$
4. **SHA-256 9ROT4 32**: Read $a'_5$ to $r_3$

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
| j + 1 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | a |
| j + 2 | $a_5$ | $a_6$ | $a_7$ | $a'_4$ | $\Sigma_0$ |
| j + 3 | $a'_5$ | $a'_6$ | $a'_7$ | $r_1$ | $r_2$ |

Sparse map gate constraints:

$$w_{o,j+1} =$$
$$w_{1,j} + w_{2,j} * 2^7 + w_{3,j} * 2^{7*2} + w_{4,j} * 2^{7*3} + w_{o,j} * 2^{7*4} + w_{1,j+2} * 2^{7*4+9} + w_{2,j+2} * 2^{7*4+9*2} + w_{3,j+2} * 2^{7*4+9*3}$$
$$w_{o,j+2} = w_{3,j+1} + w_{4,j+1} * 7^7 + w_{4,j+2} * 7^{7*2} + w_{1,j+3} * 7^{7*2+9} + w_{2,j+3} * 7^{7*2+9*2} + w_{3,j+3} * 7^{9*3+7*2} + w_{1,j+1} * 7^{9*4+7*2} + w_{2,j+1} * 7^{9*4+7*3} + w_{1,j+1} * 7^{64-18)} + w_{2,j+1} * 7^{64+(7-18)} + w_{4,j+1} * 7^{64+(7*4-18)} + + w_{4,j+2} * 7^{64+(7*4+9-18)} + w_{1,j+3} * 7^{64+(7*4+9*2-18)} + w_{2,j+3} * 7^{64+(7*4+9*3-18)} + w_{3,j+3} * 7^{64+(7*4+9*-18)} + w_{1,j+1} * 7^{64-41)} + w_{2,j+1} * 7^{64+(7-41)} + w_{3,j+1} * 7^{64+(7*2-41)} + w_{4,j+1} * 7^{64+(7*3-41)} + w_{4,j+2} * 7^{64+(7*3+9-41)} + w_{2,j+3} * 7^{64+(7*3+9*2-41)} + w_{3,j+3} * 7^{64+(7*3+9*3-41)} + w_{4,j+3} + w_{o,j+3}$$

10 plookup constraints

The sparse values $\Sigma_0$ and $\Sigma_1$ have to be normalized. We use **SHA256 NORMALIZE7** Note, that $a'$ already initialized in the row $j-2$.

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | $a'$ |
| j + 1 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $acc$ |
| j + 2 | $a'_4$ | $a'_5$ | $a'_6$ | $a'_7$ | $\sigma_i$ |
| j + 3 | $a_4$ | $a_5$ | $a_6$ | $a_7$ | |

Normalize gate constraints:

$$w_{o,j+1} =$$
$$w_{4,j+1}*256^3+w_{3,j+1}*256^2+w_{2,j+1}*256+w_{1,j+1}+w_{1,j+3}*256^4+w_{2,j+3}*256^5+w_{3,j+3}*256^6+w_{4,j+4}*256^7$$
$$w_{o,j} = w_{1,j-3} + w_{2,j-3} * 4^7 + w_{3,j-3} * 4^{7*2} + +w_{4,j-3} * 4^{7*3} + w_{4,j-2} * 4^{7*4} + w_{1,j-1} * 7^{7*4+9} + w_{2,j-1} * 7^{7*4+9*2} + w_{2,j-1} * 7^{7*4+9*3}$$ for maj or ch function. for $\Sigma_1$ replace 4 with 7
$$w_{o,j} = w_{o,j-2} - (w_{4,j} * 256^3 + w_{3,j} * 256^2 + w_{2,j} * 256 + w_{1,j})$$
$$w_{o,j+1} = w_{o,j} - (w_{1,j+3} * 256^4 + w_{2,j+3} * 256^5 + w_{3,j+3} * 256^6 + w_{4,j+4} * 256^7)$$
8 plookup constraints

**The Maj function** contain sparse mapping subcircuit with base 2 for $a, b, c$. Note, that the sparse chunks of $a$ we already have in $\Sigma_0$ in the circuit. The variables $b$ and $c$ were represented in sparse chunks in the previous rounds or it is public inputs.

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j | $a'$ | $b'$ | $c'$ | | maj |

Sparse map gate constraints:

$$w_{o,j} = w_{1,j} + w_{2,j} + w_{3,j}$$

The sparse values *maj* have to be normalized. We use **SHA256 MAJ NORMALIZE2** Note, that *maj* already initialized in the row $j - 1$.

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | $acc$ |
| j + 1 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | 0 |
| j + 2 | $a'_4$ | $a'_5$ | $a'_6$ | $a'_7$ | $maj$ |
| j + 3 | $a_4$ | $a_5$ | $a_6$ | $a_7$ | |

Normalize gate constraints:

$$w_{o,j+1} =$$
$$w_{4,j+1}*256^3+w_{3,j+1}*256^2+w_{2,j+1}*256+w_{1,j+1}+w_{1,j+3}*256^4+w_{2,j+3}*256^5+w_{3,j+3}*256^6+w_{4,j+4}*256^7$$
$$w_{o,j} = w_{o,j-1} - (w_{4,j} * 256^3 + w_{3,j} * 256^2 + w_{2,j} * 256 + w_{1,j})$$
$$w_{o,j+1} = w_{o,j} - (w_{1,j+3} * 256^4 + w_{2,j+3} * 256^5 + w_{3,j+3} * 256^6 + w_{4,j+4} * 256^7)$$
8 plookup constraints

The final addition requires one add gate.

**The Ch function** contain sparse mapping subcircuit with base 2 for $e, f, g$. Note, that $e$ we already have in the sparse from $\Sigma_1$ in the circuit. The variables $f$ and $g$ were represented in sparse form in the previous rounds or it is public inputs.

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | e' | f' | g' | | ch |

Sparse map gate constraints:

$$w_{o,j} = w_{1,j} + 2 * w_{2,j} + 3 * w_{3,j}$$

The sparse values *ch* have to be normalized. Note, that *ch* already initialized in the row $j - 1$. We use **SHA256 CH NORMALIZE7**

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_o$ |
|---|---|---|---|---|---|
| j + 0 | $a'_0$ | $a'_1$ | $a'_2$ | $a'_3$ | $acc$ |
| j + 1 | $a_0$ | $a_1$ | $a_2$ | $a_3$ | 0 |
| j + 2 | $a'_4$ | $a'_5$ | $a'_6$ | $a'_7$ | $ch$ |
| j + 3 | $a_4$ | $a_5$ | $a_6$ | $a_7$ | |

Normalize gate constraints:

$$w_{o,j+1} =$$
$$w_{4,j+1}*256^3+w_{3,j+1}*256^2+w_{2,j+1}*256+w_{1,j+1}+w_{1,j+3}*256^4+w_{2,j+3}*256^5+w_{3,j+3}*256^6+w_{4,j+4}*256^7$$
$$w_{o,j} = w_{o,j-1} - (w_{4,j} * 256^3 + w_{3,j} * 256^2 + w_{2,j} * 256 + w_{1,j})$$
$$w_{o,j+1} = w_{o,j} - (w_{1,j+3} * 256^4 + w_{2,j+3} * 256^5 + w_{3,j+3} * 256^6 + w_{4,j+4} * 256^7)$$
8 plookup constraints

The final addition requires one add gate.
The updating of variables for new rounds costs 10 add gates.
Producing the final hash value costs two add gates.

## 3 Poseidon Circuit

WIP
For now, there are two SNARK-friendly hash function candidates: Poseidon[1] and Reinforce Concrete[2].

## 4 Merkle Tree Circuit

Merkle Tree generation for set $\{H_{B_{n_1}}, ..., H_{B_{n_2}}\}$. Let $k = \lceil \log(n_2 - n_1) \rceil$

```
1. $n = n_2 - n_1$
2. $2^k = n$
3. for $i$ from 0 to $n - 1$:
    3.1 $T_i = H_i$ // just notation for simplicity, not a real part of the circuit
4. for $i$ from 0 to $k - 1$:
    4.1 for $j$ from 0 to $(n - 1)/2$:
        4.1.1 $T_i' = \texttt{hash}(T_{2 \cdot i}, T_{2 \cdot i + 1})$. // see Section 3
    4.2 $n = \frac{n}{2}$
    4.3 for $j$ from 0 to $n - 1$:
        4.3.1 $T_i = T_i'$. // just notation for simplicity, not a real part of the circuit
```

## 5 Ed25519 Circuit

To verify a signature $(R, s)$ on message $M$ using public key $A$ and a generator $B$ do:

1. Prove that $s$ in the defined range.
2. $k ==$ SHA-512$(data||R||A||M)$ //See section 2.1
3. $8sB? =?8R + 8kA$ //See section 5.1

### 5.1 The Arithmetic of Elliptic Curves

Variable-base scalar multiplication circuit per bit $b$:

1. $b^2 = b$
2. $(y_1)(2b - 1) = (y_2)$
3. $(x_2 - x_3)(\lambda_1) = (y_2 - y_3)$
4. $(B\lambda_1)(\lambda_1) = (A + x_3 + x_2 + x_4)$
5. $(x_3 - x_4)(\lambda_1 + \lambda_2) = (2y_3)$
6. $(B\lambda_2)(\lambda_2) = (A + x_4 + x_3 + x_5)$
7. $(x_3 - x_5)(\lambda_2) = (y_5 + y_3)$

EC Point Addition circuit:

1. $(x_2 - x_1)(y_3 + y_1) - (y_1 - y_2)(x_1 - x_3)$
2. $(x_1 + x_2 + x_3)(x_1 - x_3)(x_1 - x_3) - (y_3 + y_1)(y_3 + y_1)$

[1] https://www.poseidon-hash.info
[2] https://www.rc-hash.info

## 6 The Correct Validator Set Proof Circuit

WIP

## 7 Bringing it all together

Let bank-hashes of proving block set be $\{H_{B_{n_1}}, ..., H_{B_{n_2}}\}$. The last confirmed block is $H_{B_L}$. Each positively confirmed block is signed by $M$ validators.

Denote by `block_data` the data that is included in the bank hash other than the bank hash of the parent block.

```
1. $H_{B_{n_1}} = H_{B_L}$ // $H_{B_L}$ is a public input
2. Validator set constraints. // see Section 6
3. for $i$ from $n_1 + 1$ to $n_2 + 32$:
    3.1 $H_{B_i} = \texttt{sha256}(\texttt{block\_data} || H_{B_{i-1}})$ // see Section 2
4. for $j$ from 0 to $M$:
    4.1 Ed25519 constraints for $H_{B_{n_2+32}}$ // see Section 5
5. Merkle tree constraints for the set $\{H_{B_{n_1}}, ..., H_{B_{n_2}}\}$ // see Section 4
```

## References