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)

November 13, 2021

1 Introduction

This paper contains a description of PLONK-style circuits for Crypto3's In-EVM Solana "Light Client" State Verification[1].

In this section, we present a high-level overview of the verification circuit. In the following sections, we provide details on the sub-circuits.

1.1 Verification Circuit Overview

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

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.

[1] https://blog.nil.foundation/2021/10/14/solana-ethereum-bridge.html

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:

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

The range proofs are included for each input data block.

The function σ_0 contain sparse mapping subcircuit with base 2. Let a be divided to 8 bits-chunks a_0, a_1, a_2, a_3. 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		σ_0

Sparse map gate constraints:

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

The function σ_1 contain sparse mapping subcircuit with base 2. Let a be divided to 8 bits-chunks a_0, a_1, a_2, a_3. 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		σ_1

Sparse map gate constraints:

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

The sparse values σ_0 and σ_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	σ_i

Normalize gate constraints:

$$w_{o,j-1} = w_{4,j} \cdot 4^8 \cdot 3 + w_{3,j} \cdot 4^8 \cdot 2 + w_{2,j} \cdot 4^8 + w_{1,j}$$
$$w_{o,j+1} = w_{4,j+1} \cdot 256^3 + w_{3,j+1} \cdot 256^2 + w_{2,j+1} \cdot 256 + w_{1,j+1}$$

4 plookup constraints

The Σ_0 function contain sparse mapping subcircuit with base 2. Let a be divided to 8 bits-chunks a_0, a_1, a_2, a_3. 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		Σ_0

Sparse map gate constraints:

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

7 plookup constraints

The Σ_1 function contain sparse mapping subcircuit with base 2. Let a be divided to 8 bits-chunks a_0, a_1, a_2, a_3. 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		Σ_1

Sparse map gate constraints:

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

7 plookup constraints

The sparse values Σ_0 and Σ_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	Σ_i

Normalize gate constraints:

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

The Maj function contain sparse mapping subcircuit with base 2 for a, b, c. Let $a; b; c$ be divided to 8 bits-chunks $a_0, a_1, a_2, a_3; b_0, b_1, b_2, b_3; c_0, c_1, c_2, c_3$. 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 Σ_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:

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

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 8 bits-chunks $e_0, e_1, e_2, e_3; f_0, f_1, f_2, f_3; g_0, g_1, g_2, g_3$. 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 Σ_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:

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

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:

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

The range proofs are included for each input data block.

The function σ_0 contain sparse mapping subcircuit with base 2. Let a be divided to 8 bits-chunks $a_0, a_1, a_2, ..., a_7$. 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-512 8ROT1 64**: Read a'_0 to r_1
3. **SHA-512 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	σ_0
j + 3	a'_5	a'_6	a'_7	r_1	r_2

Sparse map gate constraints:

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

10 plookup constraints

The function σ_1 contain sparse mapping subcircuit with base 2. Let a be divided to 8 bits-chunks $a_0, a_1, a_2, ..., a_7$. 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-512 8ROT3 64**: Read a'_2 to r_1
3. **SHA-512 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	σ_1
j + 3	a'_5	a'_6	a'_7	r_1	r_2

Sparse map gate constraints:

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

10 plookup constraints

The sparse values σ_0 and σ_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	σ_i
j + 3	a_4	a_5	a_6	a_7	

Normalize gate constraints:

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

8 plookup constraints

The Σ_0 function contain sparse mapping subcircuit with base 2. Let a be divided to 7-bits chunks a_0, a_1, a_2, a_3 and 9 bits-chunks a_4, a_5, a_6, a_7. 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-512 9NORMALIZE2**: Read a_i to a'_i
2. **SHA-512 7NORMALIZE2**: Read a_i to a'_i
3. **SHA-512 9ROT6 64**: Read a'_4 to r_2
4. **SHA-512 9ROT2 64**: 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	Σ_0
j + 3	a'_5	a'_6	a'_7	r_1	r_2

Sparse map gate constraints:

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

10 plookup constraints

The Σ_1 function contain sparse mapping subcircuit with base 2. Let a be divided to 7-bits chunks a_0, a_1, a_2, a_3 and 9 bits-chunks a_4, a_5, a_6, a_7. 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-512 9NORMALIZE2**: Read a_i to a'_i
2. **SHA-512 7NORMALIZE2**: Read a_i to a'_i
3. **SHA-512 7ROT4 32**: Read a'_2 to r_2
4. **SHA-512 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	Σ_1
j + 3	a'_5	a'_6	a'_7	r_1	r_2

Sparse map gate constraints:

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

10 plookup constraints

The sparse values Σ_0 and Σ_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	σ_i
j + 3	a_4	a_5	a_6	a_7	

Normalize gate constraints:

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

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 Σ_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:

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

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 Σ_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:

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

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:

$$\begin{gathered} w_{o,j+1} = w_{4,j+1} \cdot 256^3 + w_{3,j+1} \cdot 256^2 + w_{2,j+1} \cdot 256 + w_{1,j+1} + w_{1,j+3} \cdot 256^4 + w_{2,j+3} \cdot 256^5 \\ + w_{3,j+3} \cdot 256^6 + w_{4,j+4} \cdot 256^7 \\ w_{o,j} = w_{o,j-1} - (w_{4,j} \cdot 256^3 + w_{3,j} \cdot 256^2 + w_{2,j} \cdot 256 + w_{1,j}) \\ w_{o,j+1} = w_{o,j} - (w_{1,j+3} \cdot 256^4 + w_{2,j+3} \cdot 256^5 + w_{3,j+3} \cdot 256^6 + w_{4,j+4} \cdot 256^7) \\ \text{8 plookup constraints} \end{gathered}$$

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

Consider a poseidon permutation $F : [0_{\mathbb{F}}, I[2], I[3]] \to [O[1], H, O[3]]$ of width 3 and $\alpha = 5$. The 1-call sponge function is used:

	w_1	w_2	w_3	w_4	w_o
j + 0	$0_{\mathbb{F}}$	$I[2]$	$I[3]$	$T_{1,0}$	$T_{1,1}$
j + 1	$T_{1,2}$	$T_{2,0}$	$T_{2,1}$	$T_{2,2}$	$T_{3,0}$
...					
j + 39	$O[1]$	H	$O[3]$	$-$	$-$

Constraints:

$$\begin{gathered} \text{For 4 rounds:} \\ [w_{4,j}, w_{o,j}, w_{1,j+1}] = [w_{1,j}^5, w_{2,j}^5, w_{3,j}^5] \times M + RC \\ \text{For 57 rounds:} \\ [w_{1,j+4}, w_{2,j+4}, w_{3,j+4}] = [w_{3,j+3}, w_{4,j+3}, w_{o,j+3}^5] \times M + RC \\ \text{For 4 rounds:} \\ [w_{2,j+37}, w_{3,j+37}, w_{4,j+37}] = [w_{4,j+36}^5, w_{o,j+36}^5, w_{1,j+37}^5] \times M + RC \end{gathered}$$

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 a message M using public key A and a generator B do:

1. Prove that s in the range $L = 2^{252} + 27742317777372353535851937790883648493$.

	w_1	w_2	w_3	w_4	w_o
j + 0	s	z_0	z_1	z_2	z_3
...					
j + 5	z_{25}			$--$	

Constraints:

$$w_{2,j} = w_{1,j} + 2^{253} + L$$

Each $w_{i,k} - 2^{10} \cdot w_{next}$, where $i = 2, .., o$ for $k = 0$ and $i = 1, .., o$ for $k = 1, .., 4$ is range-constrained by 10-bits plookup table.

$w_{1,j+5} \cdot 2^7$ is range-constrained by 10-bits plookup table.

2. $k == \text{SHA-512}(data||R||A||M)$ // See section 2.1

3. $sB? =?R + kA$:

    3.1 Fixed-base scalar multiplication circuit is used for $sB = S$

    3.2 One addition is used for $S + (-R)$. The coordinates of R and $T = S + (-R)$ are placed on the last row of fixed-base scalar multiplication circuit. In total, three constraints are used for addition:

$$x_t \cdot (1 + dx_s \cdot (-x_r) \cdot y_s \cdot y_r) = x_s \cdot y_r + (-x_r) \cdot y_s$$
$$y_t \cdot (1 - dx_s \cdot (-x_r) \cdot y_s \cdot y_r) = x_s \cdot (-x_r) + y_r \cdot y_s$$
$$-x_r^2 + y_r^2 = 1 - d \cdot x_r^2 \cdot y_r^2$$

    3.3 Variable-base scalar multiplication circuit has to be used in reversed order, where $(x_n, y_n) = (x_t, y_t)$.

5.1 Arithmetic of Elliptic Curves

WIP

This section instantiates the arithmetic of edwards25519 curve:

$$-x^2 + y^2 = 1 - (121665/121666) \cdot x^2 \cdot y^2$$

Affine coordinates are used for points. Let d be equal to 121665/121666.

Fixed-base scalar multiplication circuit : We precompute all values $w(\mathbb{G}, s, k) = k_i \cdot 8^s \mathbb{G}$, where $k_i \in \{0, ..7\}$, $s \in \{0, .., 84\}$.

	w_1	w_2	w_3	w_4	w_o
j + 0	b_{n-1}	b_{n-2}	b_{n-3}	u_1	acc
j + 1	x_2	y_2	b_{n-6}	u_2	v_1
j + 2	b_{n-4}	b_{n-5}	v_2	b_{n-7}	acc
j + 3	x_3	y_3	b_{n-8}	b_{n-9}	u_3
j + 4	x_4	y_4	v_3	$-$	acc
...					
j + 84	$-$	$-$	v_{85}	$-$	$-$

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. $\phi_2 : (x_1, x_2, x_3, x_4, x_5) \mapsto$
$x_3 \cdot (-v'_0 \cdot x_2 \cdot x_1 + v'_0 \cdot x_1 + v'_0 \cdot x_2 - v'_0 + v'_2 \cdot x_1 \cdot x_2 - v'_2 \cdot x_2 + v'_4 \cdot x_1 \cdot x_2 - v'_4 \cdot x_2 - v'_6 \cdot x_1 \cdot x_2 + v'_1 \cdot x_2 \cdot x_1 - v'_1 \cdot x_1 - v'_1 \cdot x_2 + v'_1 - v'_3 \cdot x_1 \cdot x_2 + v'_3 \cdot x_2 - v'_5 \cdot x_1 \cdot x_2 + v'_5 \cdot x_2 + v'_7 \cdot x_1 \cdot x_2) - (x_5 - v'_0 \cdot x_2 \cdot x_1 + v'_0 \cdot x_1 + v'_0 \cdot x_2 - v'_0 + v'_2 \cdot x_1 \cdot x_2 - v'_2 \cdot x_2 + v'_4 \cdot x_1 \cdot x_2 - v'_4 \cdot x_2 - v'_6 \cdot x_1 \cdot x_2)$

3. $\phi_3 : (x_1, x_2, x_3, x_4, x_5, x_6) \mapsto$
$x_1 \cdot (1 + d \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6 - (x_3 \cdot x_6 + x_4 \cdot x_5)$

4. $\phi_4 : (x_1, x_2, x_3, x_4, x_5, x_6) \mapsto$
$x_2 \cdot (1 - d \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6 - (x_3 \cdot x_5 + x_4 \cdot x_6)$

Constraints:

- For $j + 0$:
  - $w_{o,j} = w_{1,j} \cdot 2^2 + w_{2,j} \cdot 2 + w_{3,j}$
  - $\phi_3(w_{1,j+1}, w_{2,j+1}, w_{4,j}, w_{o,j+1}, w_{4,j+1}, w_{3,j+2}) = 0$
  - $\phi_4(w_{1,j+1}, w_{2,j+1}, w_{4,j}, w_{o,j+1}, w_{4,j+1}, w_{3,j+2}) = 0$
- For $j + z$, $z \equiv 0 \mod 5$, $z \neq 0$:
  - $w_{o,j+z} = w_{1,j+z} \cdot 2^2 + w_{2,j+z} \cdot 2 + w_{3,j+z} + w_{o,j+z-1} \cdot 2^3$
  - $\phi_1(w_{1,j+z}, w_{2,j+z}, w_{3,j+z}, w_{4,j+z}) = 0$, where $(u'_i, v'_i) = w(\mathbb{G}, 3 \cdot (\frac{z}{5}), i)$
  - $\phi_2(w_{1,j+z}, w_{2,j+z}, w_{3,j+z}, w_{4,j+z}, w_{o,j+z+1}) = 0$, where $(u'_i, v'_i) = w(\mathbb{G}, 3 \cdot (\frac{z}{5}), i)$
  - $\phi_3(w_{1,j+z+1}, w_{2,j+z+1}, w_{1,j+z-1}, w_{2,j+z-1}, w_{4,j+z+1}, w_{3,j+z+2}) = 0$
  - $\phi_4(w_{1,j+z+1}, w_{2,j+z+1}, w_{1,j+z-1}, w_{2,j+z-1}, w_{4,j+z+1}, w_{3,j+z+2}) = 0$
- For $j + z$, $z \equiv 2 \mod 5$:
  - $w_{o,j+z} = w_{1,j+z} \cdot 2^2 + w_{2,j+z} \cdot 2 + w_{3,j+z-1} + w_{o,j+z-2} \cdot 2^3$
  - $\phi_1(w_{1,j+z}, w_{2,j+z}, w_{3,j+z-1}, w_{4,j+z-1}) = 0$, where $(u'_i, v'_i) = w(\mathbb{G}, 3 \cdot (\frac{z-2}{5}) + 1, i)$
  - $\phi_2(w_{1,j+z}, w_{2,j+z}, w_{3,j+z-1}, w_{4,j+z-1}, w_{3,j+z}) = 0$, where $(u'_i, v'_i) = w(\mathbb{G}, 3 \cdot (\frac{z-2}{5}) + 1, i)$
  - $\phi_3(w_{1,j+z+1}, w_{2,j+z+1}, w_{1,j+z-1}, w_{2,j+z-1}, w_{o,j+z+1}, w_{3,j+z+2}) = 0$
  - $\phi_4(w_{1,j+z+1}, w_{2,j+z+1}, w_{1,j+z-1}, w_{2,j+z-1}, w_{o,j+z+1}, w_{3,j+z+2}) = 0$
- For $j + z$, $z \equiv 3 \mod 5$:
  - $\phi_1(w_{4,j+z-1}, w_{3,j+z}, w_{4,j+z}, w_{o,j+z}) = 0$, where $(u'_i, v'_i) = w(\mathbb{G}, 3 \cdot (\frac{z-3}{5}) + 2, i)$
  - $\phi_2(w_{4,j+z-1}, w_{3,j+z}, w_{4,j+z}, w_{o,j+z}, w_{3,j+z+1}) = 0$, where $(u'_i, v'_i) = w(\mathbb{G}, 3 \cdot (\frac{z-3}{5}) + 2, i)$
- For $j + z$, $z \equiv 4 \mod 5$:
  - $w_{o,j+z} = w_{4,j+z-2} \cdot 2^2 + w_{3,j+z-3} \cdot 2 + w_{4,j+z-3} + w_{o,j+z-2} \cdot 2^3$
  - $\phi_3(w_{1,j+z-2}, w_{2,j+z}, w_{1,j+z-1}, w_{2,j+z-1}, w_{4,j+z+1}, w_{o,j+z+2}) = 0$
  - $\phi_4(w_{1,j+z-2}, w_{2,j+z}, w_{1,j+z-1}, w_{2,j+z-1}, w_{4,j+z+1}, w_{o,j+z+2}) = 0$

Variable-base scalar multiplication circuit :

	w_1	w_2	w_3	w_4	w_o
j + 0	b_{n-1}	x_2	y_2	b_{n-2}	acc
j + 1	x_3	y_3	x_4	b_{n-3}	acc
j + 2	x_1	y_1	y_4	b_{n-4}	acc
j + 3	x_5	y_5	x_6	b_{n-5}	acc
j + 4	y_6	x_7	y_7	b_{n-6}	acc
...					
j + 210	...	x_{n-3}	y_{n-3}	b_2	b
j + 211	x_{n-2}	y_{n-2}	b_1	b_0	x_{n-1}
j + 212	x_1	y_1	y_{n-1}	x_n	y_n

Define the following functions:

1. $\phi_1 : (b, x_1, y_1, x_2, y_2, x_3) \mapsto$
   $x_3 \cdot ((y_1^2 - x_1^2) \cdot (2 - y_1^2 + x_1^2) + 2dx_1y_1(y_1^2 + x_1^2) \cdot x_2y_2b) - (2x_1y_1 \cdot (2 - y_1^2 + x_1^2) \cdot y_2b \cdot (1 - b) + (y_1^2 + x_1^2) \cdot (y_1^2 - x_1^2) \cdot x_2b)$

2. $\phi_2 : (b, x_1, y_1, x_2, y_2, y_3) \mapsto$
   $y_3 \cdot ((y_1^2 - x_1^2) \cdot (2 - y_1^2 + x_1^2) - 2dx_1y_1(y_1^2 + x_1^2) \cdot x_2y_2b) - (2x_1y_1 \cdot (2 - y_1^2 + x_1^2) \cdot x_2b + (y_1^2 + x_1^2) \cdot (y_1^2 - x_1^2) \cdot y_2b \cdot (1 - b))$

Constraints:

- For $j + 0$:
  - $w_{o,j} = w_{1,j} \cdot 2 + w_{4,j}$
  - $\phi_1(w_{1,j+0}, w_{1,j+2}, w_{2,j+2}, w_{1,j+2}, w_{2,j+2}, w_{2,j+0})$
  - $\phi_2(w_{1,j+0}, w_{1,j+2}, w_{2,j+2}, w_{1,j+2}, w_{2,j+2}, w_{3,j+0})$
- For $j + z$, $z \equiv 0 \mod 5$, $z \neq 0$:
  - $w_{o,j+z} = w_{1,j+z} \cdot 2 + w_{4,j+z} + w_{o,j+z-1}$
  - $\phi_1(w_{4,j+z}, w_{2,j+z-1}, w_{3,j+z-1}, w_{1,j+z+2}, w_{2,j+z+2}, w_{2,j+z})$
  - $\phi_2(w_{4,j+z}, w_{2,j+z-1}, w_{3,j+z-1}, w_{1,j+z+2}, w_{2,j+z+2}, w_{3,j+z})$
- For $j + z$, $z \equiv 1 \mod 5$:
  - $w_{o,j+z} = 2 \cdot w_{o,j+z-1} + w_{4,j+z}$
  - $\phi_1(w_{4,j+z-1}, w_{2,j+z-1}, w_{3,j+z-1}, w_{1,j+z+1}, w_{2,j+z+1}, w_{1,j+z})$
  - $\phi_2(w_{4,j+z-1}, w_{2,j+z-1}, w_{3,j+z-1}, w_{1,j+z+1}, w_{2,j+z+1}, w_{2,j+z})$
  - $\phi_1(w_{4,j+z}, w_{1,j+z}, w_{2,j+z}, w_{1,j+z+1}, w_{2,j+z+1}, w_{3,j+z})$
- For $j + z$, $z \equiv 2 \mod 5$:
  - $w_{o,j+z} = 2 \cdot w_{o,j+z-1} + w_{4,j+z}$
  - $\phi_2(w_{4,j+z-1}, w_{1,j+z-1}, w_{2,j+z-1}, w_{1,j+z}, w_{2,j+z}, w_{3,j+z})$
- For $j + z$, $z \equiv 3 \mod 5$:
  - $w_{o,j+z} = 2 \cdot w_{o,j+z-1} + w_{4,j+z}$
  - $w_{o,j+z} = 2 \cdot w_{o,j+z-1} + w_{4,j+z}$
  - $\phi_1(w_{4,j+z-1}, w_{3,j+z-2}, w_{3,j+z-1}, w_{1,j+z-1}, w_{2,j+z-1}, w_{1,j+z})$
  - $\phi_2(w_{4,j+z-1}, w_{3,j+z-2}, w_{3,j+z-1}, w_{1,j+z-1}, w_{2,j+z-1}, w_{2,j+z})$
  - $\phi_1(w_{4,j+z}, w_{1,j+z}, w_{2,j+z}, w_{1,j+z-1}, w_{2,j+z-1}, w_{3,j+z})$
- For $j + z$, $z \equiv 4 \mod 5$:
  - $w_{o,j+z} = 2 \cdot w_{o,j+z-1} + w_{4,j+z}$
  - $\phi_2(w_{4,j+z-1}, w_{1,j+z-1}, w_{2,j+z-1}, w_{1,j+z-2}, w_{2,j+z-2}, w_{1,j+z})$
  - $\phi_1(w_{4,j+z}, w_{3,j+z-1}, w_{1,j+z}, w_{1,j+z-2}, w_{2,j+z-2}, w_{2,j+z})$
  - $\phi_2(w_{4,j+z}, w_{3,j+z-1}, w_{1,j+z}, w_{1,j+z-2}, w_{2,j+z-2}, w_{3,j+z})$

6 The Correct Validator Set Proof Circuit

WIP

References