In-EVM Solana 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)

February 22, 2022

Contents

Chapter 1

Introduction

This document is a technical reference to the in-EVM Solana's 'Light-Client' state verification project.

1.1 Overview

The project's purpose is to provide Ethereum users with reliable Solana's cluster state and necessary transactions proof.

The project UX consists of several steps:

1. Retrieve Solana's 'Light-Client' state.
2. Generate a proof for it.
3. Submit the proof to EVM-enabled cluster.
4. Verify the proof with EVM.

Such a UX defines projects parts:

1. Solana's 'Light-Client' state retriever.
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 Solana's 'Light-Client' state proof generator. Crucial components which define this part design and performance are:

1. Input data format ('Light-Client' state data structure).
2. Proof system used for the proof generation.
3. Circuit definition used for the proof system.

2.1 'Light-Client' State

Block Information $\bar{B}_k$ is defined as follows:

- k - the number of the block
- $B_k = H(B_{k-1}||\texttt{account\_hash}||\texttt{signature\_count\_buf}||b_k||\texttt{validators\_state})$ - bank hash of the block[1]
- b_k Merkle Block
- B_{k-1} - the previous block's bank hash
- `validators_state` is not implemented for now.

Proof algorithm input is defined as follows:

- n_1 - current confirmed block number
- n_2 - new confirmed block number
- $\{\bar{B}_{n_1}, \ldots, \bar{B}_{n_2}, \ldots, \bar{B}_{n_2+32}\}$ - block information for blocks from n_1 to $n_2 + 32$.
- $\sigma_0, \ldots, \sigma_N$ - signatures for B_{n_2+32}

Approximate code representation of such a state data structure is as follows:

```
template<typename Hash>
struct block_data {
    typedef typename Hash::digest_type digest_type;

    std::size_t block_number;
    digest_type bank_hash;
    digest_type merkle_hash;
    digest_type previous_bank_hash;
    //    std::vector<vote_state> votes;
};

template<typename Hash, typename SignatureSchemeType>
struct state_type {
    typedef Hash hash_type;
    typedef SignatureSchemeType signature_scheme_type;
    typedef typename signature_scheme_type::signature_type signature_type;
```

[1] See https://docs.solana.com/proposals/simple-payment-and-state-verification#block-headers

```
    std::size_t n_1 confirmed;
    std::size_t n_2 new_confirmed;
    std::vector<block_data<hash_type>> repl_data;
    std::vector<signature_type> signatures;
};
```

Validator state-representing data structure (`vote_state`) supposes such a state to begin being handled by Solana replication protocol (or its implementation) for handling the tracking of votes state being unchanged 'till the end of epoch.

2.2 Proof System

WIP

The proof system used for proving Solana's 'Light-Client' state on EVM is Redshift SNARK[1]. RedShift is a transparent SNARK that uses PLONK[2] proof system but replaces the commitment scheme. Initial paper proposal is to employ 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, the reader gets referred to [1].

The original RedShift protocol utilizes the classic PLONK[2] system. To provide better performance, the original protocol is generalized to be used with PLONK with custom gates [4], [5] and lookup arguments [6], [7].

2.3 Optimizations

WIP

2.3.1 Batched FRI

Instead of check each commitment individualy, we can 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.

2.3.2 Hash By Column

Instead of committing each of the polynomials, we can use the same Merkle tree for several polynomials. It decreases the number of Merkle tree paths that need to be provided by the prover.

See [8], [1] for details.

2.3.3 Hash By Subset

On the each $i+1$ FRI round, the prover should 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.

2.4 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
$N_{\texttt{lookups}}$	Number of lookups
$\mathbf{f}_i$	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 $Z(X) = \prod a \in H^*(X - a)$
8. Let A_i be a witness lookup columns and S_i be a table columns, $i = 0, .., m$.

Preprocessing:

2.4.1 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 $\theta \in \mathbb{F}$ from $hash$(transcript)
5. Construct the witness lookup compression and table compression $S(\theta)$ and $A(\theta)$:

$$A(\theta) = \theta^{m-1}A_0 + \theta^{m-2}A_1 + ... + \theta A_{m-2} + A_{m-1}$$
$$S(\theta) = \theta^{m-1}S_0 + \theta^{m-2}S_1 + ... + \theta S_{m-2} + S_{m-1}$$

6. Produce the permutation polynomials $S'(X)$ and $A'(X)$ such that:

6.1 All the cells of column A' are arranged so that like-valued cells are vertically adjacent to each other.

6.2 The first row in a sequence of values in A' is the row that has the corresponding value in S'.

7. Compute and add commitments to A' and S' to transcript

8. Get $\beta, \gamma \in \mathbb{F}$ from $hash$(transcript)

9. For $0 \le i < N_{\texttt{perm}}$

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

10. Define:

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

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

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

12. Compute and add commitments to P and Q to transcript

13. Compute permutation product column:

$$V(\omega^i) = \frac{(\theta^{m-1} A_0(\omega^i) + \theta^{m-2} A_1(\omega^i) + \ldots + \theta A_{m-2}(\omega^i) + A_{m-1}(\omega^i) + \beta) \cdot (\theta^{m-1} S_0(\omega^i) + \theta^{m-2} S_1(\omega^i) + \ldots + \theta S_{m-2}(\omega^i) + S_{m-1}(\omega^i) + \gamma)}{(A'(\omega^i) + \beta)(S'(\omega^i) + \gamma)}$$
$$V(1) = V(\omega^{N_{\texttt{lookups}}}) = 1$$

14. Compute and add commitments to V to transcript

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

16. Get τ from $hash$(transcript)

17. Define polynomials ($F_0, \ldots, F_4$ - copy-satisfability, $\texttt{gate}_0$ is PI-constraining gate):

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

18. For the lookup:

18.1 Two selectors q_{last} and q_{blind} are used, where $q_{last} = 1$ for t last blinding rows and $q_{blind} = 1$ on the row in between the usable rows and the blinding rows.

18.2 $F_6(X) = L_0(X)(1 - V(X))$

18.3 $F_7(X) = q_{last} \cdot (V(X)^2 - V(X))$

18.4 $F_8(X) = (1 - (q_{last} + q_{blind})) \cdot (V(\omega X)(A'(X) + \beta)(S'(X) + \gamma) - V(X)(\theta^{m-1} A_0(X) + \ldots + A_{m-1}(X) + \beta)(\theta^{m-1} S_0(X) + \ldots + S_{m-1}(X) + \gamma))$

18.5 $F_9(X) = L_0(X) \cdot (A'(X) - S'(X))$

18.6 $F_{10}(X) = (1 - (q_{last} + q_{blind})) \cdot (A'(X) - S'(X)) \cdot (A'(X) - A'(\omega^{-1} X))$

19. Compute:

$$F(X) = \sum_{i=0}^{10} \alpha_i F_i(X)$$
$$T(X) = \frac{F(X)}{Z(X)}$$

20. $N_T \coloneqq \mathtt{max}(N_{\mathtt{perm}}, \mathtt{deg}_{\mathtt{gates}} - 1)$, where $\mathtt{deg}_{\mathtt{gates}}$ is the highest degree of the degrees of gate polynomials.

21. Split $T(X)$ into separate polynomials $T_0(X), ..., T_{N_T-1}(X)$[2]

22. Add commitments to $T_0(X), ..., T_{N_T-1}(X)$ to transcript

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

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

25. The proof is $\pi_{\mathtt{comm}}$ and $\pi_{\mathtt{eval}}$,, where:

    - $\pi_{\mathtt{comm}} = \{f_{0,\mathtt{comm}}, \dots, f_{N_{\mathtt{wires}-1},\mathtt{comm}}, P_{\mathtt{comm}}, Q_{\mathtt{comm}}, T_{0,\mathtt{comm}}, ..., T_{N_T-1,\mathtt{comm}}, A'_{\mathtt{comm}}, S'_{\mathtt{comm}}, V_{\mathtt{comm}}\}$
    - $\pi_{\mathtt{eval}}$ is evaluation proofs for $f_0(y), \dots, f_{N_{\mathtt{wires}}}(y), P(y), P(y\omega), Q(y), Q(y\omega)$, $T_0(y), \dots, T_{N_T-1}(y), A'(y), A'(y\omega^{-1}), S'(y), V(y), V(y\omega)$

2.4.2 Verifier View

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

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

3. $\theta = hash$(transcript)

4. Let $A'_{\mathtt{comm}}, S'_{\mathtt{comm}}$ be commitments to $A'(X), S'(X)$.

5. transcript = transcript$||A'_{\mathtt{comm}}||S'_{\mathtt{comm}}$

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

7. Let $P_{\mathtt{comm}}, Q_{\mathtt{comm}}, V_{i,\mathtt{comm}}$ be commitments to $P(X), Q(X), V(X)$.

8. transcript = transcript$||P_{\mathtt{comm}}||Q_{\mathtt{comm}}||V_{\mathtt{comm}}$

9. $\alpha_0, \dots, \alpha_5 = hash$(transcript)

10. $\tau = hash$(transcript)

11. $N_T \coloneqq \mathtt{max}(N_{\mathtt{perm}}, \mathtt{deg}_{\mathtt{gates}} - 1)$, where $\mathtt{deg}_{\mathtt{gates}}$ is the highest degree of the degrees of gate polynomials.

12. Let $T_{0,\mathtt{comm}}, ..., T_{N_T-1,\mathtt{comm}}$ be commitments to $T_0(X), ..., T_{N_T-1}(X)$

13. transcript = transcript$||T_{0,\mathtt{comm}}||...||T_{N_T-1,\mathtt{comm}}$

14. $y = hash_{\mathbb{F}/H}$(transcript)

15. Run evaluation scheme verification with the committed polynomials and y to get values $f_i(y), P(y), P(y\omega), Q(y), Q(y\omega), T_j(y), A'(y), S'(y), V(y), A'(y\omega^{-1}), V(y\omega)$.
Remark: Depending on the circuit, evaluation can be done also on $f_i(y\omega), f_i(y\omega^{-1})$ for some i.

16. Calculate:

$$F_0(y) = L_1(y)(P(y) - 1)$$
$$F_1(y) = L_1(y)(Q(y) - 1)$$
$$p'(y) = \prod p_i(y) = \prod f_i(y) + \beta \cdot S_{id_i}(y) + \gamma$$
$$F_2(y) = P(y)p'(y) - P(y\omega)$$
$$q'(y) = \prod q_i(y) = \prod f_i(y) + \beta \cdot S_{\sigma_i}(y) + \gamma$$
$$F_3(y) = Q(y)q'(y) - Q(y\omega)$$

[2]Commit scheme supposes that polynomials should be degree $\leq n$

$$F_4(y) = L_n(y)(P(y\omega) - Q(y\omega))$$
$$F_5(y) = \sum_{0 \leq i < N_{\texttt{sel}}} (\tau^i \cdot \mathbf{q}_i(y) \cdot \texttt{gate}_i(y)) + PI(y)$$
$$T(y) = \sum_{0 \leq j < N_T} y^{n \cdot j} T_j(y) \; F_6(y) = L_0(y)(1 - V(y))$$
$$F_7(y) = q_{last} \cdot (V(y)^2 - V(y))$$
$$F_8(y) = (1 - (q_{last} + q_{blind})) \cdot (V(\omega y)(A'(y) + \beta)(S'(y) + \gamma) - V(y)(\theta^{m-1} A_0(y) + ... + A_{m-1}(y) + \beta)(\theta^{m-1} S_{i,0}(y) + ... + S_{m-1}(y) + \gamma))$$
$$F_9(y) = L_0(y) \cdot (A'(y) - S'(y))$$
$$F_{10}(y) = (1 - (q_{last} + q_{blind})) \cdot (A'(y) - S'(y)) \cdot (A'(y) - A'(\omega^{-1} y))$$

17. Check the identity:

$$\sum_{i=0}^{10} \alpha_i F_i(y) = Z(y)T(y)$$

2.5 Circuit Definition

This section contains a description of PLONK-style circuits for In-EVM Solana's "Light Client" state verification[3].

This section provides a high-level overview of the circuit used for proof generation and verification. Following sections provide sub-circuits details.

2.5.1 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 2.5.9
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 2.5.2
4. for j from 0 to M:
   4.1 Ed25519 constraints for $H_{B_{n_2+32}}$ // see Section 2.5.6
5. Merkle tree constraints for the set $\{H_{B_{n_1}}, ..., H_{B_{n_2}}\}$ // see Section 2.5.5

Suppose that $M = 800$ and $n_2 - n_1 = 3600$. Thus, the total amount of rows is: $3 \cdot 3632 \cdot 755 + 800 \cdot 2839 + 3600 \cdot 22 = 8226480 + 2271200 + 79200 = 10576880$

2.5.2 SHA-256 Circuit

Suppose that input data is in the 32-bits form, which is already padded to the required size. We suppose that the checking that chunked input data corresponds to the original data out of the circuit. However, we do not need to range constrain these chunks as we get them for free from the SHA-256 circuit.

Thus, the preprocessing constraints for the SHA-256 circuit is a decomposition of k message blocks to 32 bits chunks without range proofs. For 'Solana-EVM' circuit, $k = 3$.

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

Lookup tables We use the following lookup tables:

1. **SHA-256 NORMALIZE4** with 2 columns and 2^{14} rows. The first column contains all possible 14-bits words. The second column contains corresponding sparse representations with base 4. The constraints can be used for the range check and sparse representation simultaneously.

2. **SHA-256 NORMALIZE7** with 2 columns and 2^{14} rows. The first column contains all possible 14-bits words. The second column contains corresponding sparse representations with base 7. The constraints can be used for the range check and sparse representation simultaneously.

3. **SHA-256 NORMALIZE MAJ** with 2 columns and 2^8 rows. The first column contains all possible 8-bits words. The second column contains corresponding sparse representations with base 4.

4. **SHA-256 NORMALIZE CH** with 2 columns and 2^8 rows. The first column contains all possible 8-bits words. The second column contains corresponding sparse representations with base 7.

Message scheduling For each block of 512 bits of the padded message the 64 words are constructed in the following way:

- The first 16 words are obtained by splitting the message.
- The last 48 words are obtained by using the functions σ_0, σ_1:

$$W_i = \sigma_1(W_{i-2}) \oplus W_{i-7} \oplus \sigma_0(W_{i-15}) \oplus W_{i-16} \tag{2.1}$$

Each round of the message scheduling has the following table:

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
j + 0	a	a_0	a_1	a_2	a_3	$\hat{a}_1$	$\hat{a}_2$	a'_0	
j + 1	W_i	W_j	a'_1	a'_2	a'_3	s'_0	s'_1	s'_2	s'_3
j + 2	w	s_0	s_1	s_2	s_3	s_0	s_1	s_2	s_3
j + 3		b'_0	b'_1	b'_2	b'_3	s'_0	s'_1	s'_2	s'_3
j + 4	b	b_0	b_1	b_2	b_3	$\hat{b}_0$	$\hat{b}_1$	$\hat{b}_3$	

Evaluations:

Let b be W_{i-2} and a be W_{i-15} from 2.1. The values W_i and W_j in the table corresponds to W_{i-7} and W_{i-16} respectively from 2.1. From the round $r = 2$ the copy constraints are used for values b and w from round $r - 2$. The copy constraints for W_{i-7}, W_{i-15} and W_{i-16} are used in a similar way. The output of round W_i from 2.1 is w.

The first 16 words require a range check. We get it fo free from range-constraining chunks inside functions σ_0 and σ_1. Thus, for i from 16 to 63:

1. Apply σ_0 to W_{i-15}.

2. Add the following constraint for W_i:

$$w_{0,j+2} = w_{0,j+1} + w_{1,j+1} + w_{1,j+2} + w_{2,j+2} \cdot 2^3 + w_{3,j+2} \cdot 2^7 + w_{4,j+2} \cdot 2^{18} + w_{5,j+2} + w_{6,j+2} \cdot 2^{10} + w_{7,j+2} \cdot 2^{17} + w_{8,j+2} \cdot 2^{19},$$

3. Apply σ_1 to W_{i-2}.

Thus, the message schedule takes $5 \cdot 48 = 240$ rows.

The function σ_0 contains sparse mapping with base 4. Let a be divided to chunks a_0, a_1, a_2, a_3 which equals to $3, 4, 11, 14$ bits respectively. The values a'_0, a'_1, a'_2, a'_3 are in sparse form, and a' is a sparse a. **SHA-256 NORMALIZE4** lookup table is used for mapping to sparse representation and range-constraining for each chunk a_i, where bit-length of $a_i > 3$. If a chunk is 14 bits long, then it is constrained for free. Else the prover has calculate the sparse representation $\hat{a}_i$ for $2^j \cdot a_i$, where $j + \texttt{len}(a_i) = 14$ and $\texttt{len}(a_i)$ is bit-length of a_i. The tuple $\{s'_0, s'_1, s'_2, s'_3\}$ is a sparse representation of the result of σ_0 and the tuple $\{s_0, s_1, s_2, s_3\}$ is a normal representation. The size of elements of these tuples equals to $\{14, 14, 2, 2\}$ bits respectively.

Constraints:

$$w_{0,j+0} = w_{1,j+0} + w_{2,j+0} \cdot 2^3 + w_{3,j+0} \cdot 2^7 + w_{4,j+0} \cdot 2^{18}$$
$$(w_{1,j+0} - 7) \cdot (w_{1,j+0} - 6) \cdot \ldots \cdot w_{1,j+0} = 0$$
$$w_{5,j+1} + w_{6,j+1} \cdot 4^{14} + w_{7,j+1} \cdot 4^{28} + w_{8,j+1} \cdot 2^{30} = w_{2,j+1} + w_{3,j+1} \cdot 4^4 + w_{4,j+1} \cdot 4^{15} + w_{3,j+1} + w_{4,j+1} \cdot 4^{11} + w_{7,j+0} \cdot 4^{25} + w_{2,j+1} \cdot 4^{28} + w_{4,j+1} + w_{7,j+0} \cdot 4^{14} + w_{2,j+1} \cdot 4^{17} + w_{3,j+1} \cdot 4^{21}$$
$$(w_{7,j+1} - 3) \cdot (w_{7,j+1} - 2) \cdot (w_{7,j+1} - 1) \cdot w_{7,j+1} = 0 \ (w_{8,j+1} - 3) \cdot (w_{8,j+1} - 2) \cdot (w_{8,j+1} - 1) \cdot w_{8,j+1} = 0$$
$$\text{10 plookup constraints: } (w_{1,j+0}, w_{7,j+0}), (2^{10} \cdot w_{2,j+0}, w_{5,j+0}), (w_{2,j+0}, w_{2,j+1}), (2^3 \cdot w_{3,j+0}, w_{6,j+0}), (w_{3,j+0}, w_{3,j+1}), (w_{4,j+0}, w_{4,j+1}), (w_{1,j+2}, w_{5,j+1}), (w_{2,j+2}, w_{6,j+1}), (w_{3,j+2}, w_{7,j+2}), (w_{4,j+2}, w_{8,j+2})$$

The function σ_1 contains sparse mapping subcircuit with base 4. Let a be divided to chunks a_0, a_1, a_2, a_3 which equals to $10, 7, 2, 13$ bits respectively. The values a'_0, a'_1, a'_2, a'_3 are in sparse form and a' is a sparse a. **SHA-256 NORMALIZE4** lookup table is used for mapping to sparse representation and range-constraining in the same way as for σ_0. The tuple $\{s'_0, s'_1, s'_2, s'_3\}$ is a sparse representation of the result of σ_1 and the tuple $\{s_0, s_1, s_2, s_3\}$ is a normal representation. The size of elements of these tuples equals to $\{14, 14, 2, 2\}$ bits respectively.

Constraints:

$$w_{0,j+3} = w_{1,j+3} + w_{2,j+3} \cdot 2^{10} + w_{3,j+3} \cdot 2^{17} + w_{4,j+3} \cdot 2^{19}$$
$$(w_{3,j+3} - 3) \cdot (w_{3,j+3} - 2) \cdot (w_{3,j+3} - 1) \cdot w_{3,j+3} = 0$$
$$w_{5,j+3} + w_{6,j+3} \cdot 4^{14} + w_{7,j+3} \cdot 4^{28} + w_{8,j+3} \cdot 2^{30} = w_{2,j+3} + w_{3,j+3} \cdot 4^7 + w_{4,j+3} \cdot 4^9 + w_{3,j+3} + w_{4,j+3} \cdot 4^2 + w_{1,j+3} \cdot 4^{15} + w_{2,j+3} \cdot 4^{25} + w_{4,j+3} + w_{1,j+3} \cdot 4^{13} + w_{2,j+3} \cdot 4^{23} + w_{3,j+3} \cdot 4^{30}$$
$$(w_{7,j+3} - 3) \cdot (w_{7,j+3} - 2) \cdot (w_{7,j+3} - 1) \cdot w_{7,j+3} = 0 \ (w_{8,j+3} - 3) \cdot (w_{8,j+3} - 2) \cdot (w_{8,j+3} - 1) \cdot w_{8,j+3} = 0$$
$$\text{11 plookup constraints: } (2^4 \cdot (w_{1,j+3}, w_{5,j+3}), (2^7 \cdot w_{2,j+3}, w_{6,j+3}), (2 \cdot w_{4,j+3}, w_{7,j+3}), (w_{1,j+3}, w_{1,j+2}), (w_{2,j+3}, w_{2,j+2}), (w_{3,j+3}, w_{3,j+2}), (w_{4,j+3}, w_{4,j+2}), (w_{5,j+2}, w_{5,j+3}), (w_{6,j+2}, w_{6,j+3}), (w$$

Compression There are 64 rounds of compression. Each round of compression has the following table:

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
j + 0	e	e'_0	e_0	e_1	e_2	e_3	$\hat{e}_1$	$\hat{e}_2$	$\hat{e}_3$
j + 1	e'	f'	e'_1	e'_2	e'_3	s'_0	s'_1	s'_2	s'_3
j + 2	$ch_{0,sparse}$	$ch_{1,sparse}$	$ch_{2,sparse}$	$ch_{3,sparse}$	$-$	s_0	s_1	s_2	s_3
j + 3	g'	d	h	W_r	e_{new}	ch_0	ch_1	ch_2	ch_3
j + 4	$maj_{0,sparse}$	$maj_{1,sparse}$	$maj_{2,sparse}$	$maj_{3,sparse}$	a_{new}	maj_3	maj_0	maj_1	maj_2
j + 5	a'	b'			c'	s_0	s_1	s_2	s_3
j + 6	s'_1	s'_2	a'_0	a'_1	a'_2	a'_3	s'_3	s'_4	
j + 7	a		a_0	a_1	a_2	a_3	$\hat{a}_0$	$\hat{a}_1$	$\hat{a}_3$

The working variables a, b, c, d, e, f, g, h equals to the fixed initial $SHA-256$ values for the first chunk and to the sum of previous output and initial values for the rest of chunks. The values for chunk $c, c\neg 1$ are copy-constrained with output from previous round. The variables with quotes are corresponded sparse representation. For each chunk, the following rows are used:

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
j + 0	a	a'	b	b'	d	$-$	$-$	$-$	$-$
j + 1	c	c'	e	e'	h	$-$	$-$	$-$	$-$
j + 2	f	f'	g	g'	$-$	$-$	$-$	$-$	$-$

For the first round, $a, a', b', c', d, e, e', f', g', h$ are copy constrained with corresponded values from the table above.

For the second round, b', c', d, f', g', h are copy constrained with a', b', c, e', f', g from the table. The values a, e are copy constrained with a_{new}, e_{new} from the previous round.

For the third round, c', d, g', h are copy constrained with a', b, e', f. The values a, e are copy constrained with a_{new}, e_{new} from the previous round. The values b', f' are copy constrained with a', e' from the previous round.

In the rest of the rounds the following 'non-special' copy constraints are used:

1. The values a, e are copy constrained with a_{new}, e_{new} from the previous round.
2. The values b', f' are copy constrained with a', e' from the previous round.
3. The values c', g' are copy constrained with b', c' from the previous round.
4. The values d, h are copy constrained with a', e' from the round $r - 3$, where r is current round.

The Σ_0 function contains subcircuit with base 4. Let a be divided to chunks a_0, a_1, a_2, a_3 which equals to $2, 11, 9, 10$ bits respectively. The values a'_0, a'_1, a'_2, a'_3 are in sparse form and a' is a sparse a. The tuple $\{s'_0, s'_1, s'_2, s'_3\}$ is a sparse representation of the result of Σ_0 and the tuple $\{s_0, s_1, s_2, s_3\}$ is a normal representation. The size of elements of these tuples equals to $\{14, 14, 2, 2\}$ bits respectively. **SHA-256 NORMALIZE4** lookup table is used for mapping to sparse representation and range-constraining in the same way as for σ_0.

Constraints:

$$w_{0,j+7} = w_{2,j+7} + w_{3,j+7} \cdot 2^2 + w_{4,j+7} \cdot 2^{13} + w_{5,j+7} \cdot 2^{22}$$
$$w_{0,j+5} = w_{2,j+6} + w_{3,j+6} \cdot 4^2 + w_{4,j+6} \cdot 4^{13} + w_{5,j+6} \cdot 4^{22}$$
$$(w_{2,j+6} - 3) \cdot (w_{2,j+6} - 2) \cdot (w_{2,j+6} - 1) \cdot w_{2,j+6} = 0$$
$$w_{0,j+6} + w_{1,j+6} \cdot 4^{14} + w_{6,j+6} \cdot 4^{28} + w_{7,j+6} \cdot 2^{30} = w_{3,j+6} + w_{4,j+6} \cdot 4^{11} + w_{5,j+6} \cdot 4^{20} + w_{1,j+6} \cdot 2^{30} +$$
$$w_{4,j+6} + w_{5,j+6} \cdot 4^9 + w_{2,j+6} \cdot 4^{19} + w_{3,j+6} \cdot 4^{21} + w_{5,j+6} + w_{2,j+6} \cdot 4^{10} + w_{3,j+6} \cdot 4^{12} + w_{4,j+6} \cdot 4^{23}$$
$$(w_{6,j+6} - 3) \cdot (w_{6,j+6} - 2) \cdot (w_{6,j+6} - 1) \cdot w_{6,j+6} = 0 \; (w_{7,j+6} - 3) \cdot (w_{7,j+6} - 2) \cdot (w_{7,j+6} - 1) \cdot w_{7,j+6} = 0$$
$$\text{11 plookup constraints: } (2^3 \cdot (w_{3,j+6}, w_{6,j+6}), (2^5 \cdot w_{4,j+6}, w_{7,j+6}), (2^4 \cdot$$
$$w_{5,j+6}, w_{8,j+6}), (w_{2,j+6}, w_{2,j+5}), (w_{3,j+6}, w_{3,j+5}), (w_{4,j+6}, w_{4,j+5}), (w_{5,j+6}, w_{5,j+5}), (w_{5,j+5}, w_{0,j+6}), (w_{6,j+5}, w_{1,j+6}), (w$$

The Σ_1 function contains subcircuit with base 7. Let a be divided to chunks a_0, a_1, a_2, a_3 which equals to $6, 5, 14, 7$ bits respectively. The values a'_0, a'_1, a'_2, a'_3 are in sparse form, and a' is a sparse a. The tuple $\{s'_0, s'_1, s'_2, s'_3\}$ is a sparse representation of the result of Σ_1 and the tuple $\{s_0, s_1, s_2, s_3\}$ is a normal representation. The size of elements of these tuples equals to $\{14, 14, 2, 2\}$ bits respectively. **SHA-256 NORMALIZE7** lookup table is used for mapping to sparse representation and range-constraining in the same way as for σ_0.

Constraints:

$$w_{0,j+0} = w_{2,j+0} + w_{3,j+0} \cdot 2^6 + w_{4,j+0} \cdot 2^{11} + w_{5,j+0} \cdot 2^{25}$$
$$w_{0,j+1} = w_{1,j+0} + w_{2,j+1} \cdot 7^6 + w_{3,j+1} \cdot 7^{11} + w_{4,j+1} \cdot 7^{25}$$
$$w_{5,j+1} + w_{6,j+1} \cdot 4^{14} + w_{7,j+1} \cdot 4^{28} + w_{8,j+1} \cdot 2^{30} = w_{2,j+1} + w_{3,j+1} \cdot 4^5 + w_{4,j+1} \cdot 4^{19} + w_{1,j+0} \cdot 2^{26} +$$
$$w_{3,j+1} + w_{4,j+1} \cdot 4^{14} + w_{1,j+0} \cdot 4^{21} + w_{2,j+1} \cdot 4^{27} + w_{4,j+1} + w_{1,j+0} \cdot 4^7 + w_{2,j+1} \cdot 4^{13} + w_{3,j+1} \cdot 4^{27}$$
$$(w_{3,j+1} - 3) \cdot (w_{3,j+1} - 2) \cdot (w_{3,j+1} - 1) \cdot w_{3,j+1} = 0 \; (w_{4,j+1} - 3) \cdot (w_{4,j+1} - 2) \cdot (w_{4,j+1} - 1) \cdot w_{4,j+1} = 0$$
$$\text{11 plookup constraints: } (2^8 \cdot (w_{2,j+0}, w_{1,j+0}), (2^9 \cdot w_{3,j+0}, w_{2,j+1}), (2^7 \cdot$$
$$w_{5,j+0}, w_{4,j+1}), (w_{2,j+0}, w_{1,j+0}), (w_{3,j+0}, w_{2,j+1}), (w_{4,j+0}, w_{3,j+1}), (w_{5,j+0}, w_{4,j+1}), (w_{5,j+2}, w_{5,j+1}), (w_{6,j+2}, w_{6,j+1}), (w$$

The Maj function contains subcircuit with base 4 for a, b, c. **SHA-256 NORMALIZE MAJ** lookup table is used for mapping to sparse representation in the same way as for σ_0. The value of the maj function is stored in chunks of 8 bits $\{maj_0, maj_1, maj_2, maj_3\}$ and the corresponded sparse value is $\{maj_{0,sparse}, maj_{1,sparse}, maj_{2,sparse}, maj_{3,sparse}\}$ Constraints:

$$w_{0,j+4} + w_{1,j+4} \cdot 4^8 + w_{2,j+4} \cdot 4^{8 \cdot 2} + w_{3,j+4} \cdot 4^{8 \cdot 3} = w_{0,j+5} + w_{1,j+5} + w_{4,j+5}$$
$$\text{4 plookup constraints: } (w_{5,j+4}, w_{0,j+4}), (w_{6,j+4}, w_{1,j+4}), (w_{7,j+4}, w_{2,j+4}), (w_{8,j+4}, w_{3,j+4})$$

The Ch function contain sparse mapping subcircuit with base 7 for e, f, g. **SHA-256 NORMALIZE CH** lookup table is used for mapping to sparse representation in the same way as for σ_0. The value of the ch function is stored in chunks of 8 bits $\{ch_0, ch_1, ch_2, ch_3\}$ and the corresponded sparse value is $\{ch_{0,sparse}, ch_{1,sparse}, ch_{2,sparse}, ch_{3,sparse}\}$ Constraints:

$$w_{0,j+2} + w_{1,j+2} \cdot 7^8 + w_{2,j+2} \cdot 7^{8 \cdot 2} + w_{3,j+2} \cdot 7^{8 \cdot 3} = w_{0,j+1} + 2 \cdot w_{1,j+1} + 3 \cdot w_{0,j+3}$$
$$\text{4 plookup constraints: } (w_{5,j+3}, w_{0,j+2}), (w_{6,j+3}, w_{1,j+2}), (w_{7,j+3}, w_{2,j+2}), (w_{8,j+3}, w_{3,j+2})$$

Update the values a and e The value W_r is a word, where r is a number of round. It has to be copy-constrained with the word W_r in the message scheduling. Constraints:

$$w_{4,j+3} = w_{1,j+3} + w_{2,j+3} + w_{5,j+2} + w_{6,j+2} \cdot 2^{14} + w_{7,j+2} \cdot 2^{28} + w_{8,j+2} \cdot 2^{30} + w_{5,j+3} + w_{6,j+3} \cdot 2^8 +$$
$$w_{7,j+3} \cdot 2^{8 \cdot 2} + w_{8,j+3} \cdot 2^{8 \cdot 3} + k[r] + w_{3,j+3}, \text{ where } r \text{ is a number of round.}$$
$$w_{4,j+4} =$$
$$w_{4,j+3} - w_{1,j+3} + w_{5,j+5} + w_{6,j+5} \cdot 2^{14} + w_{7,j+5} \cdot 2^{28} + w_{8,j+5} \cdot 2^{30} + w_{5,j+4} + w_{6,j+4} \cdot 2^8 + w_{7,j+4} \cdot 2^{8 \cdot 2} + w_{8,j+4} \cdot 2^{8 \cdot 3}$$

Output of the round

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
j + 0	$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	–	–	–
j + 1	h_0	h_1	h_2	h_3	h_4	h_5	–	–	–
j + 2	a	b	c	d	e	f	–	–	–
j + 3	h_6	h_7	$\overline{g}$	$\overline{h}$	g	h	–	–	–

Evaluations:

The values $\overline{\xi}$ copy constrained with initial working variables of this round. The values a, b, c, d, e, f, g, h copy constrained with variables from the compression. The output of the round is $h_0, h_1, .., h_7$

Constraints:

$$\begin{aligned}
w_{0,j+1} &= w_{0,j+0} + w_{0,j+2} \\
w_{1,j+1} &= w_{1,j+0} + w_{1,j+2} \\
w_{2,j+1} &= w_{2,j+0} + w_{2,j+2} \\
w_{3,j+1} &= w_{3,j+0} + w_{3,j+2} \\
w_{4,j+1} &= w_{4,j+0} + w_{4,j+2} \\
w_{5,j+1} &= w_{5,j+0} + w_{5,j+2} \\
w_{0,j+3} &= w_{2,j+3} + w_{4,j+3} \\
w_{1,j+3} &= w_{3,j+3} + w_{5,j+3}
\end{aligned}$$

Cost The total value of rows is $48 \cdot 5 + 8 \cdot 64 + 3 = 755$ per chunk.

2.5.3 SHA-512 Circuit

SHA-512 uses the similar logical functions as in 2.5.2 which operates on 64-bits words. Thus, the preprocessing constraints for the SHA-512‘ circuit is a decomposition of k message blocks to 64 bits chunks without range proofs. For ‘eddsa‘ circuit, $k = 2$. All evaluations are similar to SHA-256 circuit.

Lookup tables We use the following lookup tables:

1. **SHA-256 NORMALIZE4** with 2 columns and 2^{14} rows. The first column contains all possible 14-bits words. The second column contains corresponding sparse representations with base 4. The constraints can be used for the range check and sparse representation simultaneously.
2. **SHA-256 NORMALIZE7** with 2 columns and 2^{14} rows. The first column contains all possible 14-bits words. The second column contains corresponding sparse representations with base 7. The constraints can be used for the range check and sparse representation simultaneously.
3. **SHA-512 NORMALIZE MAJ** with 2 columns and 2^{16} rows. The first column contains all possible 16-bits words. The second column contains corresponding sparse representations with base 4.
4. **SHA-512 NORMALIZE CH** with 2 columns and 2^{16} rows. The first column contains all possible 16-bits words. The second column contains corresponding sparse representations with base 7.

Message scheduling For each block of 1024 bits of the padded message the 80 words are constructed in the following way:

- The first 16 words are obtained by splitting the message.
- The last 64 words are obtained by using the functions σ_0, σ_1:

$$W_i = \sigma_1(W_{i-2}) \oplus W_{i-7} \oplus \sigma_0(W_{i-15}) \oplus W_{i-16}$$

Each round of the message scheduling has the following table:

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
j + 0	a	a_0	a_1	a_2	a_3	a_4	a_5	a_6	$\hat{a}_1$
j + 1		a'_0	a'_1	a'_2	a'_3	a'_4	a'_5	a'_6	
j + 2	W_i	s'_0	s'_1	s'_2	s'_3	s'_4	$\hat{s}'_4$	$--$	W_j
j + 3	w	s_0	s_1	s_2	s_3	s_4			
j + 4	s_0	s_1	s_2	s_3	s_4	s'_2	s'_3	s'_4	$\hat{s}'_4$
j + 5	s'_0	b'_0	b'_1	b'_2	b'_3	b'_4	b'_5	s'_1	$-$
j + 6	b	b_0	b_1	b_2	b_3	b_4	b_5	$\hat{b}_0$	$\hat{b}_5$

The first 16 words require a range check. We get it fo free from range-constraining chunks inside functions σ_0 and σ_1. Thus, for i from 16 to 80:

1. Apply σ_0 to W_{i-15}.
2. Add the following constraint for W_i:

$$w_{0,j+3} = w_{0,j+2} + w_{8,j+2} + w_{1,j+3} + w_{2,j+3} \cdot 2^{14} + w_{3,j+3} \cdot 2^{28} + w_{4,j+3} \cdot 2^{42} + w_{5,j+3} \cdot 2^{56} + w_{0,j+4} + w_{1,j+4} \cdot 2^{14} + w_{2,j+4} \cdot 2^{28} + w_{3,j+4} \cdot 2^{42} + w_{4,j+4} \cdot 2^{56},$$

3. Apply σ_1 to W_{i-2}.

Thus, the message schedule takes $7 \cdot 64 = 448$ rows.

The function σ_0 contains sparse mapping with base 4. Let a be divided to chunks $a_0, a_1, a_2, a_3, a_4, a_5, a_6$ which equals to $1, 6, 1, 14, 14, 14, 14$ bits respectively. The values $a'_0, a'_1, a'_2, a'_3, a'_4, a'_5, a'_6$ are in sparse form, and a' is a sparse a. **SHA-256 NORMALIZE4** lookup table is used for mapping to sparse representation and range-constraining for each chunk a_i, where bit-length of $a_i > 3$. If a chunk is 14 bits long, then it is constrained for free. Else the prover has to calculate the sparse representation $\hat{a}_i$ for $2^j \cdot a_i$, where $j + \texttt{len}(a_i) = 14$ and $\texttt{len}(a_i)$ is bit-length of a_i.

Constraints:

$$w_{0,j+0} = w_{1,j+0} + w_{2,j+0} \cdot 2 + w_{3,j+0} \cdot 2^7 + w_{4,j+0} \cdot 2^8 + w_{5,j+0} \cdot 2^{22} + w_{6,j+0} \cdot 2^{36} + w_{7,j+0} \cdot 2^{50}$$
$$(w_{1,j+0} - 1) \cdot w_{1,j+0} = 0$$
$$(w_{3,j+0} - 1) \cdot w_{3,j+0} = 0$$
$$w_{1,j+2} + w_{2,j+2} \cdot 4^{14} + w_{3,j+2} \cdot 4^{28} + w_{4,j+2} \cdot 2^{42} + w_{5,j+2} \cdot 4^{56} = w_{2,j+1} + w_{3,j+1} \cdot 4^6 + w_{4,j+1} \cdot 4^7 + w_{5,j+1} \cdot 2^{21} + w_{6,j+1} \cdot 4^{35} + w_{7,j+1} \cdot 4^{49} + w_{1,j+1} \cdot 4^{63} + w_{3,j+1} + w_{4,j+1} \cdot 4 + w_{5,j+1} \cdot 4^{15} + w_{6,j+1} \cdot 2^{29} + w_{7,j+1} \cdot 4^{43} + w_{4,j+1} + w_{5,j+1} \cdot 4^{14} + w_{6,j+1} \cdot 4^{28} + w_{7,j+1} \cdot 2^{42} + w_{1,j+1} \cdot 4^{56} + w_{2,j+1} \cdot 4^{57} + w_{3,j+1} \cdot 4^{63}$$

15 plookup constraints: $(w_{1,j+0}, w_{1,j+1}), (2^8 \cdot w_{2,j+0}, w_{8,j+0}), (w_{2,j+0}, w_{2,j+1}), (w_{3,j+0}, w_{3,j+1}), (w_{4,j+0}, w_{4,j+1}), (w_{5,j+0}, w_{5,j+1}), (w_{6,j+0}, w_{6,j+1}), (w_{7,j+0}, w_{7,j+1}), (w$
$w_{5,j+3}, w_{6,j+2})$

The function σ_1 contains sparse mapping subcircuit with base 4. Let a be divided to chunks $a_0, a_1, a_2, a_3, a_4, a_5$ which equals to $6, 13, 14, 14, 14, 3$ bits respectively. The values $a'_0, a'_1, a'_2, a'_3, a'_4, a'_5$ are in sparse form, and a' is a sparse a. **SHA-256 NORMALIZE4** lookup table is used for mapping to sparse representation and range-constraining in the same way as for σ_0.

Constraints:

$$w_{0,j+6} = w_{1,j+6} + w_{2,j+6} \cdot 2^6 + w_{3,j+6} \cdot 2^{19} + w_{4,j+6} \cdot 2^{33} + w_{5,j+6} \cdot 2^{47} + w_{6,j+6} \cdot 2^{61}$$
$$(w_{6,j+6} - 7) \cdot (w_{6,j+6} - 6) \cdot \ldots \cdot w_{6,j+6} = 0$$
$$w_{0,j+5} + w_{7,j+5} \cdot 4^{14} + w_{5,j+4} \cdot 4^{28} + w_{6,j+4} \cdot 2^{42} + w_{7,j+4} \cdot 4^{56} = w_{2,j+5} + w_{3,j+5} \cdot 4^{13} + w_{4,j+5} \cdot 4^{27} + w_{5,j+5} \cdot 2^{41} + w_{6,j+5} \cdot 4^{55} + w_{3,j+5} + w_{4,j+5} \cdot 4^{14} + w_{5,j+5} \cdot 4^{28} + w_{6,j+5} \cdot 2^{42} + w_{1,j+5} \cdot 4^{45} + w_{2,j+5} \cdot 4^{51} + w_{6,j+5} + w_{1,j+5} \cdot 4^3 + w_{2,j+5} \cdot 4^9 + w_{3,j+5} \cdot 2^{22} + w_{4,j+5} \cdot 4^{36} + w_{5,j+5} \cdot 4^{50}$$

15 plookup constraints: $(w_{1,j+6}, w_{1,j+5}), (2^8 \cdot w_{1,j+6}, w_{7,j+6}), (w_{2,j+6}, w_{2,j+5}), (w_{3,j+6}, w_{3,j+5}), (w_{4,j+6}, w_{4,j+5}), (w_{5,j+6}, w_{5,j+5}), (w_{6,j+6}, w_{6,j+5}), (2 \cdot w_{6,j+6}, w_{8,j+6}), (w_{0,j+4}, w_{0,j+5}), (w_{1,j+4}, w_{7,j+5}), (w_{2,j+4}, w_{5,j+4}), (w_{3,j+4}, w_{6,j+4}), (w_{4,j+4}, w_{7,j+4}), (2^6 \cdot w_{4,j+4}, w_{8,j+4})$

Compression There are 80 rounds of compression. Each round of compression has the following table:

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
j + 0	e	e_0	e_1	e_2	e_3	e_4	e_5	$\hat{e}_1$	$\hat{e}_3$
j + 1	$--$	e'_0	e'_1	e'_2	e'_3	e'_4	e'_5	$\hat{e}_5$	$--$
j + 2	e'	f'	$--$	$\hat{s}'_4$	s'_0	s'_1	s'_2	s'_3	s'_4
j + 3	$ch_{0,sparse}$	$ch_{1,sparse}$	$ch_{2,sparse}$	$ch_{3,sparse}$	s_0	s_1	s_2	s_3	s_4
j + 4	g'	$--$	$--$	$--$	e_{new}	ch_0	ch_1	ch_2	ch_3
j + 5	c'	d	h	W_r	a_{new}	maj_3	maj_0	maj_1	maj_2
j + 6	$maj_{0,sparse}$	$maj_{1,sparse}$	$maj_{2,sparse}$	$maj_{3,sparse}$	s_0	s_1	s_2	s_3	s_4
j + 7	a'	b'	$--$	$\hat{s}'_4$	s'_0	s'_1	s'_2	s'_3	s'_4
j + 8		a'_0	a'_1	a'_2	a'_3	a'_4	a'_5	$\hat{a}_5$	$--$
j + 9	a	a_0	a_1	a_2	a_3	a_4	a_5	$\hat{a}_2$	$\hat{a}_3$

The working variables a, b, c, d, e, f, g, h equals to the fixed initial $SHA - 512$ values for the first chunk and to the sum of previous output and initial values for the rest of chunks. The variables with quotes are corresponded sparse representation. For each chunk, the following rows are used:

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
j + 0	a	a'	b	b'	d	$-$	$-$	$-$	$-$
j + 1	c	c'	e	e'	h	$-$	$-$	$-$	$-$
j + 2	f	f'	g	g'	$-$	$-$	$-$	$-$	$-$

For the first round, $a, a', b', c', d, e, e', f', g', h$ are copy constrained with corresponded values from the table above.

For the second round, b', c', d, f', g', h are copy constrained with a', b', c, e', f', g from the table. The values a, e are copy constrained with a_{new}, e_{new} from the previous round.

For the third round, c', d, g', h are copy constrained with a', b, e', f. The values a, e are copy constrained with a_{new}, e_{new} from the previous round. The values b', f' are copy constrained with a', e' from the previous round.

In the rest of the rounds the following 'non-special' copy constraints are used:

1. The values a, e are copy constrained with a_{new}, e_{new} from the previous round.
2. The values b', f' are copy constrained with a', e' from the previous round.
3. The values c', g' are copy constrained with b', c' from the previous round.
4. The values d, h are copy constrained with a', e' from the round $r - 3$, where r is current round.

The Σ_0 function contains subcircuit with base 4. Let a be divided to chunks $a_0, a_1, a_2, a_3, a_4, a_5$ which equals to $14, 14, 6, 5, 14, 11$ bits respectively. The values $a'_0, a'_1, a'_2, a'_3, a'_4, a'_5$ are in sparse form, and a' is a sparse a. **SHA-256 NORMALIZE4** lookup table is used for mapping to sparse representation and range-constraining in the same way as for σ_0.

Constraints:

$$w_{0,j+9} = w_{1,j+9} + w_{2,j+9} \cdot 2^{14} + w_{3,j+9} \cdot 2^{28} + w_{4,j+9} \cdot 2^{34} + w_{5,j+9} \cdot 2^{39} + w_{6,j+9} \cdot 2^{53}$$

$$w_{4,j+7} + w_{5,j+7} \cdot 4^{14} + w_{6,j+7} \cdot 4^{28} + w_{7,j+7} \cdot 2^{42} + w_{8,j+7} \cdot 4^{56} =$$
$$w_{3,j+8} + w_{4,j+8} \cdot 4^{6} + w_{5,j+8} \cdot 4^{11} + w_{6,j+8} \cdot 2^{25} + w_{1,j+8} \cdot 4^{36} + w_{2,j+8} \cdot 4^{50} + w_{4,j+8} + w_{5,j+8} \cdot 4^{5} + w_{6,j+8} \cdot 4^{19} +$$
$$w_{1,j+8} \cdot 2^{30} + w_{2,j+8} \cdot 4^{44} + w_{3,j+8} \cdot 4^{58} + w_{5,j+8} + w_{6,j+8} \cdot 4^{14} + w_{1,j+8} \cdot 4^{25} + w_{2,j+8} \cdot 2^{39} + w_{3,j+8} \cdot 4^{53} + w_{4,j+8} \cdot 4^{59}$$

15 plookup constraints: $(w_{1,j+9}, w_{1,j+8}), (w_{2,j+9}, w_{2,j+8}), (2^8 \cdot w_{3,j+9}, w_{7,j+9}), (w_{3,j+9}, w_{3,j+8}), (2^9 \cdot w_{4,j+9}, w_{8,j+9}), (w_{4,j+9}, w_{4,j+8}), (w_{5,j+9}, w_{5,j+8}), (2^3 \cdot w_{6,j+9}, w_{7,j+8}), (w_{6,j+9}, w_{6,j+8}), (w_{4,j+6}, w_{4,j+7}), (w_{5,j+6}, w_{5,j+7}), (w_{6,j+6}, w_{6,j+7}), (w_{7,j+6}, w_{7,j+7}), (w_{8,j+6}, w_{8,j+7}), (2$ $w_{8,j+7}, w_{3,j+7})$

The Σ_1 function contains subcircuit with base 7. Let a be divided to chunks $a_0, a_1, a_2, a_3, a_4, a_5$ which equals to $14, 4, 14, 9, 14, 9$ bits respectively. The values $a'_0, a'_1, a'_2, a'_3, a'_4, a'_5$ are in sparse form, and a' is a sparse a. **SHA-256 NORMALIZE7** lookup table is used for mapping to sparse representation and range-constraining in the same way as for σ_0.

Constraints:

$$w_{0,j+0} = w_{1,j+0} + w_{2,j+0} \cdot 2^{14} + w_{3,j+0} \cdot 2^{18} + w_{4,j+0} \cdot 2^{32} + w_{5,j+0} \cdot 2^{41} + w_{6,j+0} \cdot 2^{55}$$

$$w_{4,j+2} + w_{5,j+2} \cdot 4^{14} + w_{6,j+2} \cdot 4^{28} + w_{7,j+2} \cdot 2^{42} + w_{8,j+2} \cdot 4^{56} =$$
$$w_{2,j+1} + w_{3,j+1} \cdot 4^{4} + w_{4,j+1} \cdot 4^{18} + w_{5,j+1} \cdot 2^{27} + w_{6,j+1} \cdot 4^{41} + w_{1,j+1} \cdot 4^{50} + w_{3,j+1} + w_{4,j+1} \cdot 4^{14} + w_{5,j+1} \cdot 4^{23} +$$
$$w_{6,j+1} \cdot 2^{37} + w_{1,j+1} \cdot 4^{46} + w_{3,j+1} \cdot 4^{60} + w_{5,j+1} + w_{6,j+1} \cdot 4^{14} + w_{1,j+1} \cdot 4^{23} + w_{2,j+1} \cdot 2^{37} + w_{3,j+1} \cdot 4^{41} + w_{4,j+1} \cdot 4^{55}$$

15 plookup constraints: $(w_{1,j+0}, w_{1,j+1}), (w_{2,j+0}, w_{2,j+1}), (2^{10} \cdot w_{2,j+0}, w_{7,j+0}), (w_{3,j+0}, w_{3,j+1}), (2^{5} \cdot w_{4,j+0}, w_{8,j+0}), (w_{4,j+0}, w_{4,j+1}), (w_{5,j+0}, w_{5,j+1}), (2^{3} \cdot w_{6,j+0}, w_{7,j+1}), (w_{6,j+0}, w_{6,j+1}), (w_{4,j+3}, w_{4,j+2}), (w_{5,j+3}, w_{5,j+2}), (w_{6,j+3}, w_{6,j+2}), (w_{7,j+3}, w_{7,j+2}), (w_{8,j+3}, w_{8,j+2}), (2^{6}$ $w_{8,j+3}, w_{3,j+2})$

The Maj function contains subcircuit with base 4 for a, b, c. **SHA-512 NORMALIZE MAJ** lookup table is used for mapping to sparse representation in the same way as for σ_0. The value of the *maj* function is stored in chunks of 16 bits. Constraints:

$$w_{0,j+6} + w_{1,j+6} \cdot 4^{16} + w_{2,j+6} \cdot 4^{16 \cdot 2} + w_{3,j+6} \cdot 4^{16 \cdot 3} = w_{0,j+7} + w_{1,j+7} + w_{0,j+5}$$

4 plookup constraints: $(w_{5,j+5}, w_{0,j+6}), (w_{6,j+5}, w_{1,j+6}), (w_{7,j+5}, w_{2,j+6}), (w_{8,j+5}, w_{3,j+6})$

The Ch function contain sparse mapping subcircuit with base 7 for e, f, g. **SHA-512 NORMALIZE CH** lookup table is used for mapping to sparse representation in the same way as for σ_0. The value of the *ch* function is stored in chunks of 16 bits. Constraints:

$$w_{0,j+3} + w_{1,j+3} \cdot 7^{16} + w_{2,j+3} \cdot 7^{16 \cdot 2} + w_{3,j+3} \cdot 7^{16 \cdot 3} = w_{0,j+2} + 2 \cdot w_{1,j+2} + 3 \cdot w_{0,j+4}$$

4 plookup constraints: $(w_{5,j+4}, w_{0,j+3}), (w_{6,j+4}, w_{1,j+3}), (w_{7,j+4}, w_{2,j+3}), (w_{8,j+3}, w_{3,j+2})$

Update the values a and e Constraints:

$$w_{4,j+4} = w_{1,j+5} + w_{2,j+5} + w_{5,j+3} \cdot 2^{14} + w_{6,j+3} \cdot 2^{28} + w_{7,j+3} \cdot 2^{42} + w_{8,j+3} \cdot 2^{56} + w_{5,j+4} + w_{6,j+4} \cdot 2^{16} + w_{7,j+4} \cdot 2^{16 \cdot 2} + w_{8,j+4} \cdot 2^{16 \cdot 3} + k[r] + w_{3,j+5},\ \text{where } r \text{ is a number of round.}$$

$$w_{4,j+5} = w_{4,j+4} - w_{1,j+5} + w_{4,j+6} + w_{5,j+6} \cdot 2^{14} + w_{6,j+6} \cdot 2^{28} + w_{7,j+6} \cdot 2^{42} + w_{8,j+6} \cdot 2^{56} + w_{5,j+5} + w_{6,j+5} \cdot 2^{16} + w_{7,j+5} \cdot 2^{16 \cdot 2} + w_{8,j+5} \cdot 2^{16 \cdot 3}$$

Output of the round The final calculations uses the same table and constraints as in 2.5.2.

Cost The total value of rows is $64 \cdot 7 + 10 \cdot 80 + 3 = 1248$ per chunk.

2.5.4 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_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
j + 0	$0_{\mathbb{F}}$	$I[2]$	$I[3]$	$T_{1,0}$	$T_{1,1}$	$T_{1,2}$	$T_{2,0}$	$T_{2,1}$	$T_{2,2}$
j + 1	$T_{3,0}$	$T_{3,1}$	$T_{3,2}$	$T_{4,0}$	$T_{4,1}$	$T_{4,2}$	$T_{5,0}$	$T_{5,1}$	$T_{5,2}$
...									
j + 21	$T_{63,0}$	$T_{63,1}$	$T_{63,2}$	$T_{64,0}$	$T_{64,1}$	$T_{64,2}$	$O[1]$	H	$O[3]$

Constraints:

For $j + 0$:
$$[w_{3,j+0}, w_{4,j+0}, w_{5,j+0}] = [w_{0,j+0}^5, w_{1,j+0}^5, w_{2,j+0}^5] \times M + RC$$
$$[w_{6,j+0}, w_{7,j+0}, w_{8,j+0}] = [w_{2,j+0}^5, w_{3,j+0}^5, w_{4,j+0}^5] \times M + RC$$
For $j + 1$:
$$[w_{0,j+1}, w_{1,j+1}, w_{2,j+1}] = [w_{2,j+0}^5, w_{7,j+0}^5, w_{8,j+0}^5] \times M + RC$$
$$[w_{3,j+1}, w_{4,j+1}, w_{5,j+1}] = [w_{0,j+1}^5, w_{1,j+1}^5, w_{2,j+1}^5] \times M + RC$$
$$[w_{6,j+1}, w_{7,j+1}, w_{8,j+1}] = [w_{3,j+1}, w_{4,j+1}, w_{5,j+1}^5] \times M + RC$$
For $j + k$, $k \in \{2, 19\}$:
$$[w_{0,j+k}, w_{1,j+k}, w_{2,j+k}] = [w_{6,j+k-1}, w_{7,j+k-1}, w_{8,j+k-1}^5] \times M + RC$$
$$[w_{3,j+k}, w_{4,j+k}, w_{5,j+k}] = [w_{0,j+k}, w_{1,j+k}, w_{2,j+k}^5] \times M + RC$$
$$[w_{6,j+k}, w_{7,j+k}, w_{8,j+k}] = [w_{3,j+k}, w_{4,j+k}, w_{5,j+k}^5] \times M + RC$$
For $j + 20$:
$$[w_{0,j+20}, w_{1,j+20}, w_{2,j+20}] = [w_{6,j+19}, w_{7,j+19}, w_{8,j+19}^5] \times M + RC$$

$$[w_{3,j+20}, w_{4,j+20}, w_{5,j+20}] = [w_{0,j+20}, w_{1,j+20}, w^5_{2,j+20}] \times M + RC$$
$$[w_{6,j+20}, w_{7,j+20}, w_{8,j+20}] = [w^5_{2,j+20}, w^5_{3,j+20}, w^5_{4,j+20}] \times M + RC$$
$$\text{For } j+21:$$
$$[w_{0,j+21}, w_{1,j+21}, w_{2,j+21}] = [w^5_{2,j+20}, w^5_{7,j+20}, w^5_{8,j+20}] \times M + RC$$
$$[w_{3,j+21}, w_{4,j+21}, w_{5,j+21}] = [w^5_{0,j+21}, w^5_{1,j+21}, w^5_{2,j+21}] \times M + RC$$
$$[w_{6,j+21}, w_{7,j+21}, w_{8,j+21}] = [w^5_{2,j+21}, w^5_{3,j+21}, w^5_{4,j+21}] \times M + RC$$

2.5.5 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 $T_i \coloneqq H_i$ // 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 2.5.4
   4.2 $n = \frac{n}{2}$
   4.3 for j from 0 to $n-1$:
      4.3.1 $T_i \coloneqq T'_i$. // just notation for simplicity, not a real part of the circuit

2.5.6 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_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
j + 0	s	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7
j + 1	z_8	z_9	z_{10}	z_{11}	z_{12}	z_{13}	z_{14}	z_{15}	z_{16}
j + 2	z_{17}	z_{18}	z_{19}	z_{20}	z_{21}	z_{22}	z_{23}	z_{24}	z_{25}

Evaluations:

$$z_0 = s + 2^{253} - L \text{ is decomposed into 10-bit windows } k_0 + 2^{10} \cdot k_1 + 2^{10\cdot 2} \cdot k_2 + ... + 2^{10\cdot 25} \cdot k_{25}$$
$$z_i = (z_{i-1} - k_{i-1})/2^{10}$$

Constraints:

$$w_{1,j} = w_{0,j} + 2^{253} - L$$
Each $w_{i,k} - 2^{10} \cdot w_{i+a,k+b}$, where $i = 1, .., 8$ for $k = 0$, $i = 0, .., 8$ for $k = 1$ and $i = 0, .., 7$ for $k = 2$, $(i+1) = b \cdot 9 + a$ is range-constrained by 10-bits plookup table.
$w_{8,j+2} \cdot 2^7$ is range-constrained by 10-bits plookup table.

It costs 3 rows.

2. $k = \{k_0, k_1, ..., k_7\}$ == SHA-512$(data||R||A||M)$ // See section **??** It costs $1248 \cdot 2 = 2496$ rows.
3. $sB? =?R + kA$:
   3.1 Fixed-base scalar multiplication circuit is used for $sB = S$. The cell $w_{0,j+84}$ is copy-constrained with $w_{0,j+0}$ from the range circuit.
   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.

j + 0	x_s	x_r	y_r	x_t	y_t	y_s	$--$	$--$	$--$

In total, three constraints are used for addition:

$$w_{3,j+0} \cdot (1 + dw_{0,j+0} \cdot (-w_{1,j+0}) \cdot w_{5,j+0} \cdot w_{2,j+0}) = w_{0,j+0} \cdot w_{2,j+0} + (-w_{1,j+0}) \cdot w_{5,j+0}$$
$$w_{4,j+0} \cdot (1 - dw_{0,j+0} \cdot (-w_{1,j+0}) \cdot w_{5,j+0} \cdot w_{2,j+0}) = w_{0,j+0} \cdot (-w_{1,j+0}) + w_{2,j+0} \cdot w_{5,j+0}$$
$$(-w_{1,j+0})^2 + w_{2,j+0}^2 = 1 - d \cdot w_{1,j+0}^2 \cdot w_{2,j+0}^2$$ $w_{1,j+0}, w_{5,j+0}$ are copy-constrained with $w_{6,j+84}, w_{7,j+84}$ from fixed-base multiplication circuit.

3.3 Variable-base scalar multiplication circuit for $T = k \cdot A$, where cells $w_{1,j+254}, w_{2,j+254}$ are copy constrained with $w_{3,j+0}, w_{5,j+0}$.

It costs $3 + 2496 + 85 + 255 + 1 = 2840$ rows.

2.5.7 Elliptic Curves Arithmetics

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.

Computations over a non-native field. Let $\mathbb{F}_p$ be an edwards25519 field, i.e. the size of the field is $2^{255} - 19$. In order to provide computations over non-native $\mathbb{F}_p$ we use constraints over native field $\mathbb{F}_k$. Let $k < p$ be a prime number, which size is 254 bits. Additionally, we compute an integer t, such that $2^t \cdot k \geq p^2 + p$. In our case, $t = 257$. Now, we want to check equality:

$$a \cdot b = p \cdot q + r, r = a \cdot b \mod p$$

Each positive integer a, b, q, r is divided into 13 limbs, where the sizes of limbs are $20, 20, ..., 20, 15$ bits respectively, where 15 is the least significant bits. To check that a, b, q and r are less than p, we use range proofs. For this purpose, a lookup table with two columns is used. The first column contains all integers in the range $[0, 2^{20})$, and the second column contains almost all zeros except 18 ones from $2^{15} - 19$ to $2^{15} - 1$.

1. The limbs $a_0, a_1, ..., a_{12}$ are range-constrained by the lookup table.
2. The value $a_{12} \cdot 2^5$ are range-constrained by the lookup table.
3. Let $\xi = (\sum_{i=0}^{11}(a_i - 2^{20} + 1))^{-1}$.
4. $(\sum_{i=0}^{11}(a_i - 2^{20} + 1) \cdot (\xi \cdot (\sum_{i=0}^{11}(a_i - 2^{20} + 1) - 1) = 0$
5. $\xi \cdot (\sum_{i=0}^{11}(a_i - 2^{20} + 1) + (1 - \xi \cdot (\sum_{i=0}^{11}(a_i - 2^{20} + 1)) \cdot c - 1 = 0$, where c is corresponding second column's value for $a_1 2$.

Then we constrain the equation modulo n and 2^t as follows:

1. $(a \cdot b) \mod k = (p \cdot q + r) \mod k$
2. $a'_0 = a_{12} + a_{11} \cdot 2^{15} + a_{10} \cdot 2^{35} + a_9 \cdot 2^{55}$, $a'_1 = a_8 + a_7 \cdot 2^{20} + a_6 \cdot 2^{40}$, $a'_1 = a_5 + a_4 \cdot 2^{20} + a_3 \cdot 2^{40}$, $a'_1 = a_2 + a_1 \cdot 2^{20} + a_0 \cdot 2^{40}$. The new limbs for b, q, and r are constructed similarly.
3. Let p' be $-p \mod 2^t$ and $p' = p'_0 + p'_1 \cdot 2^{75} + p'_2 \cdot 2^{135} + p'_3 \cdot 2^{195}$. The limbs p'_0, p'_1, p'_2 and p'_3 are circuits parameters.
4. Compute the following limbs:
   4.1 $t_0 = a'_0 \cdot b'_0 + p'_0 \cdot q'_0$
   4.2 $t_1 = a'_1 \cdot b'_0 + a'_0 \cdot b'_1 + p'_0 \cdot q'_1 + p'_1 \cdot q'_0$
   4.3 $t_2 = a'_2 \cdot b'_0 + a'_0 \cdot b'_2 + a'_1 \cdot b'_1 + p'_0 \cdot q'_2 + p'_2 \cdot q'_0 + p'_1 \cdot q'_1$
   4.4 $t_3 = a'_3 \cdot b'_0 + a'_0 \cdot b'_3 + a'_1 \cdot b'_2 + a'_2 \cdot b'_1 + p'_0 \cdot q'_3 + p'_3 \cdot q'_0 + p'_1 \cdot q'_2 + p'_2 \cdot q'_1$
   4.5 $t_4 = a'_3 \cdot b'_1 + a'_1 \cdot b'_3 + a'_2 \cdot b'_2 + p'_1 \cdot q'_3 + p'_3 \cdot q'_1 + p'_2 \cdot q'_2$
5. $u_0 = t_0 - r'_0 + t_1 \cdot 2^{75} - r'_1 \cdot 2^{75} = v_0 \cdot 2^{135}$

6. $u_1 = t_2 - r'_2 + t_3 \cdot 2^{60} - r'_3 \cdot 2^{60} + t_4 \cdot 2^{120} + v_0 = v_1 \cdot 2^{122}$

7. The value v_0 has to be less than 2^{68} and $v_1 \leq 2^{78}$.

   7.1 $v_0 = v_{0,3} + v_{0,2} \cdot 2^8 + v_{0,1} \cdot 2^{28} + v_{0,0} \cdot 2^{48}$

   7.2 Lookup constraints: $(v_{0,3}), (v_{0,2}), (v_{0,1}), (v_{0,0}), (v_{0,3} \cdot 2^{12})$

   7.3 $v_1 = v_{1,3} + v_{1,2} \cdot 2^{18} + v_{1,1} \cdot 2^{38} + v_{1,0} \cdot 2^{58}$

   7.4 Lookup constraints: $(v_{1,3}), (v_{1,2}), (v_{1,1}), (v_{1,0}), (v_{1,3} \cdot 2^2)$

Non-native miltiplication circuit for $a \cdot b$

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
j + 0	a'_0	a_0	a_1	a_2	a_3	a_4	a_5	a_6	ξ
j + 1	a'_1	a'_2	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	c
j + 2	b'_0	b_0	b_1	b_2	b_3	b_4	b_5	b_6	ξ
j + 3	b'_1	b'_2	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	c
j + 4	q'_0	q_0	q_1	q_2	q_3	q_4	q_5	q_6	ξ
j + 5	q'_1	q'_2	q_7	q_8	q_9	q_{10}	q_{11}	q_{12}	c
j + 6	r'_0	r_0	r_1	r_2	r_3	r_4	r_5	r_6	ξ
j + 7	r'_1	r'_3	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}	c
j + 8	q'_0	q'_1	q'_2	r'_3	r'_2	r'_1	r'_0	q_0	q_1
j + 9	b'_1	b'_2	q_2	b_0	b_1	b_2	$--$	v_0	v_1
j + 10	a'_0	a'_1	a'_2	b'_0	a_0	a_1	a_2	$v_{0,3}$	$v_{1,3}$
j + 11	v_0	$v_{0,0}$	$v_{0,1}$	$v_{0,2}$	v_1	$v_{1,0}$	$v_{1,1}$	$v_{1,2}$	$--$

Let s_a be $(w_{1,j+0}+w_{2,j+0}+w_{3,j+0}+w_{4,j+0}+w_{5,j+0}+w_{6,j+0}+w_{7,j+0}+w_{2,j+1}+w_{3,j+1}, w_{4,j+1}, w_{5,j+1}+w_{6,j+1} - 12 \cdot (2^20 - 1))$.
Let s_b be $(w_{1,j+2}+w_{2,j+2}+w_{3,j+2}+w_{4,j+2}+w_{5,j+2}+w_{6,j+2}+w_{7,j+2}+w_{2,j+3}+w_{3,j+3}, w_{4,j+3}, w_{5,j+3}+w_{6,j+3} - 12 \cdot (2^20 - 1))$.
Let s_q be $(w_{1,j+4}+w_{2,j+4}+w_{3,j+4}+w_{4,j+4}+w_{5,j+4}+w_{6,j+4}+w_{7,j+4}+w_{2,j+5}+w_{3,j+5}, w_{4,j+5}, w_{5,j+5}+w_{6,j+5} - 12 \cdot (2^20 - 1))$.
Let s_r be $(w_{1,j+6}+w_{2,j+6}+w_{3,j+6}+w_{4,j+6}+w_{5,j+6}+w_{6,j+6}+w_{7,j+6}+w_{2,j+7}+w_{3,j+7}, w_{4,j+7}, w_{5,j+7}+w_{6,j+7} - 12 \cdot (2^20 - 1))$.
Constraints:

- $s_a \cdot (w_{8,j+0} \cdot s_a - 1) = 0$
- $w_{8,j+0} \cdot (s_a) + (1 - w_{8,j+0} \cdot s_a) \cdot w_{8,j+1} - 1 = 0$
- $w_{0,j+0} = w_{7,j+1} + w_{6,j+1} \cdot 2^{15} + w_{5,j+1} \cdot 2^{35} + w_{4,j+1} \cdot 2^{55}$
- $w_{0,j+1} = w_{3,j+1} + w_{2,j+1} \cdot 2^{20} + w_{7,j+0} \cdot 2^{40}$
- $w_{1,j+1} = w_{6,j+0} + w_{5,j+0} \cdot 2^{20} + w_{4,j+0} \cdot 2^{40}$
- $s_b \cdot (w_{8,j+2} \cdot s_b - 1) = 0$
- $w_{8,j+2} \cdot (s_b) + (1 - w_{8,j+2} \cdot s_b) \cdot w_{8,j+3} - 1 = 0$
- $w_{0,j+2} = w_{7,j+3} + w_{6,j+3} \cdot 2^{15} + w_{5,j+3} \cdot 2^{35} + w_{4,j+3} \cdot 2^{55}$
- $w_{0,j+3} = w_{3,j+3} + w_{2,j+3} \cdot 2^{20} + w_{7,j+2} \cdot 2^{40}$
- $w_{1,j+3} = w_{6,j+2} + w_{5,j+2} \cdot 2^{20} + w_{4,j+2} \cdot 2^{40}$
- $s_q \cdot (w_{8,j+4} \cdot s_q - 1) = 0$
- $w_{8,j+4} \cdot (s_q) + (1 - w_{8,j+4} \cdot s_q) \cdot w_{8,j+5} - 1 = 0$
- $w_{0,j+4} = w_{7,j+5} + w_{6,j+5} \cdot 2^{15} + w_{5,j+5} \cdot 2^{35} + w_{4,j+5} \cdot 2^{55}$
- $w_{0,j+5} = w_{3,j+5} + w_{2,j+5} \cdot 2^{20} + w_{7,j+4} \cdot 2^{40}$
- $w_{1,j+5} = w_{6,j+4} + w_{5,j+4} \cdot 2^{20} + w_{4,j+4} \cdot 2^{40}$
- $s_r \cdot (w_{8,j+6} \cdot s_r - 1) = 0$
- $w_{8,j+6} \cdot (s_r) + (1 - w_{8,j+6} \cdot s_r) \cdot w_{8,j+7} - 1 = 0$
- $w_{4,j+8} = w_{7,j+7} + w_{6,j+7} \cdot 2^{15} + w_{5,j+7} \cdot 2^{35} + w_{4,j+7} \cdot 2^{55}$
- $w_{0,j+7} = w_{3,j+7} + w_{2,j+7} \cdot 2^{20} + w_{7,j+6} \cdot 2^{40}$
- $w_{1,j+7} = w_{6,j+6} + w_{5,j+6} \cdot 2^{20} + w_{4,j+6} \cdot 2^{40}$
- $w_{1,j+7} = w_{1,j+6} + w_{2,j+6} \cdot 2^{20} + w_{3,j+6} \cdot 2^{40}$
- $w_{3,j+8} - w_{1,j+7} = 0$
- $w_{5,j+8} - w_{0,j+7} = 0$

- $w_{0,j+10} \cdot w_{3,j+10} + p'_0 \cdot w_{0,j+8} - w_{6,j+8} + 2^{75} \cdot (w_{1,j+10} \cdot w_{3,j+10} + w_{0,j+10} \cdot w_{0,j+9} + p'_0 \cdot w_{1,j+8} + p'_1 \cdot w_{0,j+8}) - w_{5,j+8}\ dot 2^{75} - w_{7,j+9} \cdot 2^{135} = 0$
- $w_{2,j+10} \cdot w_{3,j+10} + w_{0,j+10} \cdot w_{1,j+9} + w_{1,j+10} \cdot w_{0,j+9} + p'_0 \cdot w_{2,j+8} + p'_2 \cdot w_{0,j+8} + p'_1 \cdot w_{1,j+8} - w_{4,j+8} + 2^{60} \cdot ((w_{4,j+10} \cdot 2^{40} + w_{5,j+10} \cdot 2^{20} + w_{6,j+10}) \cdot w_{3,j+10} + w_{0,j+10} \cdot (w_{3,j+9} \cdot 2^{40} + w_{4,j+9} \cdot 2^{20} + w_{5,j+9}) + w_{1,j+10} \cdot w_{1,j+9} + w_{2,j+10} \cdot w_{0,j+9} + p'_0 \cdot (w_{7,j+8} \cdot 2^{40} + w_{8,j+8} \cdot 2^{20} + w_{2,j+9}) + p'_3 \cdot w_{0,j+8} + p'_1 \cdot w_{2,j+8} + p'_2 \cdot w_{1,j+8}) - 2^{60} \cdot w_{3,j+8} + 2^{120} \cdot ((w_{4,j+10} \cdot 2^{40} + w_{5,j+10} \cdot 2^{20} + w_{6,j+10}) \cdot w_{0,j+9} + w_{1,j+10} \cdot (w_{3,j+9} \cdot 2^{40} + w_{4,j+9} \cdot 2^{20} + w_{5,j+9}) + w_{2,j+10} \cdot w_{1,j+9} + p'_1 \cdot (w_{7,j+8} \cdot 2^{40} + w_{8,j+8} \cdot 2^{20} + w_{2,j+9}) + p'_3 \cdot w_{1,j+8} + p'_2 \cdot w_{2,j+8}) + w_{7,j+9} - 2^{122} \cdot w_{8,j+9} = 0$
- $w_{4,j+11} = w_{5,j+11} \cdot 2^{58} + w_{6,j+11} \cdot 2^{38} + w_{7,j+11} \cdot 2^{18} + w_{8,j+10}$
- $w_{0,j+11} = w_{1,j+11} \cdot 2^{48} + w_{2,j+11} \cdot 2^{28} + w_{3,j+11} \cdot 2^{8} + w_{7,j+10}$

Lookup constraints:

- $(w_{1,j+0}), (w_{2,j+0}), (w_{3,j+0}, (w_{4,j+0}), (w_{5,j+0}), (w_{6,j+0}), (w_{7,j+0}), (w_{2,j+1}), (w_{3,j+1}), (w_{4,j+1}), (w_{5,j+1}), (w_{6,j+1}), (w$ $2^5)$
- $(w_{1,j+2}), (w_{2,j+2}), (w_{3,j+2}, (w_{4,j+2}), (w_{5,j+2}), (w_{6,j+2}), (w_{7,j+2}), (w_{2,j+3}), (w_{3,j+3}), (w_{4,j+3}), (w_{5,j+3}), (w_{6,j+3}), (w$ $2^5)$
- $(w_{1,j+4}), (w_{2,j+4}), (w_{3,j+4}, (w_{4,j+4}), (w_{5,j+4}), (w_{6,j+4}), (w_{7,j+4}), (w_{2,j+5}), (w_{3,j+5}), (w_{4,j+5}), (w_{5,j+5}), (w_{6,j+5}), (w$ $2^5)$
- $(w_{1,j+6}), (w_{2,j+6}), (w_{3,j+6}, (w_{4,j+6}), (w_{5,j+6}), (w_{6,j+6}), (w_{7,j+6}), (w_{2,j+7}), (w_{3,j+7}), (w_{4,j+7}), (w_{5,j+7}), (w_{6,j+7}), (w$ $2^5)$
- $(w_{1,j+11}), (w_{2,j+11}), (w_{3,j+11}), (w_{7,j+10}), (w_{7,j+10} \cdot 2^{12})$
- $(w_{5,j+11}), (w_{6,j+11}), (w_{7,j+11}), (w_{8,j+10}), (w_{8,j+10} \cdot 2^{2})$

Copy constraints:

$$\begin{gathered}
(w_{0,j+8}, w_{0,j+4}), (w_{1,j+8}, w_{0,j+5}), (w_{2,j+8}, w_{1,j+5}), \\
(w_{6,j+8}, w_{0,j+6}), (w_{7,j+8}, w_{1,j+4}), (w_{8,j+8}, w_{2,j+4}), (w_{0,j+9}, w_{0,j+3}), \\
(w_{1,j+9}, w_{1,j+3}), (w_{2,j+9}, w_{3,j+4}), (w_{3,j+9}, w_{1,j+2}), (w_{4,j+9}, w_{2,j+2}), \\
(w_{5,j+9}, w_{3,j+2}), (w_{7,j+9}, w_{0,j+11}), (w_{8,j+9}, w_{4,j+11}), (w_{0,j+10}, w_{0,j+0}), \\
(w_{1,j+10}, w_{0,j+1}), (w_{2,j+10}, w_{1,j+1}), (w_{3,j+10}, w_{0,j+2}), (w_{4,j+10}, w_{1,j+0}), \\
(w_{5,j+10}, w_{2,j+0}), (w_{6,j+10}, w_{3,j+0})
\end{gathered}$$

The proof of the addition of the numbers from $\mathbb{F}_p$ proceeds as in the multiplication. We check an equation modulo k and 2^t:

$$a + b = p \cdot q + r$$

We use the range proofs as above for a, b, and r. Since the value q can be equal to 0 or 1, we use the short-range check without any lookups. The second part of the proof can be implemented as the following:

1. $(a \cdot b) \mod k = (p \cdot q + r) \mod k$
2. $a_0 \cdot b_0 + p' \cdot q_0 - r_0 = v \cdot 2^3$, where p' is $-p \mod 2^3$.
3. Range-check that $v \leq 2^{27}$.

It is possible to extend to $n < p$ additions. Thus, the value q is equal to an amount of additions minus 1, $t = q + 2$. The number of t_i is increased by depending on t. Particularly, the scalar multiplication proceeds as an extension of additions.

However, we need more special cases of non-native arithmetics for the elliptic curve's multiplication circuits.

1. Let $a^2 \mp b^2 \mp c = p \cdot q + r$, where c is constant. We change a range check for q to q < 2p. The total amount of the limbs does not change, but the last limb has to be checked by multiplication to 2^4.
2. Let $2 \cdot a \cdot b$. This case is similar to the case from step 1.

Fixed-base scalar multiplication circuit : We precompute all values $(u'_{s',i}, v'_{s',i}) = w(B, s', k) = k_i \cdot 8^{s'} B$, where $k_i \in \{0, ..7\}$, $s' \in \{0, .., 84\}$. The values b_i, $i = 0, .., 252$ are binary representation of the scalar s.
$(u, v) = (b_{i+2} \cdot 2^2 + b_{i+1} \cdot 2 + b_i) \cdot B$.
$(x_{acc_0}, y_{acc_0}) = (u, v)$
$(x_{acc_{j+k}}, y_{acc_{j+k}}) = (u_{j+k}, v_{j+k}) + (x_{acc_{j+k-1}}, y_{acc_{j+k-1}})$
:

1. $b_i \cdot ((-u'_0 + u'_2 + u'_4 - u'_6 + u'_1 - u'_3 + u'_7 - u'_5) \cdot b_{i+2} \cdot b_{i+1} + (u'_0 - u'_1) \cdot b_{i+2} + (u'_0 - u'_2 - u'_4 - u'_1 + u'_3 + u'_5) \cdot b_{i+1} - u'_0 + u'_1) - (u - (u'_0 + u'_2 + u'_4 - u'_6) \cdot b_{i+1} \cdot b_{i+2} + u'_0 \cdot b_{i+2} + (u'_0 - u'_2 - u'_4) \cdot b_{i+1} - u'_0)$
2. $b_i \cdot ((-v'_0 + v'_2 + v'_4 - v'_6 + v'_1 - v'_3 + v'_7 - v'_5) \cdot b_{i+2} \cdot b_{i+1} + (v'_0 - v'_1) \cdot \cdot b_{i+2} + (v'_0 - v'_2 - v'_4 - v'_1 + v'_3 + v'_5) \cdot \cdot b_{i+1} - v'_0 + v'_1) - (v - (v'_0 + v'_2 + v'_4 - v'_6) \cdot b_{i+1} \cdot b_{i+2} + v'_0 \cdot b_{i+2} + (v'_0 - v'_2 - v'_4) \cdot b_{i+1} - v'_0)$
3. $x_2 \cdot (1 + d \cdot x_1 \cdot y_1 \cdot u \cdot v) - (u \cdot y_1 + v \cdot x_1)$
4. $y_2 \cdot (1 - d \cdot x_1 \cdot y_1 \cdot u \cdot v) - (u \cdot x_1 + v \cdot y_1)$

Per 3 bits b_{i+2}, b_{i+1}, b_i:

1. $b_i \cdot ((-u'_0 + u'_2 + u'_4 - u'_6 + u'_1 - u'_3 + u'_7 - u'_5) \cdot b_{i+2} \cdot b_{i+1} + (u'_0 - u'_1) \cdot b_{i+2} + (u'_0 - u'_2 - u'_4 - u'_1 + u'_3 + u'_5) \cdot b_{i+1} - u'_0 + u'_1) - (u - (u'_0 + u'_2 + u'_4 - u'_6) \cdot b_{i+1} \cdot b_{i+2} + u'_0 \cdot b_{i+2} + (u'_0 - u'_2 - u'_4) \cdot b_{i+1} - u'_0)$ (3 rows)
2. $b_i \cdot ((-v'_0 + v'_2 + v'_4 - v'_6 + v'_1 - v'_3 + v'_7 - v'_5) \cdot b_{i+2} \cdot b_{i+1} + (v'_0 - v'_1) \cdot \cdot b_{i+2} + (v'_0 - v'_2 - v'_4 - v'_1 + v'_3 + v'_5) \cdot \cdot b_{i+1} - v'_0 + v'_1) - (v - (v'_0 + v'_2 + v'_4 - v'_6) \cdot b_{i+1} \cdot b_{i+2} + v'_0 \cdot b_{i+2} + (v'_0 - v'_2 - v'_4) \cdot b_{i+1} - v'_0)$ (3 rows)
3. $t_0 = u \cdot v$ (11 rows)
4. $t_1 = x_1 \cdot y_1$ (11 rows)
5. $t_2 = t_0 \cdot t_1$ (11 rows)
6. $t_3 = d \cdot t_2$ (9 rows)
7. $t_4 = x_2 \cdot (1 + t_3)$ (11 rows)
8. $z_0 = u \cdot y_1$ (11 rows)
9. $z_1 = v \cdot x_1$ (11 rows)
10. $t_4 - z_1 == z_0$ (7 rows)
11. $c_0 = y_2 \cdot (1 - t_3)$ (11 rows)
12. $d_0 = u \cdot x_1$ (11 rows)
13. $d_1 = v \cdot y_1$ (11 rows)
14. $c_0 - d_1 == d_0$ (7 rows)

Totally, it costs 10880 rows.

Decomposition circuit The decomposition circuit is a specific function for SHA-512, which prepares output to the non-native variable base scalar multiplication. Let $\{k_0, k_1, k_2, ..., k_7\}$ be a SHA-512 output. Suppose that we want to constrain $k_0 + k_1 \cdot 2^{64} + ... + k_7 \cdot 2^{448} = L \cdot q + r$, where $L = 2^{252} + 27742317777372353535851937790883648493$. The size of each k_i is range-constrained by SHA-512 circuit. Since each degree of two can be reduced modulo L on the circuit definition's step, the value q is range-constrained by 2^67 and $t = 69$. Thus, the q decomposed to q_0, q_1, q_2, q_3, which corresponds to $20, 20, 20, 7$ bits.

Non-native decomposition circuit

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
j + 0	r_0	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
j + 1	r_9	r_{10}	r_{11}	r_{12}	r	ξ	c		
j + 2	q_0	q_1	q_2	q_3	v_0	v_1	v_2	v_3	
j + 3	k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	v

Let s_r be $(w_{0,j+0} + w_{1,j+0} + w_{2,j+0} + w_{3,j+0} + w_{4,j+0} + w_{5,j+0} + w_{6,j+0} + w_{7,j+0} + w_{8,j+0}, w_{0,j+1}, w_{1,j+1} + w_{2,j+1} - 12 \cdot (2^20 - 1))$.
Constraints:

- $w_{0,j+3}+w_{1,j+3}\cdot 2^{64}+w_{2,j+3}\cdot 2^{128}+w_{3,j+3}\cdot 2^{192}+w_{4,j+3}\cdot(2^{256} \mod k)+w_{5,j+3}\cdot(2^{320} \mod k)+w_{6,j+3}\cdot(2^{384} \mod k)+w_{7,j+3}\cdot(2^{448} \mod k)-(w_{0,j+2}\cdot 2^{47}+w_{1,j+2}\cdot 2^{27}+w_{2,j+2}\cdot 2^{7}+w_{3,j+2})\cdot L+(w_{4,j+1})=0$
- $w_{4,j+1} = w_{3,j+1} + w_{2,j+1} \cdot 2^{13} + w_{1,j+1} \cdot 2^{33} + w_{0,j+1} \cdot 2^{53} + w_{8,j+0} \cdot 2^{73} + w_{7,j+0} \cdot 2^{93} + w_{6,j+0} \cdot 2^{113} + w_{5,j+0} \cdot 2^{133} + w_{4,j+0} \cdot 2^{153} + w_{3,j+0} \cdot 2^{173} + w_{2,j+0} \cdot 2^{193} + w_{1,j+0} \cdot 2^{213} + w_{0,j+0} \cdot 2^{233}$
- $s_r \cdot (w_{5,j+1} \cdot s_r - 1) = 0$
- $w_{5,j+1} \cdot (s_r) + (1 - w_{5,j+1} \cdot s_r) \cdot w_{6,j+1} - 1 = 0$
- $w_{0,j+3} + w_{1,j+3} \cdot 2^{64} + (w_{0,j+2} \cdot 2^{47} + w_{1,j+2} \cdot 2^{27} + w_{2,j+2} \cdot 2^{7} + w_{3,j+2}) \cdot (-p \mod 2^t) - (w_{3,j+1} + w_{2,j+1} \cdot 2^{13} + w_{1,j+1} \cdot 2^{33} + w_{0,j+1} \cdot 2^{53}) = v \cdot 2^{69}$
- $w_{8,j+3} = w_{4,j+2} \cdot 2^{41} + w_{5,j+2} \cdot 2^{21} + w_{6,j+2} \cdot 2 + w_{7,j+2}$
- $(w_{8,j+2} - 1) \cdot w_{8,j+2} = 0$

Lookup constraints:

- $(w_{0,j+0}), (w_{1,j+0}), (w_{2,j+0}), (w_{3,j+0}), (w_{4,j+0}), (w_{5,j+0}), (w_{6,j+0}), (w_{7,j+0}), (w_{8,j+0}), (w_{0,j+1}), (w_{1,j+1}), (w_{2,j+1}), (w$ $2^7)$
- $(w_{0,j+2}), (w_{1,j+2}), (w_{2,j+2}), (w_{3,j+2}), (w_{3,j+2} \cdot 2^{13})$
- $(w_{4,j+2}), (w_{5,j+2}), (w_{6,j+2}), (w_{7,j+2})$

Variable-base scalar multiplication :

The values b_i, $i = 0, .., 252$ are binary representation of the scalar k'.
The values $(x_1, y_1) = A$.
$(x_2, y_2) = 2(b_{252} \cdot (x_1, y_1)) + b_{251} \cdot (x_1, y_1)$
$(x_i, y_i) = 2(x_{i-1}, y_{i-1}) + b_{253-i} \cdot (x_1, y_1)$, for $i \in \{3, ..., 253\}$
For (x_i, y_i) the following is checked:

1. $x_3 \cdot ((y_1^2 - x_1^2) \cdot (2 - y_1^2 + x_1^2) + 2dx_1y_1(y_1^2 + x_1^2) \cdot x_2y_2b) - (2x_1y_1 \cdot (2 - y_1^2 + x_1^2) \cdot (y_2b + (1 - b)) + (y_1^2 + x_1^2) \cdot (y_1^2 - x_1^2) \cdot x_2b)$
2. $y_3 \cdot ((y_1^2 - x_1^2) \cdot (2 - y_1^2 + x_1^2) - 2dx_1y_1(y_1^2 + x_1^2) \cdot x_2y_2b) - (2x_1y_1 \cdot (2 - y_1^2 + x_1^2) \cdot x_2b + (y_1^2 + x_1^2) \cdot (y_1^2 - x_1^2) \cdot (y_2b + (1 - b)))$

This can be implemented in the following algorithm:

1. $t_0 = (y_1^2 - x_1^2)$. (11 rows)
2. $t_1 = (2 - y_1^2 + x_1^2)$. (11 rows)
3. $t_2 = (t_0 \cdot t_1)$. (11 rows)
4. $t_3 = (y_1^2 + x_1^2)$. (11 rows)
5. $t_4 = 2 \cdot x_1 \cdot y_1$. (11 rows)
6. $t_5 = b \cdot x_2 \cdot y_2$. (11 rows)
7. $t_6 = t_3 \cdot t_4$. (11 rows)
8. $t_7 = t_6 \cdot t_3$. (11 rows)
9. $t_8 = d \cdot t_7$. (9 rows)
10. $t_9 = (t_8 + t_2) \cdot x_3$. (13 rows)
11. $z_0 = t_4 \cdot t_1$. (11 rows)
12. $z_1 = z_0 \cdot (y_2 \cdot b + (1 - b))$. (11 rows)
13. $z_2 = t_3 \cdot t_0$. (11 rows)
14. $z_3 = b \cdot z_2 \cdot x_2$. (11 rows)
15. $t_9 - z_3 == z_1$. (7 rows)
16. $c_0 = y_3 \cdot (t_2 - t_8)$. (13 rows)
17. $d_0 = b \cdot z_0 \cdot x_2$. (11 rows)
18. $d_1 = z_2 \cdot (y_2 \cdot b + (1 - b)))$. (11 rows)
19. $c_0 - d_0 == d_1$. (7 rows)

Thus, it costs 203 rows per bit. Totally, it is $50953 + 4$ rows.

2.5.8 Redshift Verification

WIP

Redshift circuit repeats all steps from Section 2.4.2. The verification circuit is a part of bridge design, and it is supposed that any output of the basic proof is an input to the verification circuit. Thus, we do not suppose any decoding for the proof because it can be represented directly in the desirable form.

In the previous sections, we described circuits for most of the steps of the verifier algorithm. However, steps 15-16 require additional clarification.

We consider step 16 firstly as a simpler one. It contains basic arithmetic operations over finite field elements. These operations can be done with standard generic PLONK gate:

$$\mathbf{q}_L \cdot w_0 + \mathbf{q}_R \cdot w_1 + \mathbf{q}_M \cdot w_0 \cdot w_1 + \mathbf{q}_O \cdot w_2 + \mathbf{q}_C$$

There are more optimal ways to perform these calculations. However, the number of arithmetic operations is much less than in Step 15. It means that any optimizations do not decrease prover or verifier complexities in any noticeable way.

FRI Verification is the main part of Step 15. It contains two operations: Merkle tree path check and polynomial interpolation. The circuit version of Merkle path check algorithm does not differ from the original one. The circuit form Section 2.5.4 is used to check hash operations correctness.

To check polynomial interpolation, the following circuit is used:

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
j + 0	a_0	a_1	s_0	s_1	x	y	α	β	...

Constraints (**max degree** $= 2$):

1. $w_6 \cdot w_0 + w_7 = w_2 \longleftrightarrow \alpha \cdot a_0 + \beta = s_0$
2. $w_6 \cdot w_0 + w_7 = w_2 \longleftrightarrow \alpha \cdot a_1 + \beta = s_1$
3. $w_6 \cdot w_0 + w_7 = w_2 \longleftrightarrow \alpha \cdot x + \beta = y$

Copy constraints:

1. a_0, a_1, s_0, s_1, y are constrained by public input.

The gate uses the line equation to check that all three points are on the same line. This means, it checks $f(a_0) = s_0$, $f(a_1) = s_1$, $f(x) = y$ for $f(X) = \alpha \cdot X + \beta$.

2.5.9 Validator Set Proof Circuit

WIP

Chapter 3

In-EVM State Proof Verifier

This introduces a description for Solana's 'Light-Client' state proof in-EVM verifier. 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.