

In-EVM Mina State Verification Proof System Description

Cherniaeva Alisa

a.cherniaeva@nil.foundation

=nil; Crypto3 (<https://crypto3.nil.foundation>)

Shirobokov Ilia

i.shirobokov@nil.foundation

=nil; Crypto3 (<https://crypto3.nil.foundation>)

October 30, 2021

1 Introduction

WIP

To prove Mina blockchain's state on the Ethereum Virtual Machine, we use Redshift SNARK[1]. RedShift is a transparent SNARK that uses PLONK[2] proof system but replaces the commitment scheme. The authors utilize FRI[3] protocol to obtain transparency for the PLONK system.

However, FRI cannot be straightforwardly used with the PLONK system. To achieve the required security level without huge overheads, the authors introduce *list polynomial commitment* scheme as a part of the protocol. For more details, we refer the reader to [1].

The original RedShift protocol utilizes the classic PLONK[2] system. To provide better performance, we generalize the original protocol for use with PLONK with custom gates [4], [5] and lookup arguments [6], [7].

2 RedShift Protocol

WIP

Notations:

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

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

Preprocessing:

-
1. $\mathcal{L}' = (\mathbf{q}_0, \dots, \mathbf{q}_{N_{\text{sel}}})$
 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_{perm} permutation polynomials $S_{\sigma_i}(X)$ such that $S_{\sigma_i}(\omega^j) = \delta^{i'} \cdot \omega^{j'}$
 5. Compute N_{perm} identity permutation polynomials: $S_{id_i}(X)$ such that $S_{id_i}(\omega^j) = \delta^i \cdot \omega^j$
 6. Let $H = \{\omega^0, \dots, \omega^n\}$ be a cyclic subgroup of \mathbb{F}^*
 7. Let $Z(X) = \prod a \in H^*(X - a)$
-

Protocol (Prover):

1. Choose masking polynomials:

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

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

2. Define new witness polynomials:

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

3. Send commitments to f_i to \mathbf{V}
4. Get $\beta, \gamma \leftarrow \mathbb{F}$ from \mathbf{V}
5. For $0 \leq i < N_{\text{perm}}$

$$\begin{aligned} p_i &= f_i + \beta \cdot S_{id_i} + \gamma \\ q_i &= f_i + \beta \cdot S_{\sigma_i} + \gamma \end{aligned}$$

6. Define:

$$\begin{aligned} p'(X) &= \prod_{0 \leq i < N_{\text{perm}}} p_i(X) \in \mathbb{F}_{<N_{\text{perm}} \cdot n}[X] \\ q'(X) &= \prod_{0 \leq i < N_{\text{perm}}} q_i(X) \in \mathbb{F}_{<N_{\text{perm}} \cdot n}[X] \end{aligned}$$

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

$$\begin{aligned} P(\omega) &= Q(\omega) = 1 \\ P(\omega^i) &= \prod_{1 \leq j < i} p'(\omega^j) \text{ for } i \in 2, \dots, n+1 \\ Q(\omega^i) &= \prod_{1 \leq j < i} q'(\omega^j) \text{ for } i \in 2, \dots, n+1 \end{aligned}$$

8. Compute and send commitments to P and Q to \mathbf{V}
9. Get $\alpha_0, \dots, \alpha_5 \leftarrow \mathbb{F}$ from \mathbf{V}
10. Define polynomials $(F_0, \dots, F_4 - \text{copy-satisfiability})$:

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

11. Compute:

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

$$T(X) = \frac{F(X)}{Z(X)}$$

12. Split $T(X)$ into separate polynomials $T_0(X), \dots, T_{N_{\text{perm}}}(X)$

13. Send commitments to $T_0(X), \dots, T_{N_{\text{perm}}}(X)$ to \mathbf{V}

14. Get $y \leftarrow \mathbb{F}/H$ from \mathbf{V}

15. Run evaluation scheme with the committed polynomials and y

16. Send proof π to \mathbf{V}

2.1 Non-Interactive Verification

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

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

3. $\beta, \gamma = H(\text{transcript})$

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

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

6. $\alpha_0, \dots, \alpha_5 = H(\text{transcript})$

7. Let $T_{0,\text{comm}}, \dots, T_{N_{\text{perm}},\text{comm}}$ be commitments to $T_0(X), \dots, T_{N_{\text{perm}}}(X)$

8. $\text{transcript} = \text{transcript} || T_{0,\text{comm}} || \dots || T_{N_{\text{perm}},\text{comm}}$

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

10. Run evaluation scheme verification with the committed polynomials and y to get values $f_i(y), P(y), P(y\omega), Q(y), Q(y\omega), T_j(y)$.

Remark: Depending on the circuit, evaluation can be done also on $f_i(y\omega), f_i(y\omega^{-1})$ for some i .

11. Calculate:

$$F_0(y) = L_1(y)(P(y) - 1)$$

$$F_1(y) = L_1(y)(Q(y) - 1)$$

$$p'(y) = \prod p_i(y) = \prod f_i(y) + \beta \cdot S_{id_i}(y) + \gamma$$

$$F_2(y) = P(y)p'(y) - P(y\omega)$$

$$q'(y) = \prod q_i(y) = \prod f_i(y) + \beta \cdot S_{\sigma_i}(y) + \gamma$$

$$F_3(y) = Q(y)q'(y) - Q(y\omega)$$

$$F_4(y) = L_n(y)(P(y\omega) - Q(y\omega))$$

$$F_5(y) = \sum_{0 \leq i < N_{\text{sel}}} (\mathbf{q}_i(y) \cdot \text{gate}_i(y)) + \sum_{0 \leq i < N_{\text{const}}} (\mathbf{f}_{c_i}(y)) + PI(y)$$

$$T(y) = \sum_{0 \leq j < N_{\text{perm}}+1} y^{n \cdot j} T_j(y)$$

12. Check the identity:

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

3 Optimizations

WIP

References

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>.