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 28, 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 generilize the original protocol for use with PLONK with custom gates [4], [5] and lookup arguments [6], [7].

2 RedShift Protocol

WIP

Notations:

- $N_{\texttt{wires}}$ is the number of wires ('advice columns').
- $N_{\texttt{perm}}$ is the number of wires that are included in the permutation argument
- $N_{\texttt{sel}}$ is the number of selectors used in the circuit
- $N_{\texttt{const}}$ is the number of constant columns
- Permutation over the table: $\sigma(\text{column} : i, \text{row} : j) = (\text{column} : i', \text{row} : j')$
- $\mathbf{f}_i$ is witness polynomial for $0 \leq i < N_{\texttt{wires}}$
- $\mathbf{f}_{c_i}$ is constant-related polynomial for $0 \leq i < N_{\texttt{wires}}$
- $\mathbf{gate}_i$ is gate polynomial for $0 \leq i < N_{\texttt{const}}$

`adv` be the number of advice columns, τ be the number the fixed columns. Let .

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

Preprocessing:

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 $S_{id_i}(\omega^j) = \delta^i \cdot \omega^j$

Protocol:

1. **P**:

   1.1 Choose masking polynomials:

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

   1.2 Define new witness polynomials:

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

2. **V**:

   2.1 Send to **P**: $\beta, \gamma \leftarrow \mathbb{F}$

3. **P**:

   3.1 Compute for $0 \leq j < N_{\texttt{perm}}$

   $$p_j = f_j + \beta \cdot S_{id_j} + \gamma$$
   $$q_j = f_j + \beta \cdot S_{\sigma_j} + \gamma$$

   3.2 Define:

   $$p'(X) = \prod_{0 \leq j < N_{\texttt{perm}}} p_j(X) \in \mathbb{F}_{<N_{\texttt{perm}} \cdot n}[X]$$
   $$q'(X) = \prod_{0 \leq j < N_{\texttt{perm}}} q_j(X) \in \mathbb{F}_{<N_{\texttt{perm}} \cdot n}[X]$$

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

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

   3.4 Compute and send commitments to P and Q to **V**

4. **V**:

   4.1 Send to **P**: $a_1, \ldots, a_6 \leftarrow \mathbb{F}$

5. **P**:

   5.1 Define polynomials ($F_1, \ldots, F_5$ - copy-satisfability):

   $$F_1(x) = L_1(x)(P(x) - 1)$$
   $$F_2(x) = L_1(x)(Q(x) - 1)$$
   $$F_3(x) = P(x)p'(x) - P(xg)$$
   $$F_4(x) = Q(x)q'(x) - Q(xg)$$
   $$F_5(x) = L_n(x)(P(xg) - Q(xg))$$
   $$F_6(x) = \sum_{0 \leq i < N_{\texttt{sel}}} (\mathbf{q}_i(x) \cdot \texttt{gate}_i(x)) + (\sum_{0 \leq i < N_{\texttt{const}}} (\mathbf{f}_{c_i}(x)) + PI(x))$$

   5.2 Compute:

   $$F(x) = \sum_{i=1}^{6} a_i F_i(x)$$
   $$T(x) = \frac{F(x)}{Z(x)}$$

   5.3 Split $T(x)$ into seprate polynomials $T_0(x), ..., T_{N_{\texttt{perm}}+1}$

   5.4 Send commitment to $T_0(x), ..., T_{N_{\texttt{perm}}+1}$ to **V**

6. **V**:

   6.1 Send to **P**: $y \leftarrow \mathbb{F}/H$

7. **P**:

8. Run evaluation scheme over committed polynomials and y

9. **V**

   9.1 Checks the identity:

$$\sum_{i=1}^{6} a_i F_i(y) = Z(y)T(y)$$

3 RedShift Verification

WIP

4 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.