In-EVM Solana 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 29, 2021

1 Introduction

WIP

To prove Solana 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:

$N_{\texttt{wires}}$	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
$\mathbf{f}_i$	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].

Preprocessing:

1. $\mathcal{L}' = (\mathbf{q}_0, ..., \mathbf{q}_{N_{\mathtt{sel}}})$
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 $N_{\mathtt{perm}}$ permutation polynomials $S_{\sigma_i}(X)$ such that $S_{\sigma_i}(\omega^j) = \delta^{i'} \cdot \omega^{j'}$
5. Compute $N_{\mathtt{perm}}$ identity permutation polynomials: $S_{id_i}(X)$ such that $S_{id_i}(\omega^j) = \delta^i \cdot \omega^j$

Protocol (Prover):

1. Choose masking polynomials:

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

2. Define new witness polynomials:

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

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

$$\begin{aligned} p_j &= f_j + \beta \cdot S_{id_j} + \gamma \\ q_j &= f_j + \beta \cdot S_{\sigma_j} + \gamma \end{aligned}$$

6. Define:

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

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

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

8. Compute and send commitments to P and Q to $\mathbf{V}$
9. Get $a_1, \ldots, a_6 \leftarrow \mathbb{F}$ from $\mathbf{V}$
10. Define polynomials ($F_1, \ldots, F_5$ - copy-satisfability):

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

11. Compute:

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

12. Split $T(x)$ into seprate polynomials $T_0(x), ..., T_{N_{\texttt{perm}}+1}$
13. Send commitments to $T_0(x), ..., T_{N_{\texttt{perm}}+1}$ to **V**
14. Get **P** $y \leftarrow \mathbb{F}/H$ from **V**
15. Run evaluation scheme with the committed polynomials and y
16. **V** checks the identity:

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

2.1 Non-Interactive Verification

1. Let $f_{0,\texttt{comm}}, \ldots, f_{N_{\texttt{wires}},\texttt{comm}}$ be commitments to $f_0, \ldots, f_{N_{\texttt{wires}}}$
2. transcript = setup_values$||f_{0,\texttt{comm}}|| \ldots ||f_{N_{\texttt{wires}},\texttt{comm}}$
3. $\beta, \gamma = H(\text{transcript})$
4. Let $P_{\texttt{comm}}, Q_{\texttt{comm}}$ be commitments to $P(X), Q(X)$
5. transcript = transcript$||P_{\texttt{comm}}||Q_{\texttt{comm}}$
6. $a_1, \ldots, a_6 = H(\text{transcript})$
7. Let $T_{0,\texttt{comm}}(x), ..., T_{N_{\texttt{perm},\texttt{comm}}+1}$ be commitments to $T_0(x), ..., T_{N_{\texttt{perm}}+1}$
8. transcript = transcript$||T_{0,\texttt{comm}}(x)||...||T_{N_{\texttt{perm},\texttt{comm}}+1}$
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:

$$\begin{gathered}
F_1(y) = L_1(y)(P(y) - 1) \\
F_2(y) = L_1(y)(Q(y) - 1) \\
p'(y) = \prod p_i(y) = \prod f_i(y) + \beta \cdot S_{id_i}(y) + \gamma \\
F_3(y) = P(y)p'(y) - P(y\omega) \\
q'(y) = \prod q_i(y) = \prod f_i(y) + \beta \cdot S_{\sigma_i}(y) + \gamma \\
F_4(y) = Q(y)q'(y) - Q(y\omega) \\
F_5(y) = L_n(y)(P(y\omega) - Q(y\omega)) \\
F_6(y) = \sum_{0 \le i < N_{\text{sel}}} (\mathbf{q}_i(y) \cdot \texttt{gate}_i(y)) + (\sum_{0 \le i < N_{\text{const}}} (\mathbf{f}_{c_i}(y)) + PI(y)) \\
T(y) = \sum_{0 \le j < N_{\text{perm}}} y^{n \cdot j} T_j(y)
\end{gathered}$$

12. Check the identity:

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

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.