The Journal of Open Source Software

# solana-pqzk-fullchain: Full on-chain verification of ZK-STARK proofs and post-quantum signatures on Solana

**Jotaro Yano** [1,¶]

**1** Independent Researcher, Japan ¶ Corresponding author

**DOI:** 10.xxxxxx/draft

**Software**

- Review
- Repository
- Archive

**Editor:** Neea Rusch

**Reviewers:**

- @ravikanthreddy89
- @amitkumarj441
- @Abinashbunty

**Submitted:** 09 December 2025
**Published:** unpublished





## Summary

`solana-pqzk-fullchain` is an open-source, research-oriented reference implementation that demonstrates **fully on-chain verification** of (i) a **hash-based zero-knowledge proof** (a STARK) and (ii) a **post-quantum digital signature** (SLH-DSA / SPHINCS+) on **Solana L1** (Yano, 2025b). The repository contains an on-chain verifier program (Anchor/SBF), a minimal off-chain STARK prover (Winterfell), a TypeScript CLI demo that runs an end-to-end flow (encrypt → prove → sign → upload → finalize → verify → receive), and benchmarking utilities that record transaction-level compute units (CUs) for each verification phase.

## Statement of need

Public blockchains preserve artifacts indefinitely, which creates a "record now, break later" risk profile for protocols whose security relies on assumptions that may be weakened by future capabilities, including Shor-type quantum attacks on discrete-log based systems (Shor, 1999). In practice, many deployed succinct proof systems use pairing-based SNARKs (e.g., Groth16, PLONK) because of small proofs and fast verification (Gabizon et al., 2019; Groth, 2016). In contrast, STARKs are transparent (no trusted setup) and primarily hash-based, making them attractive when long-term post-quantum orientation is a design goal (Ben-Sasson et al., 2018b, 2018a). The trade-off is that STARK verification is typically hashing-heavy and proof sizes are larger, so engineering feasibility depends strongly on the target execution environment.

Solana is a useful platform for studying this feasibility because it exposes explicit transaction compute accounting and strict runtime constraints (compute-unit limits, bounded stack, explicit heap-frame requests, and transaction/instruction size limits) that directly shape what can be verified on L1 (Solana Foundation, 2025b, 2025a, 2025c). However, implementing a full verifier pipeline on Solana L1 is non-trivial: hashing costs dominate, stack pressure can trigger SBF failures, memory allocation must be controlled, and large inputs require DoS- and fee-aware streaming.

`solana-pqzk-fullchain` addresses this gap by providing a complete, reproducible software stack focused on **engineering practicality** rather than proposing a new proof system:

- **CPI-friendly on-chain verifier surface:** a Solana program that verifies an **SLH-DSA (FIPS 205) signature** and a **Winterfell STARK proof** in separate phases to enable early rejection of malformed or unauthorized payloads (Meta, 2025; National Institute of Standards and Technology, 2024b; Yano, 2025b).
- **Binding of proofs to application artifacts:** the verifier derives public inputs from `SHA256(ciphertext)` and verifies a minimal AIR (affine counter) as a baseline mechanism for binding a proof to the uploaded ciphertext (Yano, 2025b).

- **DoS- and fee-aware streaming uploads:** payloads are uploaded in bounded chunks with fixed-offset appends and a rolling SHA-256 hash chain, so invalid uploads can be rejected with constant work per chunk (Yano, 2025b).
- **Runtime-aware adaptations for Solana SBF:** patched components route SHA-256 hashing through Solana's hashv syscall, suppress inlining in FRI hotspots to respect stack limits, and use a bump allocator synchronized to the requested heap frame for predictable memory behavior (Yano, 2025b).
- **End-to-end demo and benchmarking:** the repository includes a demo encryption path using a Kyber768/ML-KEM-style KEM for deriving an AEAD key (HKDF-SHA256) and AES-256-GCM encryption, plus scripts that repeatedly run the pipeline and log per-transaction compute units for the verification phases (National Institute of Standards and Technology, 2024a; Yano, 2025b).

This package is intended for researchers and engineers who need a reproducible baseline to (a) evaluate the feasibility and cost of PQ-oriented verification on Solana L1, (b) experiment with engineering levers (hashing path, stack discipline, heap sizing, streaming I/O), and (c) integrate a verifier via CPI into other Solana programs.

A longer methods-and-measurement report describing the same artifacts is available as a preprint on Zenodo and IACR ePrint; the JOSS paper intentionally focuses on the software contribution, interfaces, and reproducibility rather than new scientific findings (Yano, 2025a, 2025c).

# Software design

The software is organized as an end-to-end reference stack for Solana L1: an on-chain verifier program (Anchor/SBF), an off-chain prover workflow, a CLI demo, and benchmark scripts. Verification is split into two phases—(1) SLH-DSA signature verification and (2) STARK proof verification—to reject unauthorized or malformed payloads before paying the higher STARK verification cost. Large inputs are handled via bounded, chunked uploads with constant work per chunk (including a rolling SHA-256 chain) to fit Solana transaction/account limits and improve predictability under adversarial inputs. To operate within Solana's SBF constraints, the implementation routes SHA-256 hashing through Solana's hashv syscall and manages stack/heap usage in line with requested heap frames.

Build vs. contribute: the cryptographic primitives come from existing standards and libraries; the contribution is the Solana-specific integration and reproducible workflow (program + client + benchmarks) that shows how to run these components under Solana's execution and transaction constraints.

# Research impact statement

This project provides credible near-term significance as a reproducible baseline for evaluating post-quantum–oriented verification on Solana L1. At the time of writing, we are not aware of many publicly available Solana L1 reference implementations that verify both a STARK proof and an NIST-standard post-quantum signature (SLH-DSA, FIPS 205) fully on-chain in a single end-to-end pipeline. The repository includes a runnable CLI demo and benchmark utilities that record transaction-level compute units for each verification phase, enabling others to reproduce results and compare engineering trade-offs (hashing strategy, memory settings, and I/O chunking). The documented adaptations for the Solana SBF environment are intended to help developers who are considering STARK verification or PQ signatures, and to support future CPI-friendly reuse.

## AI usage disclosure

Generative AI (ChatGPT) was used in a limited way. The core software design and implementation were created by the author. ChatGPT was occasionally used for small code snippets/boilerplate (e.g., helper functions or simple CLI handling); all AI-suggested code was reviewed, edited, and tested by the author. For the paper and documentation, the technical content and structure were written by the author, and ChatGPT was used to improve English wording. The author reviewed and corrected the final manuscript and takes responsibility for the code and paper.

## Acknowledgements

This project builds on the Winterfell STARK ecosystem (Meta, 2025) and references the NIST post-quantum standards for ML-KEM and SLH-DSA (National Institute of Standards and Technology, 2024a, 2024b). It also relies on Solana's public documentation for compute budgeting and transaction constraints (Solana Foundation, 2025b, 2025a, 2025c). No specific financial support was received for this work. The author declares no conflicts of interest.

## References

Ben-Sasson, E., Bentov, I., Horesh, Y., & Riabzev, M. (2018a). Fast reed-solomon interactive oracle proofs of proximity. *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. https://doi.org/10.4230/LIPIcs.ICALP.2018.14

Ben-Sasson, E., Bentov, I., Horesh, Y., & Riabzev, M. (2018b). *Scalable, transparent, and post-quantum secure computational integrity*. Cryptology ePrint Archive, Report 2018/046. https://eprint.iacr.org/2018/046

Gabizon, A., Williamson, Z. J., & Ciobotaru, O. (2019). *PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge*. IACR Cryptology ePrint Archive, Report 2019/953. https://eprint.iacr.org/2019/953

Groth, J. (2016). On the size of pairing-based non-interactive arguments. *Advances in Cryptology – EUROCRYPT 2016*, *9665*, 305–326. https://doi.org/10.1007/978-3-662-49896-5_11

Meta. (2025). *Winterfell: A STARK prover and verifier for arbitrary computations* (Version 0.12.0). https://github.com/facebook/winterfell

National Institute of Standards and Technology. (2024a). *Module-lattice-based key-encapsulation mechanism standard (FIPS 203)*. NIST. https://doi.org/10.6028/NIST.FIPS.203

National Institute of Standards and Technology. (2024b). *Stateless hash-based digital signature standard (FIPS 205)*. NIST. https://doi.org/10.6028/NIST.FIPS.205

Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, *41*(2), 303–332. https://doi.org/10.1137/S0036144598347011

Solana Foundation. (2025a). *How to request optimal compute budget*. https://solana.com/developers/guides/advanced/how-to-request-optimal-compute

Solana Foundation. (2025b). *Transaction fees — compute units and limits*. https://solana.com/docs/core/fees

Solana Foundation. (2025c). *Transactions and instructions — transaction size limit*. https://solana.com/docs/core/transactions

Yano, J. (2025a). *Full L1 on-chain ZK-STARK+PQC verification on solana: A measurement study*. IACR Cryptology ePrint Archive, Report 2025/1741. https://eprint.iacr.org/2025/1741

Yano, J. (2025b). *Pqzk-labs/solana-pqzk-fullchain: Full on-chain ZK-STARK + PQC verification on solana* (Version v0.1.2). https://github.com/pqzk-labs/solana-pqzk-fullchain

Yano, J. (2025c). *Full L1 on-chain ZK-STARK+PQC verification on solana: A measurement study*. Zenodo. https://doi.org/10.5281/zenodo.17186800