Proving Code Execution On-Chain: Zk-Coprocessors for Decentralized Applications

Santa Clara University Blockchain Club

Arthur Guiot
Director of Engineering
aguiot@scu.edu

Zach Teal
Director of Research
zteal@scu.edu

February 7, 2025

Abstract

Blockchain networks are often limited by on-chain computation constraints and high gas costs, forcing protocol developers to compromise on decentralization or complexity. In this paper, we propose and analyze the use of zero-knowledge (ZK) coprocessors, focusing on solutions such as RISC0, SP1, and Axiom, to enable verifiable off-chain computation. We explore the technical fundamentals of zkSNARKs, zkSTARKs, and RISC-V-based ZK virtual machines, while also examining alternative approaches such as trusted execution environments (TEEs) and secure multi-party computation (MPC). We provide usage examples ranging from dynamic lending controllers in DeFi to verifiable margining for derivatives. By presenting concrete details on memory, proof size, gas considerations, and system constraints, we illustrate how these systems unlock new classes of applications for Ethereum and other blockchain networks while preserving trustlessness.

1 Introduction

Blockchain technology promises decentralized, trustless computation. However, real-world protocols face challenges when implementing advanced functionality on-chain. Networks like Ethereum incur high costs and low throughput for heavy computation, forcing developers to move complex logic off-chain. This move can compromise trust assumptions, effectively introducing centralization risks.

Zero-knowledge (ZK) proof systems offer a cryptographic means to bridge this gap. By performing computation off-chain and then submitting succinct proofs of correct execution on-chain, we preserve decentralization while massively expanding computational capabilities. Recent advances in general-purpose zero-knowledge architectures (e.g., RISC0, SP1, Axiom) provide templates for building "zk-coprocessors" analogous to hardware coprocessors in traditional CPUs.

In this paper, we:

- Survey the background of zero-knowledge proofs in blockchain.
- Highlight RISC0, SP1, and Axiom as key ZK-coprocessor solutions, alongside alternative models such as TEEs and MPC.
- Present an expanded system design for verifiable off-chain computation, focusing on practical aspects such as memory models, proof sizes, and gas constraints.
- Demonstrate usage examples, including dynamic lending controllers, on-chain risk analytics, and verifiable derivatives margining.

- Discuss open challenges around performance, trust models, and next steps in decentralized coprocessor adoption.

2 Background

2.1 Zero-Knowledge Proofs in Blockchain

Zero-knowledge proofs (ZKPs) allow a prover to convince a verifier of a statement's truth without revealing any additional information. Their application in blockchains has evolved significantly, mainly in the forms of zkSNARKs and zkSTARKs.

zkSNARKs (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) offer very small proof sizes (on the order of 128 bytes) but often rely on a trusted setup. This setup can be a single-ceremony event that introduces additional trust assumptions.

zkSTARKs (Zero-Knowledge Scalable Transparent Argument of Knowledge) remove the need for a trusted setup, typically at the cost of larger proofs (tens of kilobytes) and higher gas costs for on-chain verification. They leverage collision-resistant hash functions and are considered post-quantum secure.

2.2 RISC0 and SP1 Technical Foundation

RISC0 is a general-purpose zkSTARK-based proving system designed around a RISC-V architecture, specifically `riscv32im-risc0-zkvm-elf`. This choice balances computational expressiveness and proving efficiency. The memory model relies on segmented pages that enable proofs to scale to large programs while staying tractable. RISC0 implements a maximum of 2^{32} cycles per segment, permitting substantial off-chain computation.

SP1 is an augmented version of the RISC0 architecture focusing on expanding computational capacity and introducing advanced pre-compiled circuits dedicated to specific tasks such as cryptographic acceleration or specialized arithmetic. These modifications aim to push performance boundaries, reducing proof generation overhead while still producing succinct proofs.

2.3 Axiom: A ZK Coprocessor for On-Chain Data Access

Axiom focuses on off-chain computation that accesses large swaths of historical on-chain data. Ethereum contracts typically cannot query extensive historical data without incurring prohibitive costs. Axiom nodes retrieve such data, perform computations off-chain, and generate zk-proofs for on-chain verification. This approach has unlocked use cases like auditing historical token transfers, building advanced on-chain analytics, and verifying contract states across large time windows.

2.4 Alternative Solutions: MPC and TEEs

While ZK-coprocessors are powerful, other trust-minimized off-chain computation models exist:

MPC (Multi-Party Computation) enables multiple parties to jointly compute functions over private inputs. Though powerful for privacy-preserving collaborations, MPC can have higher communication overhead and typically lacks the succinct, easily verifiable proof model of ZK. Fairness and correctness are also more challenging to guarantee if the majority of participants are corrupted.

TEEs (Trusted Execution Environments) like Intel SGX and AWS Nitro Enclaves use secure hardware to protect computations from external inspection or tampering. TEEs can be easier to integrate, but they require trusting hardware vendors and effectively centralize security in these environments. They do not inherently produce succinct proofs for on-chain verification, although attestation can be used to provide some trust signals.

3 System Design

3.1 High-Level Architecture

Figure 1 conceptually illustrates a zk-coprocessor architecture:

1. **Proof Generator (Off-Chain):** Executes arbitrary computations using a RISC-V-based zkVM (e.g., RISC0, SP1). It produces a proof attesting to the correctness of the computation.

2. **Verification Smart Contract (On-Chain):** Receives the proof, verifies it, and updates on-chain state accordingly.

3. **Executor Environment and Guest Code:** The guest code is compiled for the specialized RISC-V target, ensuring determinism. Public data is committed into a proof journal, while private data is excluded from the on-chain proof.

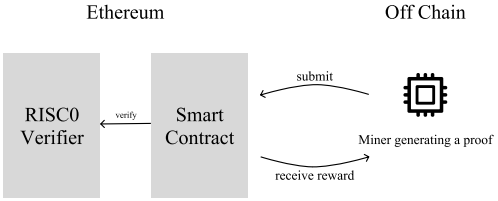

Figure 1: High-Level ZK-Coprocessor Architecture

The host environment orchestrates proof generation, data provisioning, and calls to the blockchain for final verification. This separation ensures that even large or complex computations remain efficient off-chain, with only a succinct proof posted on-chain.

3.2 Proof Generation and Verification

During proof generation, the zkVM interprets every instruction (load, store, arithmetic, branching) while building a cryptographic commitment to the execution trace. The final proof is then combined with any necessary public inputs (journal data) and submitted on-chain to a verification contract.

Equation (1) provides a simplified representation of a zkSNARK verification check:

$$e\big(A_1, B_1\big) \cdot e\big(A_2, B_2\big) \stackrel{?}{=} e\big(C, D\big), \tag{1}$$

where $e(\cdot)$ is a bilinear pairing, $\big(A_1, B_1\big)$ and $\big(A_2, B_2\big)$ represent the proof elements, and $\big(C, D\big)$ is a public verifier key component. Though abstract, it highlights the fundamental group operations that must be performed on-chain. For zkSTARKs, the verification process uses polynomial IOPs (Interactive Oracle Proofs) and typically involves hash-based Merkle commitments.

3.3 Smart Contract Design

A minimal two-contract architecture can be used:

- **Application Contract:** Contains business logic or state updates contingent on proof verification.

- **Verification Contract:** A library or contract (e.g., RISC0 or Axiom-provided) that validates the zk-proof. It returns a boolean success/failure to the Application Contract.

Since proof verification can cost hundreds of thousands of gas on Ethereum L1, many developers opt to deploy verification on an L2 chain (e.g., Arbitrum, Optimism, Base) for cheaper on-chain operations. This can significantly lower overhead, but introduces cross-layer latency for state finality.

4 Use Cases and Examples

4.1 Dynamic Lending Controllers in DeFi

Lending protocols can dynamically update interest rates or collateral factors based on real-time on-chain data. By using a zk-coprocessor, the system can retrieve liquidity and utilization parameters, run a

control algorithm off-chain (potentially even an RL-based or advanced model), and post a succinct proof on-chain that the updates are derived honestly.

For instance, if U denotes the utilization ratio:

$$U = \frac{\text{Total Borrowed}}{\text{Total Liquidity}},$$

a feedback controller might set interest rates r as follows:

$$r = r_0 + k_p(U - U_{\text{target}}),$$

where r_0 is a base rate, and k_p is a proportional parameter. The zk-coprocessor ensures that the off-chain computations (e.g., reading historical data or performing BFS-like calculations for liquidity modeling) are done correctly. The final interest rate adjustment is published and verified on-chain, removing the trust assumption of a centralized off-chain engine.

4.2 Machine Learning-Based Risk Scoring

Smart contract systems often rely on up-to-date risk profiles of users or tokens. A protocol could, for example, run an ML model that analyzes on-chain address histories to classify risk levels for credit or lending. The model might be complex (e.g., a small neural network or gradient boosting pipeline), thus expensive to run fully on-chain. A zk-coprocessor would:

1. Load the model weights and user address data.
2. Compute a risk score off-chain (in a zero-knowledge environment).
3. Generate a proof that the model was executed correctly against specific parameters.
4. Submit the proof on-chain, which can then conditionally update an address's borrowing limit.

Potential challenges include large model sizes, which increase proof generation time. Continuations or parallel proving (through scalable GPU based proving services like Bonsai) can mitigate these concerns.

4.3 Verifiable Derivatives Margining

Perpetual swaps and options require continuous position monitoring to prevent undercollateralized accounts. Centralized exchanges implement margin calculations behind closed doors, whereas a zk-coprocessor can make these calculations transparent and trustless:

- Off-chain engine reads positions and price data, calculates margin requirements and potential liquidations.
- A proof is generated that the margin logic is correct, referencing the on-chain oracle price feed.
- The proof is submitted to an on-chain contract, which triggers liquidation if certain thresholds are met.

This architecture preserves user trust and removes the black-box nature of margin calls.

5 Comparisons with Other Off-Chain Models

5.1 MPC

Although multi-party computation can enable privacy-preserving analytics, verifying the final result on-chain in a succinct way is non-trivial. MPC typically lacks succinct proof structures, and if many MPC participants are corrupted, correctness can fail. ZK-coprocessors offer a single-prover setup with a single succinct proof, more straightforward for on-chain verification.

5.2 TEEs

Trusted execution environments rely on hardware attestation rather than cryptographic zero-knowledge proofs. While TEEs are extremely fast, they introduce vendor trust assumptions and do not produce easily verifiable proofs. For certain lower-stakes computations, TEEs may be acceptable due to speed and simplicity. But in a fully trustless environment, ZK-coprocessors often represent a more secure solution.

6 Open Challenges and Future Work

- **Performance and Scalability:** While RISC0 and SP1 push the boundaries of general-purpose ZK computation, further optimization is needed. Large-scale ML or real-time analytics remain expensive in proof generation.
- **Storage and Data Availability:** Off-chain data feeding into computations may require trust assumptions unless bridged through solutions like data availability committees or Axiom's approach to reading large historical datasets.
- **Developer Tooling and Abstraction:** User-friendly toolchains, debugging interfaces, and robust standard libraries are needed to reduce complexity for mainstream adoption.
- **Rollups vs. ZK-Coprocessors:** While L2 rollups offer scale via batch processing of transactions, they may not be sufficient for all advanced compute needs. Hybrid solutions that combine rollups and ZK-coprocessors are an evolving area.

7 Conclusion

Zero-knowledge coprocessors like RISC0, SP1, and Axiom bring new capabilities to decentralized applications, enabling off-chain computation with on-chain verifiability. From DeFi use cases such as dynamic interest rates and derivatives margining to advanced analytics and machine learning, these technologies preserve trustlessness while overcoming the computational limits of the EVM. Although alternative models like TEEs and MPC play valuable roles, zk-coprocessors provide a powerful balance of verifiability, privacy, and flexibility.

Continued research is needed to improve performance, expand developer tooling, and incorporate better data availability solutions. As these platforms mature, on-chain protocols will increasingly integrate high-performance off-chain compute without conceding core decentralization guarantees, unlocking a richer class of blockchain applications.

References

[1] RISC0 Project. *RISC0 zkVM Documentation.* Available at: `https://www.risczero.com/`.

[2] Axiom. *Axiom Documentation.* Available at: `https://www.axiom.xyz/`.

[3] Intel. *Software Guard Extensions (SGX).* `https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html`.

[4] Miller, A. *MPC as a Blockchain Confidentiality Layer.*