RISC Zero MCP: A tool to run verifiable Agentic Tasks

Ronan Takizawa

August 7, 2025

Abstract

The integration of zero-knowledge proofs with AI-driven workflows presents significant opportunities for privacy-preserving computation and verifiable AI operations. This paper introduces RISC Zero MCP, a Model Context Protocol (MCP) server implementation that enables seamless integration between AI applications and RISC Zero's zero-knowledge virtual machine (zkVM). Our system provides standardized interfaces for generating cryptographic proofs of computational integrity while maintaining the privacy of sensitive inputs. We present a comprehensive architecture that supports multiple mathematical operations including arithmetic, square root computation, modular exponentiation, and range proofs, as well as private machine learning algorithms including K-means clustering, linear regression, and neural network inference, all packaged as MCP tools accessible to AI agents. The machine learning capabilities enable AI agents to perform statistical analysis and pattern recognition while generating zero-knowledge proofs that verify computational correctness without revealing training data, model weights, or intermediate processing steps. The implementation demonstrates how MCP can serve as a bridge between complex cryptographic systems and AI applications, enabling new paradigms for trustless computation verification in both mathematical and machine learning contexts. We evaluate the system's performance, security considerations, and discuss implications for privacy-preserving AI workflows. Our work contributes to the growing ecosystem of AI-tool integration while addressing the critical need for verifiable computation in autonomous agent systems.

1 Introduction

The convergence of artificial intelligence and cryptographic systems has opened new frontiers in privacy-preserving computation and verifiable AI operations. As AI agents become increasingly autonomous and handle sensitive data, the need for cryptographic proof systems that can verify computational integrity without revealing private inputs has become paramount. Zero-knowledge proofs (ZKPs) offer a compelling solution by enabling the verification of computations while maintaining data privacy.

The introduction of the Model Context Protocol (MCP) by Anthropic has provided a standardized framework for AI applications to interact with external tools and data sources. MCP enables seamless communication between AI models and external systems through a unified protocol, breaking down integration barriers and facilitating interoperability across diverse platforms.

This paper presents RISC Zero MCP, a comprehensive system that bridges zero-knowledge proof generation and AI tool integration through the Model Context Protocol. We designed a complete Model Context Protocol (MCP) server that exposes RISC Zero zkVM capabilities as standardized AI tools, enabling AI agents to generate cryptographic proofs for various mathematical operations and private machine learning computations. The system includes specialized guest programs for K-means clustering, linear regression analysis, and neural network inference that execute within the zkVM to generate proofs of computational correctness while preserving the privacy of sensitive training data and model parameters.

2 Background and Motivation

2.1 Zero-Knowledge Proofs and RISC Zero

Zero-knowledge proofs allow one party (the prover) to prove to another party (the verifier) that a statement is true without revealing any information beyond the validity of the statement itself. RISC Zero's zkVM implements a STARK-based proof system that can generate proofs for arbitrary RISC-V programs, making it particularly suitable for complex computational tasks.

The RISC Zero architecture consists of three main components that work together to provide verifiable computation. Guest Programs are RISC-V executables that perform the computation to be verified, executing within a constrained environment that enables proof generation. The Host Environment provides the execution environment that runs guest programs and orchestrates the proof generation process, managing resources and coordinating between different system components. Finally, the Verification System encompasses all components responsible for validating the generated proofs, ensuring that computations were executed correctly without requiring re-execution.

2.2 Model Context Protocol

The Model Context Protocol (MCP) standardizes how AI applications interact with external tools and data sources through a client-server architecture that encompasses three key components working together to enable seamless integration. MCP Clients represent AI applications that consume external capabilities, serving as the consumer endpoints that request and utilize services from external systems, while MCP Servers act as services that expose tools, resources, and prompts to clients, functioning as the provider endpoints that implement and deliver specific capabilities. The Transport Layer provides the communication infrastructure enabling bidirectional data exchange between clients and servers, ensuring reliable and efficient message passing throughout the entire interaction lifecycle. MCP servers provide three distinct types of capabilities that address different aspects of AI-tool integration: Tools enable external operations and API invocations, allowing AI applications to perform actions and interact with external systems beyond their native capabilities; Resources expose both structured and unstructured data, providing AI applications with access to external information sources and datasets; and Prompts provide reusable templates for workflow optimization, enabling consistent and efficient interaction patterns across different use cases.

2.3 Motivation

As AI agents assume more responsibility for conducting tasks for organizations, they will start collaborating with other organizations' AI agents to complete tasks. This cross-party agentic workflow raises an issue: How can one AI agent verify claims made by another AI agent? For example, if an AI agent for a logistics company claimed it has calculated the most optimal shipping route, an AI agent for a manufacturing company may seek to verify this calculation. Similarly, if an AI agent claims to have performed machine learning analysis on sensitive customer data to make predictions or classifications, other agents need a way to verify the computational correctness without accessing the private training datasets or proprietary model architectures. In a single organization's agentic workflow, all tasks were completed internally, and an AI agent's tasks were verifiable through the logs of their function calls and responses. However, in a cross-party agentic workflow, organizations cannot simply send their logs, and even if they do, other organizations may not trust them. This problem is particularly acute for machine learning operations where training data often contains sensitive information that cannot be shared due to privacy regulations or competitive considerations. I believe this problem can be solved using ZK technology. By leveraging zero-knowledge proofs, AI agents can

verify deterministic computations to each other and omit sensitive data used in the computation if necessary, enabling verification of machine learning results without exposing underlying datasets or model parameters.

3 System Architecture

3.1 Overview

Our RISC Zero MCP system architecture, illustrated in Figure 1, consists of three main components that work together to provide comprehensive zero-knowledge proof capabilities. The AI Client represents applications like Claude Desktop and MCP Inspector that consume zero-knowledge proof functionality through standardized MCP protocols, providing tool discovery, request formatting, and result processing capabilities. The MCP Server acts as the orchestration layer, implemented in TypeScript, that manages proof generation workflows, validates schemas, handles processes, and provides robust error handling while serving as the bridge between AI clients and the underlying cryptographic system. The RISC Zero zkVM component encompasses the core cryptographic infrastructure, including Rust-based host binaries, specialized guest programs, and verification tools that execute computations, generate ZK-STARK proofs, and verify proof authenticity.

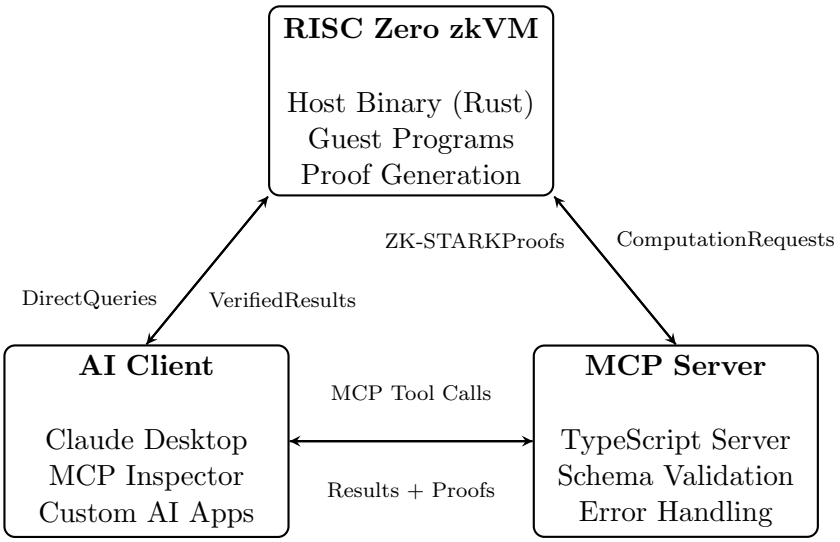

Figure 1: RISC Zero MCP System Architecture with Data Flow

3.2 AI Client

The AI Client component encompasses applications like Claude Desktop, MCP Inspector, and custom AI applications that consume zero-knowledge proof functionality through standardized MCP protocols. These clients provide tool discovery capabilities that automatically detect and register available zkVM operations, request formatting that validates input parameters and prepares them for transmission to the MCP server, and result processing that interprets proof artifacts and extracts computational results for presentation to end users. The clients support both interactive and programmatic access patterns, enabling AI agents to seamlessly integrate zero-knowledge proof generation into their workflows while maintaining familiar interaction paradigms. The standardized MCP interface ensures broad compatibility across different client implementations, allowing the same proof generation capabilities to be accessed from various AI applications and platforms.

3.3 MCP Server

The MCP Server acts as the orchestration layer, implemented as a TypeScript server that manages proof generation workflows and serves as the bridge between AI clients and the underlying cryptographic system. The server exposes ten primary tools to AI clients: mathematical operations including zkvm_add, zkvm_multiply, zkvm_sqrt, zkvm_modexp, and zkvm_range for range proofs; machine learning capabilities including zkvm_k_means, zkvm_linear_regression, zkvm_neural_network, and zkvm_logistic_regression for privacy-preserving statistical analysis; and verify_proof for independent verification of computational integrity. The server implements comprehensive schema validation to ensure input parameters meet cryptographic requirements, robust error handling with meaningful error messages for debugging, and asynchronous processing with appropriate timeout management to prevent resource exhaustion. Each tool follows the MCP specification with structured input schemas and standardized response formats containing computational results, proof metadata, and verification status.

3.4 RISC Zero Integration Layer

This layer provides the core integration with RISC Zero's zkVM system:

The RISC Zero Integration Layer provides the core integration with RISC Zero's zkVM system through four essential components that handle all low-level cryptographic operations. Guest Program Management maintains and executes specialized RISC-V programs for each supported operation, ensuring that computational logic is properly isolated and executed within the zkVM environment, while Proof Generation orchestrates the zkVM execution and STARK proof generation process, coordinating the complex sequence of operations required to produce cryptographic proofs. The Verification Interface provides standardized interfaces for proof verification, abstracting the complexity of cryptographic validation from higher-level components, and Performance Optimization implements caching and optimization strategies to improve proof generation performance, reducing computational overhead and improving system responsiveness.

3.5 RISC Zero zkVM

The RISC Zero zkVM component encompasses the core cryptographic infrastructure, including Rust-based host binaries, specialized guest programs, and proof generation capabilities that execute computations and generate ZK-STARK proofs. The host binary orchestrates zkVM execution and manages the proof generation workflow, configuring the executor environment with appropriate inputs and coordinating the complex sequence of cryptographic operations required to produce valid proofs. Guest programs are specialized RISC-V executables developed for each supported operation, implementing fixed-point arithmetic for decimal operations, optimized algorithms for machine learning computations, and comprehensive input validation to ensure mathematical correctness. The system includes dedicated guest programs for K-means clustering with distance calculations and centroid updates, linear regression with least squares optimization, neural network inference with pre-trained weights and sigmoid activation, and logistic regression for privacy-preserving binary classification. Proof generation produces cryptographic artifacts that verify computational correctness while maintaining privacy of sensitive inputs, and verification capabilities validate generated proofs against expected image identifiers to ensure authenticity and integrity.

4 Agent-to-Agent Verification Workflow

To demonstrate the practical application of our system in cross-party agentic workflows, we present a concrete example of AI agents performing logistic regression with zero-knowledge

proof verification. Figure 2 illustrates the interaction between a Prover Agent and a Verifier Agent, showcasing how one AI agent can verify machine learning computations performed by another agent without accessing sensitive training data or model parameters.

Figure 2: Agent-to-Agent Zero-Knowledge Proof Verification Workflow for Logistic Regression

The workflow demonstrates four key stages of cross-party agent verification:

Stage 1: Proof Generation - The Prover Agent executes the logistic regression computation using our system's zkvm_logistic_regression tool, which generates a ZK-STARK proof of the computation while keeping the training data and model parameters completely private. This corresponds to running 'npm run test:logistic-regression' in our implementation.

Stage 2: Proof Transmission - The Prover Agent transmits both the computational result (probability value 0.6967) and the accompanying ZK-STARK proof to the Verifier Agent. The proof serves as cryptographic evidence that the computation was performed correctly without revealing any sensitive information about the underlying data or model architecture.

Stage 3: Verification - The Verifier Agent uses the verify_proof tool to validate the received proof, confirming that the claimed logistic regression computation was indeed performed correctly and that the reported probability value is authentic. This verification process is computationally efficient (typically 15-25ms) and requires no access to the original training data or configuration.

This workflow exemplifies how our RISC Zero MCP system enables trustless computation verification in multi-party AI scenarios, addressing the critical challenge of verifiable computation in autonomous agent systems while maintaining privacy and competitive advantages.

5 Evaluation

5.1 Performance Analysis

We conducted comprehensive performance evaluation across multiple dimensions:

Proof Generation Time: We conducted comprehensive performance evaluation across multiple dimensions, with Table 1 showing proof generation times for different operations where complex operations like modular exponentiation require more time due to increased computational complexity.

Table 1: Proof Generation Performance

Operation	Mean Time (ms)	Std Dev (ms)
Addition	3,247	145
Multiplication	3,398	178
Square Root	3,521	203
Modular Exponentiation	4,127	267
Range Proof	3,876	198

Storage Efficiency: Our binary proof format achieves significant storage savings compared to traditional approaches, with the binary format requiring approximately 210KB per proof while the hexadecimal format requires approximately 420KB per proof, resulting in a storage reduction of approximately 50

Verification Performance: Proof verification demonstrates consistently fast performance across all operation types, typically completing in 15-25ms regardless of the original computation complexity, indicating that verification overhead remains minimal even for complex cryptographic operations.

5.2 Integration Testing

We validated MCP integration through comprehensive testing:

We validated MCP integration through comprehensive testing across multiple dimensions to ensure broad compatibility and reliable operation. Client Compatibility testing successfully validated operation with multiple MCP clients including Claude Desktop and custom implementations, demonstrating the system's adherence to MCP specifications and broad interoperability, while Tool Discovery verification confirmed automatic tool discovery and schema validation across different client implementations, ensuring seamless integration regardless of client-specific variations. Error Handling testing confirmed robust error propagation and handling across the MCP interface, validating that error conditions are properly communicated and handled throughout the entire system stack.

6 Conclusion

This paper presents RISC Zero MCP, a comprehensive system that bridges zero-knowledge proof generation and AI tool integration through the Model Context Protocol. Our implementation demonstrates that complex cryptographic systems can be made accessible to AI applications through standardized interfaces, enabling new paradigms for privacy-preserving and verifiable computation. Our evaluation shows that the system provides great performance characteristics while maintaining cryptographic security guarantees. As AI systems become more autonomous and handle increasingly sensitive data, the integration of cryptographic proof systems becomes essential. Our work provides a foundation for this integration, demonstrating practical approaches and identifying key considerations for future development.