Assessing RISC Zero using ZKit: An Extensible Test & Benchmarking Suite for ZKP Frameworks

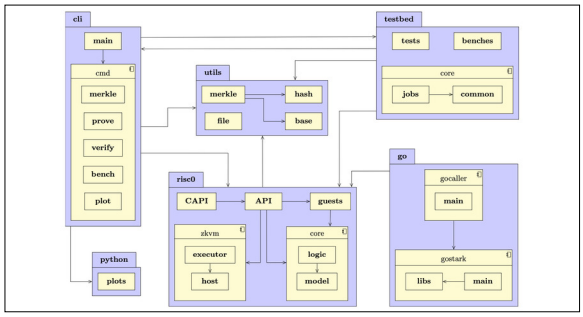
Graduate

Roman Bögli

Introduction: In the realm of digital money, a common use case involves proving the membership of a coin or token within a publicly accessible set of valid coins. The authority responsible for issuing or minting these valid coins establishes this set, for example, using a Merkle tree data structure. To demonstrate the inclusion of a specific coin in this set, a prover P provides an authentication path or Inclusion Proof (IP), which is a list of hash values or digests from the tree's leaf to its root. The verification process involves verifier V hashing the leaf data, appending the next digest to the result, and repeating this step until processing all elements in the authentication path. V is convinced of the coin's inclusion in the set when the final digest matches the publicly known root hash of the Merkle tree. While the above described method is considered an easy-to-implement and efficient way to demonstrate an item's inclusion in a set, it necessitates revealing the particular item in question. In the context of digital money, however, this contradicts with upholding user privacy. In other words, P should be able to convince V that a given coin belongs to a set of valid coins without disclosing the specific coin's identity. The emerging field of Zero-Knowledge Proof (ZKP) protocols serves as a solution to this problem. RISC Zero or risc0 enables verifiable general-purpose computations in zero-knowledge through its Virtual Machine (VM), which emulates a reduced Instruction Set Architecture (ISA). This so-called Reduced Instruction Set Computer (RISC) inspired the framework's naming. The reason for focusing on risc0 in this thesis is twofold. First, it allows to specify circuits or proof logic natively using Rust, which leverages the ease to develop custom ZKPs. Second, it implements the Scalable Transparent Argument of Knowledge (STARK) protocol which, in contrast to Succinct Non-Interactive Argument of Knowledge (SNARK) systems, does not require a trusted setup and is post-quantum secure.

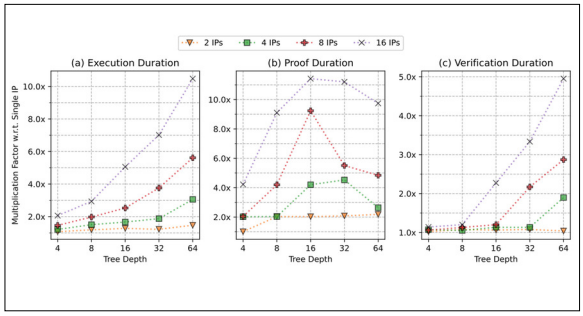
Approach / Technology: This thesis summarizes the most important properties of ZKPs and highlights the key differences between two famous implementation families, namely SNARK and STARK systems. Also, we provide a summary of promising software libraries or frameworks that help to create and verify ZKPs. Next, we analyze risc0 in detail using the above-addressed use case of an IP, i.e., proving a the inclusion of leaf in a Merkle tree data structure. Therefore, we propose the concept of ZKit, an extensible toolkit for testing and benchmarking various ZKP frameworks. Besides a Command Line Interface (CLI) to execute parameterized benchmarks or generate Merkle tree test data, it also contains functionality to define and exchange IPs in a unified way. Furthermore, ZKit exemplifies how ZKP circuits written in Rust can be ported to the Go ecosystem through a wrapper library. This portability facilitates the integration of ZKPs in existing Go projects such as for example the Fabric Token-SDK (FTS). Last but not least, we share our risc0 benchmark results on IPs in two different settings. In the first setting, we measured the performance and allocated resources to create one proof for a single IP. In the second setting, we aggregate or batch multiple IPs in a single proof. We compare and interpret the results of these two settings at the end and state our recommendations that we drew from it.

ZKit Component Diagram
Own presentment



Screenshot of the ZKit CLI
Own presentment

Comparative Duration of Batched IPs Relative to Single IP
Own presentment



Advisor
Dr. Alexandru Caracas

Co-Examiner
Dr. Thorsten Kramp

Subject Area
Computer Science, Data Science, Software and Systems

Project Partner
IBM Research, Rüschlikon, Zürich