Succinct Bootcamp: Introduction to Proof Systems and Arithmetization

Lecture Notes

September 16, 2024

1 Introduction

These notes are part of Succinct's internal "bootcamp" for new team members, covering zero-knowledge proofs (ZKPs) and our ZKVM, 0-SP1. Succinct aims to make ZKPs accessible to any developer through our open-source tools and decentralized prover network.

2 Proof Systems

2.1 What is a Proof?

An interactive proof is an algorithm allowing a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ of a claim's truth through dialogue.

Definition 1 (Claim)**.** A claim is represented by valid pairs (X, W), where:

- X is common input for both prover and verifier.
- W is the witness, known only to the prover.

The set of valid (X, W) pairs is the validity relation.

2.2 Examples of Validity Relations

1. **Fibonacci Relation:** (A, W) where $A = \text{Fibonacci}(n)$.
2. **Addition Relation:** $((a, b, c), W)$ valid if $a = b + c \mod 2^{32}$.
3. **Secret Key Relation:** (P, α) valid if α is the secret key for public key P.
4. **SP1 Relation:** $((P, I, O), W)$ valid if program P with input I produces output O.

2.3 Properties of Interactive Proofs

1. **Completeness:** Honest provers convince the verifier for valid claims.
2. **Soundness:** Probability of accepting invalid claims is negligible ($< 2^{-100}$).

A protocol is *public coin* if verifier messages are primarily random values.

2.4 Merkle Trees: A Blockchain Example

Consider a list of account balances $(i, b_i)_{i=1}^n$. To prove a specific account balance without storing the entire list:

1. Compute hash $h(b_i)$ for each balance b_i. 2. Combine hashes pairwise: $h(h(b_i), h(b_{i+1}))$. 3. Continue until reaching a single root hash.

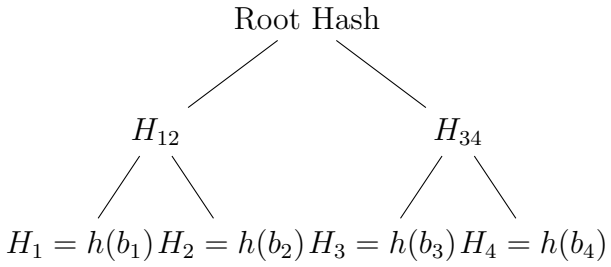

Figure 1: Simple Merkle Tree with 4 Leaf Nodes

To prove b_2's value, the prover provides:

- The value of b_2
- H_1 (sibling hash)
- H_{34} (other branch hash)

The verifier, storing only the root hash, can verify by: 1. Computing $H_2 = h(b_2)$ 2. Computing $H_{12} = h(H_1||H_2)$ 3. Computing $\text{Root} = h(H_{12}||H_{34})$ 4. Checking if the computed root matches the stored root hash

3 Arithmetization

Arithmetization encodes computations as polynomial equations over finite fields.

3.1 Finite Fields

We work over $\mathbb{F}_p$, where p is prime, consisting of $\{0, 1, ..., p-1\}$ with modular arithmetic.

Proposition 1. *For non-zero $a \in \mathbb{F}_p$, there exists unique $c \in \mathbb{F}_p$ with $a \cdot c \equiv 1 \pmod{p}$.*

3.2 Types of Arithmetization

1. **Arithmetic Circuits:** Computational graphs with addition and multiplication.
2. **Algebraic Intermediate Representation (AIR):**

- Table T of size $n \times m$

- Public outputs
- Constraints: First row, last row, and transition constraints

3.3 Fibonacci AIR Example

- First row: $a = 0$, $b = 1$
- Transition: $a' = b$, $b' = a + b$
- Last row: $a = c$ (public output)

3.4 Polynomial Encoding

For $n = 2^d$, $g \in \mathbb{F}_p$ with $g^{2^d} = 1$, there exists a unique polynomial $\tilde{T}_j(x)$ of degree $2^d - 1$ for each column j, where $\tilde{T}_j(g^i) = T_{ij}$ for all $0 \leq i < n$.

The AIR claim reduces to proving:

$$\tilde{P}(x) = Q(x)(x^{2^d} - 1) \tag{1}$$

Verified probabilistically at a random point $r \in \mathbb{F}_p$:

$$\tilde{P}(r) = Q(r)(r^{2^d} - 1) \tag{2}$$

4 Polynomial Commitment Schemes

Polynomial commitment schemes allow provers to commit to polynomials and prove evaluations. FRI (Fast Reed-Solomon Interactive Oracle Proof of Proximity) is one such scheme used in SP1.

5 Conclusion: Key Takeaways

As we conclude this introduction to proof systems and arithmetization, let's review the key concepts and their importance in the context of zero-knowledge proofs and Succinct's SP1 zkVM:

1. **Interactive Proofs:** These systems allow a prover to convince a verifier of a claim's truth through dialogue, forming the foundation of zero-knowledge protocols.
2. **Validity Relations:** Understanding various types of relations (e.g., Fibonacci, Addition, Secret Key) helps in grasping the wide applicability of ZKPs in different domains.
3. **Merkle Trees:** This data structure is crucial for efficient verification of membership in large datasets, particularly important in blockchain applications.

4. **Arithmetization:** The process of encoding computations as polynomial equations over finite fields is a key step in creating efficient ZKPs.

5. **AIR (Algebraic Intermediate Representation):** This representation allows complex computations to be expressed in a form suitable for ZKP systems.

6. **Polynomial Encoding:** Transforming computational statements into polynomial equations enables the use of algebraic techniques in proof generation and verification.

7. **Polynomial Commitment Schemes:** These schemes, like FRI, are essential for efficiently proving properties of polynomials without revealing the entire polynomial.

These concepts form the theoretical backbone of modern ZKP systems, including Succinct's SP1 zkVM. By leveraging these principles, SP1 allows developers to write ZKPs in Rust, making the technology more accessible and practical for a wide range of applications.

As we move forward in this bootcamp, we'll explore how these foundational ideas are implemented in SP1, enabling efficient proof generation, verification, and the development of complex zero-knowledge applications. The goal is to empower developers to harness the power of ZKPs in their projects, contributing to a new era of privacy-preserving and scalable blockchain solutions.