The ZK Zeitgeist: A Mathematical Primer for University Students

09/02/2025

1 A Note to Readers

Gm, privacy geeks. This is a zeitgeist you should check out that will keep you sane during the workshop. I took a great deal of inspiration from the RareSkills ZKP book written by @Jeyffre, the Privacy and Scaling Explorations' YouTube channel & of course ChatGPT. Hope you read this before attending the IRL workshop.

NOTE: I have zero knowledge about zero knowledge proofs, and even if I do, I have no way to prove it.

2 Introduction

Zero-Knowledge Proofs (ZKPs) allow one party (the prover) to convince another party (the verifier) that a statement is true without revealing any additional information. This primer will introduce the key mathematical foundations required to understand ZKPs.

The *P vs NP* problem is a fundamental question in computer science that explores the relationship between two classes of problems: **P** and **NP**. Understanding this problem not only deepens our grasp of computational complexity but also sheds light on advanced concepts like zero-knowledge proofs (ZKPs).

3 Defining P and NP

In computational complexity theory, problems are categorized based on the resources required to solve them.

- **P (Polynomial Time):** This class consists of problems that can be solved in polynomial time. An example is sorting a list of numbers; algorithms like mergesort accomplish this in $O(n \log n)$ time, where n is the number of elements in the list.

- **NP (Nondeterministic Polynomial Time):** This class includes problems where a given solution can be verified efficiently, even if finding that solution is computationally hard. A classic example is the Boolean satisfiability problem (SAT), where checking whether a set of variable assignments satisfies a Boolean formula can be done in polynomial time.

The central question of the **P vs NP** problem is: *If a solution to a problem can be verified quickly (in polynomial time), can the solution also be found quickly?* In other words, does $P = NP$? Most researchers believe that $P \neq NP$, implying that some problems have solutions that can be verified efficiently but not found efficiently.

4 NP-Completeness

Within NP, there exists a subset known as **NP-complete** problems. These are the hardest problems in NP, in the sense that if any NP-complete problem can be solved in polynomial time, then every problem in NP can also be solved in polynomial time. An example of an NP-complete problem is the *traveling salesman problem* (TSP), which seeks to determine the shortest possible route that visits a set of cities and returns to the origin.

5 Zero-Knowledge Proofs and Their Connection to P vs NP

Zero-knowledge proofs (ZKPs) are cryptographic protocols that allow one party (the prover) to convince another party (the verifier) that a statement is true without revealing any additional information. These proofs are particularly significant in cryptography, privacy-preserving applications, and blockchain technologies.

The feasibility of ZKPs is closely tied to the P vs NP problem. For a problem to have a zero-knowledge proof, it must be in NP. This is because ZKPs rely on efficient verification: the verifier must be able to confirm the solution's validity in polynomial time without actually learning the solution itself. This property is essential in cryptographic protocols like zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge), which allow for scalable and private verification of computations.

6 Mathematical Foundations

6.1 Modular Arithmetic

- **Definition**: Modular arithmetic deals with numbers wrapping around after reaching a certain value (the modulus).

- **Example**: $a \equiv b \mod m$ means that $a - b$ is divisible by m.
- **Key Properties**:
  - Addition: $(a + b) \mod m = [(a \mod m) + (b \mod m)] \mod m$
  - Multiplication: $(a \times b) \mod m = [(a \mod m) \times (b \mod m)] \mod m$
  - Exponentiation: $(a^b) \mod m$ (important in cryptography)

6.2 Finite Fields

- A **finite field** (or Galois field, $GF(p)$) consists of a finite set of elements where addition, subtraction, multiplication, and division (except by zero) are defined.
- **Example**: $GF(7) = 0, 1, 2, 3, 4, 5, 6,$ where calculations are performed modulo 7.
- Finite fields are fundamental in constructing SNARKs and STARKs.

6.3 Elliptic Curves

- **Definition**: An elliptic curve is defined by an equation of the form:

  over a finite field.
- **Key Operations**:
  - **Point Addition**: Given two points P and Q, their sum $P + Q$ is computed using geometric rules.
  - **Scalar Multiplication**: Computing kP, which is crucial for cryptographic security.
- Used in SNARKs due to their efficient bilinear pairings.

6.4 Discrete Logarithm Problem (DLP)

- Given $g^x \mod p = h$, finding x is computationally hard.
- The security of many cryptographic schemes, including Schnorr proofs and SNARKs, relies on the difficulty of DLP.

7 Cryptographic Protocols & Proof Techniques

7.1 Arithmetization

- Converts a computation into a system of equations (constraints) over a finite field.
- **Rank-1 Constraint Systems (R1CS)**:

  - Used in SNARKs like Groth16 to express a computation as quadratic constraints.
  - Example: Encoding $w = x \cdot y$ as $A \cdot B = C$.

7.2 Polynomial Commitment Schemes (PCS)

- Used to efficiently prove that a polynomial has certain properties without revealing the polynomial.
- **KZG Commitments**:
  - Use elliptic curve pairings to commit to a polynomial and later prove evaluations at specific points.

7.3 Fiat-Shamir Heuristic

- Transforms an **interactive proof** into a **non-interactive proof** using a cryptographic hash function.
- Example: Used in SNARKs to remove the need for multiple rounds of interaction.

7.4 Sumcheck Protocol

- A key tool in scalable verification (GKR, STARKs).
- Used to prove that a polynomial sum evaluates to a certain value efficiently.

8 ZKP Constructions

8.1 SNARKs (Succinct Non-Interactive Argument of Knowledge)

- **Groth16**:
  - Uses R1CS and bilinear pairings for efficient proofs.
  - Very short proofs (constant-size) but requires a trusted setup.
- **PLONK**:
  - General-purpose SNARK with universal trusted setup.
  - Uses KZG commitments and permutation arguments.

8.2 STARKs (Scalable Transparent Argument of Knowledge)

- Post-quantum secure (relies on hash functions rather than pairings).
- Uses **FRI (Fast Reed-Solomon Interactive Oracle Proofs)** for efficient verification.
- No trusted setup required.

9 Practical Applications

9.1 Privacy-Preserving Applications

- **Tornado Cash**: Enables private Ethereum transactions using zkSNARKs.
- **zkEmail**: Proves email attributes (e.g., domain verification) without revealing content.

10 Future Directions

10.1 Post-Quantum ZKPs

- Exploring hash-based and lattice-based ZKP constructions for quantum resistance.

10.2 Recursive ZK Proofs & zkML

- **Recursive SNARKs**: Proofs verifying other proofs, enabling rollups and scalability.
- **zkML**: Zero-knowledge verification of machine learning models.