UNIVERSIDAD DE ANTIOQUIA

Zero Knowledge Proof

University of Antioquia, Medellín

Electiva de Blockchain

Énfasis en Construcción de Desarrollo Seguro

ADRIEN ELOI MORIN

Índice

Introducción

The concept of Zero-Knowledge Proofs (ZKPs) represents a fascinating and powerful cryptographic tool that has gained increasing attention in recent years. In a world where privacy and security are paramount, particularly in digital communication and blockchain technologies, Zero-Knowledge Proofs provide a unique solution to a central problem: how can one party prove knowledge of a specific piece of information to another party without revealing the information itself?

A Zero-Knowledge Proof allows one party, known as the "prover", to convince another party, known as the "verifier", that they possess certain knowledge or have completed a specific task, without sharing any details beyond the fact that the knowledge or task completion is genuine. This principle is counterintuitive at first, but offers immense potential for privacy in digital systems and blockchain.

Origins and Importance of Zero-Knowledge Proofs

Zero-Knowledge Proofs were first introduced in the 1980s by researchers Shafi Goldwasser, Silvio Micali, and Charles Rackoff. In their foundational paper "The Knowledge Complexity of Interactive Proof-Systems" (1985), they formalized the concept and laid the groundwork for modern applications of this cryptographic primitive. Their work was revolutionary, as it addressed a major issue in cryptographic communications: the need to prove or verify a statement without revealing sensitive data.

The invention of ZKPs arose from the broader context of public-key cryptography, a field that deals with securely transmitting information over insecure channels. While encryption allows two parties to exchange data in a confidential manner, proving knowledge of a fact or attribute without exposing it presents a different challenge. Goldwasser, Micali, and Rackoff’s work opened new avenues for addressing this challenge.

Defining Zero-Knowledge Proofs

A Zero-Knowledge Proof must satisfy three essential properties:

1. **Completeness**: If the statement is true, an honest prover can convince an honest verifier that they possess the necessary knowledge.
2. **Soundness**: If the statement is false, no dishonest prover can convince the verifier that the statement is true, except with some negligible probability.
3. **Zero-Knowledge**: If the statement is true, the verifier learns nothing beyond the fact that the statement is true. In other words, the verifier does not gain any additional information about the statement or the knowledge the prover has.

These three properties ensure that ZKPs are both secure and private, making them ideal for applications where confidentiality is critical. A ZKP can be interactive or non-interactive, depending on whether multiple rounds of communication between the prover and verifier are required.

A Simple Example of Zero-Knowledge Proof

To illustrate the concept of Zero-Knowledge Proofs, consider a simple analogy known as the "Ali Baba cave". Imagine a cave shaped like a ring, with an entrance and a magic door deep inside that only the prover knows how to open. The prover wishes to convince the verifier that they know how to open the door, but without revealing the secret.

The verifier stands at the entrance of the cave, and the prover walks down one of the two paths in the cave, either the left path or the right path, until they reach the door. The verifier, who cannot see which path the prover took, asks the prover to reappear from a specific path. If the prover truly knows how to open the magic door, they can switch between the two paths using the door and exit through the one requested by the verifier.

By repeating this process multiple times, the verifier can be confident that the prover knows how to open the door (because otherwise, the prover would fail at least some of the time). However, the verifier learns nothing about *how* the prover opens the door. This is a Zero-Knowledge Proof: the prover convinces the verifier of a fact without revealing any knowledge about how they accomplish it.

Types of Zero-Knowledge Proofs

Zero-Knowledge Proofs come in several varieties, each suited for different scenarios:

1. **Interactive Zero-Knowledge Proofs**: In this model, the prover and verifier engage in a back-and-forth communication process. The prover provides responses based on challenges posed by the verifier. Interactive proofs are often used in protocols where real-time interaction is possible.
2. **Non-Interactive Zero-Knowledge Proofs (NIZK)**: In this version, there is no need for interaction between the prover and verifier. The prover generates a proof that can be verified by the verifier at any time, without further communication. Non-interactive proofs are more practical for distributed systems like blockchain, where interaction between users and systems may be limited or impossible.
3. **Succinct Non-Interactive Arguments of Knowledge (SNARKs)**: SNARKs are a type of NIZK proof that is compact and can be verified quickly. This makes them ideal for applications like blockchain, where efficiency is critical. SNARKs are currently used in projects like Zcash, a cryptocurrency that provides enhanced privacy through Zero-Knowledge Proofs.
4. **Zero-Knowledge Succinct Transparent Arguments of Knowledge (ZK-STARKs)**: ZK-STARKs are another variant of Zero-Knowledge Proofs that emphasize transparency and scalability. They avoid some of the cryptographic assumptions required by SNARKs, making them more secure and less reliant on trusted setups.

Algorithms

Interactive Zero-Knowledge Proofs

The general algorithm

Here's a high-level schematic algorithm for an Interactive Zero-Knowledge Proof:

1. Setup Phase:
   - The prover and verifier agree on a common reference string or parameters.
2. Commitment Phase:
   - The prover commits to a secret value or a solution to a problem.
   - This commitment is sent to the verifier.
3. Challenge Phase:
   - The verifier sends a random challenge to the prover.
   - This challenge is typically a random value or a question related to the secret.
4. Response Phase:
   - The prover computes a response based on the challenge and the secret.
   - This response is sent back to the verifier.
5. Verification Phase:
   - The verifier checks the response against the challenge and the initial commitment.
   - If the response is consistent with the challenge and the commitment, the verifier is convinced that the prover knows the secret.

Example: The Magic Door

Imagine there is a magical cave shaped like a ring, with two identical doors at the entrance. The cave is perfectly circular, and the two doors are indistinguishable from the outside. The prover (Peggy) claims to know a secret word that opens either door. The verifier (Victor) wants to be convinced that Peggy knows the secret word, but he doesn't want to learn the word itself.

Here's how the Interactive Zero-Knowledge Proof works in this scenario:

1. Setup Phase:
   - Peggy and Victor agree on the structure of the cave and the two doors.
2. Commitment Phase:
   - Peggy enters the cave through one of the doors (let's say Door A) and walks to the center of the cave.
   - Peggy does not reveal which door she used to enter.
3. Challenge Phase:
   - Victor, standing outside, randomly chooses a door (either Door A or Door B) and shouts it to Peggy.
4. Response Phase:
   - Peggy exits the cave through the door that Victor chose.
   - If Victor chose Door A, Peggy exits through Door A.
   - If Victor chose Door B, Peggy exits through Door B.
5. Verification Phase:
   - Victor checks that Peggy exits through the door he chose.
   - If Peggy exits through the correct door, Victor is convinced that Peggy knows the secret word to open the doors.

Repeated Rounds

To reduce the probability of Peggy cheating (e.g., guessing the door), this process can be repeated multiple times. Each round, Victor randomly chooses a door, and Peggy must exit through the chosen door. After several rounds, Victor becomes increasingly confident that Peggy knows the secret word.

Non-Interactive Zero-Knowledge Proofs (NIZK)

The general algorithm

1. **Setup**:
    - A *trusted setup* phase generates public parameters, often called a **common reference string (CRS)**.
    - This CRS is shared between the prover and verifier. It is essential for both parties to trust these parameters (i.e., they are generated securely).
2. **Statement & Witness**:
    - The *statement* is the claim the prover wants to prove. Example: *"I know a solution to the equation y = g^x (mod p)."*
    - The *witness* is secret information that proves the statement. Example: The secret value of **x** in the above equation.
3. **Prover's Computation**:
    - Using the CRS, the prover generates a **proof** that the statement is true based on the witness.
    - The result is a **single message** containing this proof (no need for multiple exchanges with the verifier).
4. **Verification**:
    - The verifier uses the CRS and the prover’s message (proof) to check if the statement is valid.
    - If the proof passes the verification process, the verifier is convinced the statement is true—without learning anything about the secret witness.

Algorithm Outline

1. **Setup Phase**:
    - Input: Security parameter k
    - Output: Common Reference String CRS
2. **Prover’s Proof Generation**:
    - Input: CRS, statement S, witness W
    - Output: Proof P

3. **Verifier's Verification**:
   - Input: CRS, statement S, proof P
   - Output: Accept/Reject

Succinct Non-Interactive Arguments of Knowledge (SNARKs)

The general algorithm

1. **Setup**:
   - A trusted party generates a **Common Reference String (CRS)**, including public and secret parameters.
   - The public part is shared with everyone and used for proving and verifying.
2. **Statement & Witness**:
   - The prover has a statement to prove. Example: *"This computation is correct."*
   - The witness is the secret data needed to prove the statement. Example: A hidden input to the computation.
3. **Prover's Proof Generation**:
   - The prover uses the CRS, the statement, and the witness to generate a **succinct proof** (small in size).
   - This proof can be easily shared as a single message with the verifier.
4. **Verification**:
   - The verifier uses the CRS and the proof to quickly check the validity of the statement.
   - If the proof is valid, the verifier accepts the statement as true—without knowing any extra information.

Algorithm Outline

1. **Setup Phase**:
   - Input: Security parameter k
   - Output: Common Reference String CRS

2. **Prover's Proof Generation**:
    - Input: CRS, statement S, witness W
    - Output: Proof P
3. **Verifier's Verification**:
    - Input: CRS, statement S, proof P
    - Output: Accept/Reject

Zero-Knowledge Succinct Transparent Arguments of Knowledge (ZK-STARKs)

The general algorithm

1. **Setup**:
    - No trusted setup is required! Instead, ZK-STARKs rely on **public randomness** to generate the parameters.
    - This ensures transparency—everyone can verify that the parameters were generated correctly.
2. **Statement & Witness**:
    - The prover wants to prove a statement. Example: *"This large computation is correct."*
    - The witness is the secret data or input to this computation.
3. **Prover's Proof Generation**:
    - The prover uses **polynomial-based algorithms** (like the Fast Fourier Transform) to generate a **succinct proof** that the statement is true.
    - The proof size scales logarithmically with the size of the computation, meaning it can remain small even for large data sets.
4. **Verification**:
    - The verifier checks the proof using a **lightweight verification algorithm**, confirming the statement is true.
    - The proof reveals nothing about the witness and can be verified in milliseconds.

Algorithm Outline

1. **Setup Phase**:
   - Input: Security parameter k, public randomness source.
   - Output: Transparent public parameters.
2. **Prover’s Proof Generation**:
   - Input: Public parameters, statement S, witness W.
   - Output: Proof P.
3. **Verifier’s Verification**:
   - Input: Public parameters, statement S, proof P.
   - Output: Accept/Reject.

UNIVERSIDAD DE ANTIOQUIA

Reference bibliography

1. Goldwasser, S., Micali, S., & Rackoff, C. (1985). The Knowledge Complexity of Interactive Proof-Systems. *SIAM Journal on Computing*, 18(1), 186-208.
2. Circularise. (2022). Zero-Knowledge Proofs Explained in 3 Examples. Retrieved from Circularise blog.
3. Partisia. (n.d.). The Power of Zero-Knowledge Proofs: Prove Anything Without Revealing Everything.
4. Wikipedia. (n.d.). Zero-Knowledge Proof. Retrieved from Wikipedia.
5. Aleo. (n.d.). What is a Zero-Knowledge Proof?
6. Cryptology ePrint Archive. (2024). Do You Need a Zero Knowledge Proof? Technical University of Munich, Imperial College London, University College London.