Tutorial 6:

Zero-Knowledge (ZK) Proofs

CSCD71: Blockchains & Decentralized Applications

Nikhil Lakhwani, Nov 24 2023.

Acknowledgements

Chainlink Research Labs.

Bibliography:

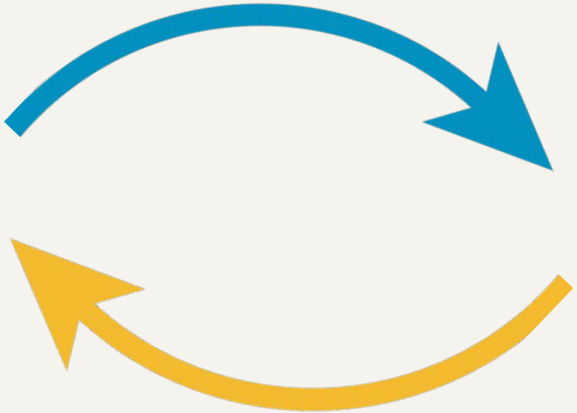
[1] A. Juels, “What is a Zero-Knowledge Proof (ZKP)?”, Chainlink Research Labs, 2022. [Online].

[2] A. Juels, “Interactive Zero-Knowledge Proofs”, Chainlink Research Labs, 2022. [Online]

[3] S. Goldwasser, S. Micali and C. Rackoff, “The knowledge complexity of interactive proof-systems”, in Proceedings of the seventeenth annual ACM symposium on Theory of computing, Providence, Rhode Island, USA, 1985, pp. 291-304. doi: 3.

Coke-or-Pepsi ZK-Proof Protocol

ZK-Proofs



Prover

Verifier

Coke-or-Pepsi ZK-Proof Protocol

Our protocol allows a prover to show that it knows how to differentiate between Coke and Pepsi without revealing how to the Verifier.

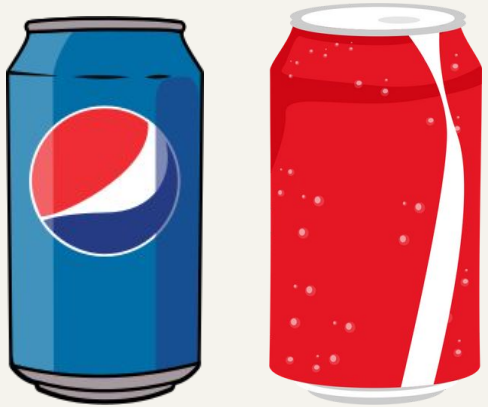
Coke-or-Pepsi ZK-Proof Protocol

(1) The verifier prepares a glass of Coke and Pepsi and randomly shuffles them.

Coke-or-Pepsi ZK-Proof Protocol

(1) The verifier prepares a glass of Coke and Pepsi and randomly shuffles them.

(2) With the verifier looking away, the prover observes/tastes both glasses and decides which one is Coke.



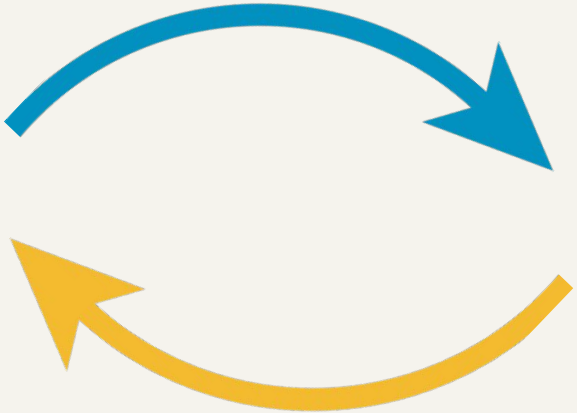
Coke-or-Pepsi ZK-Proof Protocol

(1) The verifier prepares a glass of Coke and Pepsi and randomly shuffles them.

(2) With the verifier looking away, the prover observes/tastes both glasses and decides which one is Coke.

(3) Repeat the above process 40 times, and the verifier accepts that the prover indeed knows how to differentiate between Coke and Pepsi if the prover is correct every time.

Interactive ZK-Proofs



Prover

Verifier

What are **ZK-Proofs**?

Definition:

A **ZK-Proof** is a cryptographic method by which one party (the prover) can **prove** to another party (the verifier) that a given statement is true, **without conveying any secret information**, apart from the fact that the statement is indeed true (proof). [1]

What are **ZK-Proofs**?

Properties:

Completeness: If the statement is true, an honest verifier will be convinced by an honest prover.
Soundness: If the statement is false, no cheating prover can convince an honest verifier that it's true.
Zero-knowledge: If the statement is true, no verifier learns anything other than the fact that the statement is true. [2]

Types of ZK-Proofs

Interactive ZK-Proofs:

Interactive Zero-Knowledge Proofs (ZKPs) are protocols that require communication between the prover and the verifier. The prover can convince the verifier of a statement without revealing any information except that the statement is true.

Example: Coke-or-Pepsi game, where the prover can prove they know the difference between the two drinks by answering the verifier’s questions correctly.

Types of ZK-Proofs

Non-Interactive ZK-Proofs:

Non-interactive zero-knowledge (NIZKs) proofs have found many applications in the blockchain domain due to their attractive feature of transferability: The prover, without knowing the identity of the verifier, can generate one proof that can be used to convince anyone who receives it.

NIZKs with very small proofs are generally categorized as zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) protocol. [2]

Types of ZK-Proofs

Non-Interactive ZK-Proofs:

Non-Interactive Zero-Knowledge Proofs (NIZKs) are protocols that do not require communication between the prover and the verifier. The prover can generate a proof that convinces any verifier that they know a secret without revealing it.

Example: Digital signatures, The signer uses a private key to sign a public document and generates a signature; any verifier can check the signature with the document and a public key.

Use Cases

Authentication Systems:
Allowing users to prove their identity (e.g., logging into a system) without revealing their actual credentials (like a password).

Voting Systems:
Ensuring the integrity of votes while maintaining the privacy of voters' choices.

Data Sharing:
Securely proving the possession of certain data or qualifications without revealing the data itself (useful in confidential business processes).

Use Cases (for us!)

ZK Proofs are a powerful tool that we can use for scalability and privacy on the blockchain.

Privacy-Enhanced Transactions: Enabling transactions without revealing sensitive details (such as recipient/sender addresses or transaction amounts like in Zcash, achieving transaction privacy on a public blockchain).

Scalability Solutions: Used in Layer 2 solutions (like ZK-Rollups) to aggregate multiple transactions into a single proof, reducing the load on the main blockchain.

Use Cases (for us!)

Scalability Solutions: Used in Layer 2 solutions (like ZK-Rollups) to aggregate multiple transactions into a single proof, reducing the load on the main blockchain.

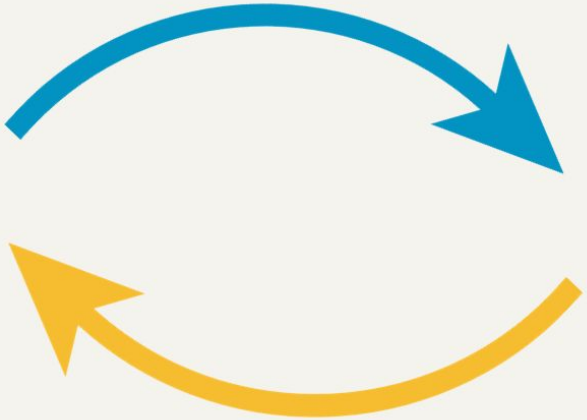
ZK-Rollups:
A ZK-rollup works by combining hundreds of off-chain transactions into a single transaction and sending a summary of the changes and a validity proof to the main chain.

The validity proof is a cryptographic assurance that the off-chain transactions are correct and follow the rules of the main chain.

The main chain only needs to verify the proof and update the state accordingly, without processing all the transaction details.

Non-interactive ZK-Proofs

Sends a proof (construct demonstrating truth of the statemnt)



Prover

takes the proof and knowledge of the protocol to validate the proof, without gaining knowledge of the secret information.

Verifier

Demo: ZK-Rollup Objectives

1. Set up a tau ceremony.
2. Write and compile our own circuit for a ZK-Rollup.
3. Provide inputs into our circuit and calculate a witness.
4. Export a verification key, used to verify the proof.
5. Create the proof using the witness, circuit & proving key.
6. Verify the proof locally.
7. Verify the proof on chain, by turning the verifier into a smart contract, and deploying it.
8. Verifying a proof on chain using our verifier.

ZK-Proof Glossary

Tau Ceremony
Definition: A multi-party computation protocol to generate a common reference string (CRS) or structured reference string (SRS) for zk-SNARKs.
Usage: Ensures the security of the zk-SNARK system by distributing trust among multiple participants.

Circuit
Definition: A computational blueprint defining the logic and constraints of the computation for which you are proving knowledge.
Usage: Written and compiled to create the necessary files for generating proofs (e.g., r1cs, wasm).

Witness Calculation
Definition: The process of assigning values to the inputs of a circuit so that all its constraints are satisfied.
Usage: Generates the witness, which is used along with the proving key to create the zk-proof.

Verification Key
Definition: A key generated during the trusted setup, used for verifying zk-SNARK proofs.
Usage: Exported for use in the proof verification process.

Proof *Generation*
Definition: The creation of a cryptographic proof demonstrating that the witness satisfies the circuit's constraints.
Usage: The proof, along with public inputs, is what the prover submits for verification.

ZK-Proof: Pepsi & Coke Edition

Tau Ceremony
Imagine a scenario where Pepsi and Coke collaborate on a joint campaign but want to keep certain formula details secret. They participate in a tau ceremony to establish a common foundation for a zk-SNARK system, where multiple parties contribute to the setup's security. Here, neither company can independently compromise the system.

Circuit
Suppose Pepsi wants to prove that their drink has fewer calories than Coke without revealing the exact number. They would design a circuit that encodes the logic "Pepsi's calories < Coke's calories" and compile it to create the necessary files for proof generation.

Witness Calculation
Pepsi would then assign specific values (e.g., calorie counts) to their circuit inputs, ensuring these satisfy the encoded constraints. This calculation results in a witness, confirming Pepsi's claim without revealing the actual calorie count.

Verification Key
During the trusted setup, a verification key is generated. If Coke wants to verify Pepsi's claim, they would use this key to check the validity of Pepsi's proof, ensuring it aligns with the agreed-upon circuit constraints.

Proof Generation
Pepsi-Coke Context: Pepsi generates a zk-SNARK proof using their witness and the proving key. This proof is what Pepsi would provide to Coke. It cryptographically demonstrates that Pepsi's drink has fewer calories, according to the circuit's logic, without revealing the exact calorie count.

Build our own Circuit

Version 1: Simple Arithmetic Circuit
Version 2: Verify EdDSA Algorithm
Version 3: Get Merkle Root, Check Existence for a Leaf
Version 4: Process Transactions -> **ZK Rollup!**

Demo: ZK-Rollup Objectives

1. Set up a tau ceremony.
2. Write and compile our own circuit for a ZK-Rollup.
3. Provide inputs into our circuit and calculate a witness.
4. Export a verification key, used to verify the proof.
5. Create the proof using the witness, circuit & proving key.
6. Verify the proof locally.
7. Verify the proof on chain, by turning the verifier into a smart contract, and deploying it.
8. Verifying a proof on chain using our verifier.

Thanks!

Do you have any questions?