Comparative Analysis of Zero-Knowledge Proofs in Privacy-Preserving Blockchain Notarization

Dilpriya T K - 21

Guided by Dr. Ajish Kumar K S
Department of Computer Science
Government College of Engineering, Kannur

October 2024

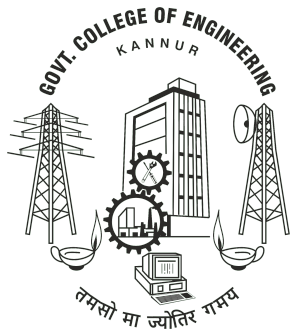

Table of Contents

Introduction

- Blockchain notarization uses decentralized ledgers to verify the authenticity and integrity of documents or transactions.
- It eliminates the need for intermediaries (e.g., traditional notaries) by ensuring transparency and immutability.
- Privacy is a key concern as public blockchains expose transaction data. Zero-knowledge proofs (ZKPs) enable verifications without revealing private information.
- This seminar will explore various types of ZKPs (e.g., zk-SNARKs, zk-STARKs, Bulletproofs) and their role in privacy-preserving blockchain notarization.
- The discussion will cover the strengths, limitations, and real-world applications of each ZKP mechanism.

Basic Terminologies

Zero-Knowledge Proof (ZKP)

A cryptographic method by which one party (prover) can prove to another (verifier) that they know a piece of information without revealing the information itself.

zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge)

A type of ZKP that allows proof of knowledge without interaction between prover and verifier after a setup phase.

zk-STARKs (Zero-Knowledge Scalable Transparent Arguments of Knowledge)

A ZKP system that provides scalability and security without requiring a trusted setup.

Basic Terminologies

Bulletproofs

A ZKP that focuses on reducing proof size and improving verification time without needing a trusted setup.

Trusted Setup

An initial phase in some ZKP systems (e.g., zk-SNARKs) where cryptographic parameters are generated in a way that must remain secure.

Cryptography

The science of encoding and decoding information to protect it from unauthorized access.

Papers Considered

- Zk-SNARKs As A Cryptographic Solution For Data Privacy And Security In The Digital Era.
- Ligerolight: Optimized IOP-Based Zero-Knowledge Argument for Blockchain Scalability.
- Bulletproofs: Short Proofs for Confidential Transactions and More.

Zk-SNARKs As A Cryptographic Solution For Data Privacy And Security In The Digital Era

OBJECTIVES:

- Verify the validity of a legal document without revealing its full content.
- Allow parties to prove the document contains valid information, such as signatures or compliance with legal standards, using zk-SNARKs.
- Enable a third party (e.g., a verifier) to confirm the authenticity of the document without requiring access to the actual document. Analyze multi-channel sEMG signals for nine common hand movements.

Solution Strategy

1: Define the Proof Statement

Establish the specific information you want to prove without revealing the entire document.

2: Create the zk-SNARK Circuit

Design a mathematical circuit that translates the proof statement into verifiable cryptographic terms.

Use libraries like ZoKrates or snarkjs to define the computation that zk-SNARK will prove.

3: Key Generation (Trusted Setup)

Generate the public key and private key for the zk-SNARK system.

- The public key is used by the verifier (e.g., a court or government agency) to verify the proof.
- The private key is used by the prover (e.g., the legal party) to generate the zk-SNARK proof.

Solution Strategy

- Trusted Setup: This step generates parameters for the zk-SNARK system. Ensure that this setup is secure, as compromising this step can jeopardize the entire system.

4: Generate the zk-SNARK Proof

The prover (the legal party) uses their private key to generate the zk-SNARK proof.

Process:

- The legal document is hashed.
- The prover generates a zk-SNARK proof using the private key and the zk-SNARK circuit created earlier. This proof certifies that the document meets specific criteria (e.g., it is signed by an authorized party) without revealing the document itself.

Solution Strategy

5: Verification Process

The verifier (e.g., a third-party, such as a government or legal institution) uses the public key and the zk-SNARK proof to verify the validity of the legal document.

- Process:
- The verifier inputs the public key and the zk-SNARK proof into the verification algorithm.
- If the zk-SNARK proof is valid, the verifier can be certain that the document meets the required legal conditions (e.g., valid signatures, proper formatting) without needing to see the document's content.

Solution Strategy

6: Deploying on Blockchain

Store the zk-SNARK proof on the blockchain for decentralized verification.

- Process:

- The zk-SNARK proof, along with relevant metadata (e.g., document hash), can be recorded on the blockchain.

- Any future verifier can use the stored zk-SNARK proof and the public key to verify the document's compliance with the legal standards without revealing the sensitive content of the document.

Zk-SNARKs As A Cryptographic Solution For Data Privacy And Security In The Digital Era

RESULT

Upon successful verification, the document is proven to be valid without exposing the sensitive information within.

Ligerolight: Optimized zk-STARKs protocol for Blockchain Scalability.

OBJECTIVES:

- Enable Scalable and Efficient Notarization
- zk-STARKs does not requires a trusted setup, ensuring the integrity of the notarization process without relying on external or centralized entities.
- Supporting decentralized file notarization systems for enhanced transparency and privacy.

Solution Strategy

1: Define the Computational Problem
Identify the specific computation (e.g., smart contract state transition) that you need to prove without revealing the underlying data.
2: Choose the Mathematical Model (Arithmetic Circuits)
Convert the computation into an arithmetic circuit, which can be evaluated in a zk-STARK-friendly format.
3: Interactive Oracle Proof (IOP) Construction
zk-STARKs use Interactive Oracle Proofs (IOP), which involve multiple rounds of interaction between the prover and verifier. The verifier queries a few locations in the oracle's output to verify the proof's validity.

Solution Strategy

4: Batch zk-IPA (Zero-Knowledge Inner Product Argument)
To enhance efficiency, zk-STARKs (like Ligerolight) implement a batch zk-IPA. This allows the prover to process multiple proofs at once, reducing computational overhead and communication size.
5:Commit to Data Using Merkle Trees
Data commitments are often made using Merkle Trees, a structure that allows you to commit to a large dataset (e.g., blockchain state) with a small root hash. zk-STARKs rely on these structures to ensure data integrity.
Process:
- Generate Merkle root for the transaction or state data.
- Use the Merkle proof path to verify certain data points (like individual transactions) without revealing the entire dataset.

Solution Strategy

6: Polynomial Commitments and Reed-Solomon Codes
Encode proof elements using polynomials. The verifier checks if the encoded polynomials match the claimed outputs using Fast Fourier Transform (FFT) and other algebraic checks.
7: Prover-Verifier Interaction (Non-Interactive Mode)
zk-STARKs do not require a trusted setup, and can be verified without back-and-forth communication once the proof is generated.
Process:
- The prover computes the proof and sends it to the verifier.
- The verifier checks the proof by performing several algebraic checks on a few selected data points (oracle queries).
- If all checks pass, the verifier accepts the proof as valid.

Solution Strategy

8: Deployment on Blockchain

Once the proof is generated, it can be posted on the blockchain as a verifiable proof of the transaction or state transition.

The blockchain nodes (miners/validators) verify the proof using the public zk-STARK verification algorithm, ensuring that the transaction is valid without needing to see the transaction's details.

Ligerolight: Optimized zk-STARKs protocol for Blockchain Scalability.

RESULT

- By following these steps, zk-STARKs can be implemented to provide scalable, transparent, and secure zero-knowledge proofs, suitable for blockchain scalability and privacy applications. They ensure privacy without trusted setups and offer long-term security against quantum threats, making them a powerful tool for decentralized systems.

Bulletproofs: Short Proofs for Confidential Transactions and More

OBJECTIVES:

- Create a zero-knowledge proof system that does not require a trusted setup, improving security and trust in decentralized systems like blockchains.
- Minimize Proof Size for Efficient Storage.
- Maintain High Verification Efficiency.
- Enable Aggregation of Proofs

Solution Strategy

1. File Hashing and Metadata Extraction

Hash the file (e.g., using SHA-256) to create a unique fingerprint of the file.

- Extract relevant metadata (e.g., creation time, file size) that will be notarized along with the hash.
- Ensure that only the file's hash and metadata are notarized, preserving the privacy of the file content.

Solution Strategy

2. Generate Bulletproof for the File

Use Bulletproofs to create a proof that demonstrates the file's hash and metadata without revealing sensitive details. Bulletproofs are efficient for proving certain properties such as:

- The file is within a certain size range.
- The timestamp falls within a valid range.
- The file's hash matches the original file.

3. Batch Aggregation for Multiple Files

Use Bulletproof's aggregation capability to batch notarize multiple files simultaneously, reducing computational and storage overhead.

Solution Strategy

4. Merkle Tree Integration

Store the file hashes in a Merkle Tree structure, with the root of the tree being notarized on the blockchain. This allows you to:

- Efficiently prove the inclusion of any file's hash without revealing all files in the tree.
- Use Merkle proofs to verify file integrity.

5. Post the Proof on Blockchain

Post the compact Bulletproof and the Merkle root of the hashed files on the blockchain. This allows decentralized and immutable storage of the notarization proof.

Solution Strategy

6. Verification by Third Parties

Enable third parties to verify the file's hash and metadata by referencing the Merkle root and proof.

7. Privacy and Security

Ensure privacy using Bulletproofs and maintain *post-quantum security* by relying on cryptographic primitives.

8. Optimize for Scalability

Optimize the proof size and verification time using Bulletproof's logarithmic growth for large datasets.

- Implement batching and aggregation techniques for large-scale notarization scenarios.

Bulletproofs: Short Proofs for Confidential Transactions and More

RESULT

- The implementation of Bulletproofs in file notarization successfully preserved privacy by notarizing only the file's hash and metadata, producing compact proofs of 2-3 KB, which reduced storage and transmission costs.

Comparative Analysis

Attribute	zk-SNARKs	zk-STARKs	Bulletproofs
Trusted Setup	Requires a trusted setup (vulnerable if compromised)	No trusted setup required (more decentralized)	No trusted setup required
Proof Size	Very small(200-300 bytes)	Larger(tens to hundreds of KB)	Moderate (2-3 KB)
Verification Time	Very fast(constant time)	Slower(due to larger proof sizes)	Slower than zk-SNARKs, similar to zk-STARKs
Computation Time	Fast proving but expensive setup	Slower proving but scales better	Moderate proving time
Scalability	Suitable for smaller-scale systems	Highly scalable, best for large computations	Moderately scalable, good for medium-scale

Conclusions

- zk-SNARKs are efficient with small proofs and fast verification, ideal for scenarios needing minimal storage. However, their reliance on a trusted setup and vulnerability to quantum attacks limit their long-term security in decentralized systems.
- zk-STARKs are scalable, transparent, and post-quantum secure, requiring no trusted setup. However, their larger proof sizes and slower verification times can be challenging in resource-constrained environments.
- Bulletproofs provide a no-trusted-setup option with moderately small proofs, though larger than zk-SNARKs. They have slower verification times and lack post-quantum security but are well-suited for confidentiality-focused applications, especially range proofs.

References

- Santoso, I., Christyono, Y. (2023). Zk-SNARKs As A Cryptographic Solution For Data Privacy And Security In The Digital Era. International Journal of Mechanical Computational and Manufacturing Research, 12(2), 53–58. https://doi.org/10.35335/computational.v12i2.122
- Z. Zhang, W. Li, X. Liu, X. Chen and Q. Peng, "Ligerolight: Optimized IOP-Based Zero-Knowledge Argument for Blockchain Scalability," in IEEE Transactions on Dependable and Secure Computing, vol. 21, no. 4, pp. 3656-3670, July-Aug. 2024, doi: 10.1109/TDSC.2023.3336717.
- B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille and G. Maxwell, "Bulletproofs: Short Proofs for Confidential Transactions and More," 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2018, pp. 315-334, doi: 10.1109/SP.2018.00020.

Thank you