Zero Knowledge Proofs Implementation for Secure Voting System using Elliptic Curve Digital Signatures

I. Introduction

A. Definition of Zero-Knowledge Proofs (ZKPs)

A Zero-knowledge Proof (ZKP) is a protocol in cryptography that allows one party, the prover, to demonstrate the validity of a statement to another party, the verifier, without revealing any additional information beyond the fact that the statement is true. ZKPs offer a powerful way to achieve privacy and security in a wide range of applications by ensuring that sensitive information remains hidden even during the verification process.

In simple terms, a ZKP can be compared to a secret handshake between two parties. In a secret handshake, two people shake hands in a particular way to confirm that they know each other without revealing their identity to anyone else. Similarly, a ZKP allows a prover to convince a verifier that they know something (such as a password or private key) without revealing that information to anyone else.

What are Zero Knowledge Proofs?

B. Applications of ZKPs in various domains

One of the most important advantages of ZKPs is that they provide robust protection against data breaches, identity theft, and other cyber threats while maintaining transparency and integrity. For example, ZKPs can be used to ensure that only authorized parties can access certain information, such as medical records or financial data. ZKPs can also be used in digital currencies to prevent fraud and double-spending.

Zero knowledge proofs (ZKPs) have numerous applications in various domains, such as secure authentication, privacy-preserving data sharing, blockchain technology, digital currencies, and password security. ZKPs are particularly useful in situations where data breaches, identity theft, and other cyber threats are common, as they can provide robust protection against such threats while maintaining transparency and integrity.

C. Objective of the project

The objective of this project is to design and implement a secure voting system using elliptic curve digital signatures and zero knowledge proofs. The project aims to develop a client-server architecture where voters can prove their eligibility to vote without revealing their private key, ensuring both privacy and security during the voting process. This paper will provide an overview of zero knowledge proofs, compare ZKP protocols, and present a detailed description of the secure voting system's design and implementation.

II. Literature Review

A. Overview of existing ZKP protocols

1. *zk-SNARKs*

zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) are a class of non-interactive zero-knowledge proofs that enable a prover to demonstrate the validity of a statement efficiently, without revealing any additional information. Non-interactive means that the prover and verifier only need to exchange messages once, without any further interaction required. This is different from interactive zero-knowledge proofs, where the prover and verifier need to exchange multiple messages back and forth to complete the proof.

In non-interactive zero-knowledge proofs, the prover can generate the proof ahead of time and send it to the verifier, who can then verify the proof without any further communication with the prover. This is particularly useful in situations where the prover and verifier may not be online at the same time, or where multiple parties may need to verify the same proof independently.

zk-SNARKs are particularly known for their succinctness, meaning that the proofs are small in size and can be verified quickly. zk-SNARKs are widely used in blockchain applications, such as Zcash, a privacy-focused cryptocurrency.[1]

2. *zk-STARKs*

zk-STARKs (Zero-Knowledge Scalable Transparent ARguments of Knowledge) are another class of non-interactive zero-knowledge proofs that offer similar benefits to zk-SNARKs but with improved transparency and scalability. Unlike zk-SNARKs, zk-STARKs do not rely on a trusted setup, making them more secure against potential vulnerabilities associated with the setup process. zk-STARKs are also more resilient to quantum attacks due to their use of post-quantum cryptographic primitives. However, zk-STARKs generally have larger proof sizes and higher computational complexity compared to zk-SNARKs.[2]

3. *Schnorr*

The Schnorr protocol is a simple and efficient interactive zero-knowledge proof scheme based on the hardness of the discrete logarithm problem. It allows a prover to demonstrate knowledge of a discrete logarithm without revealing it. The Schnorr protocol is widely used in cryptographic applications, such as digital signatures (Schnorr signatures), secure authentication, and secure voting systems. The protocol is known for its simplicity and relatively low computational overhead, making it easier to implement and more efficient than some other ZKP schemes

[1] "zk-SNARK vs zkSTARK - Explained Simple - Chainlink Blog." 19 Feb. 2023, https://blog.chain.link/zk-snarks-vs-zk-starks/.
[2] "zk-SNARK vs zkSTARK - Explained Simple - Chainlink Blog." 19 Feb. 2023, https://blog.chain.link/zk-snarks-vs-zk-starks/.

B. Overview of ECDSA

ECDSA (Elliptic Curve Digital Signature Algorithm) is a widely used digital signature algorithm based on elliptic curve cryptography, offering improved security and performance compared to traditional schemes. With shorter key lengths and faster computations, ECDSA enables secure communication, authentication, and transactions in various applications, including cryptocurrencies. The algorithm involves generating a key pair (private and public keys), hashing the message to be signed, generating a signature using the private key, and verifying the signature using the public key, all while leveraging the mathematical properties of elliptic curves to ensure robust security.[3]

ECDSA can be used to implement Zero Knowledge Proofs by using the public key to verify the private key. Hence, a user can be verified simply by providing a signature and public key, and withholding all other details

C. Selection of an appropriate ZKP protocol for the project

This project will use Elliptic Curve Digital Signatures to generate key-pairs of shorter lengths, and will use this to implement an e-voting system that uses zero-knowledge proofs.

[3] "What is ECDSA Encryption? How does it work?." https://www.encryptionconsulting.com/education-center/what-is-ecdsa/.

III. System Design

A. System Architecture

1. *Components and their interactions*

The proposed secure voting system consists of the following key components:

- Voter Client: Represents individual voters, providing a user interface for registration, authentication, and voting.
- Voting Server: Represents the election authority, responsible for managing the registration of voters, verifying voter authentication, and recording votes.
- Voting Database: A database maintained by the voting server to store anonymized votes securely, as well as user details

The components interact as follows:

- During the registration phase, the voter client generates a public-private key pair and sends the public key to the voting server, which stores it in the database.
- During the authentication phase, the voter client initiates the key-pair generation by sending the required data to the voting server, which verifies the proof without learning the voter's private key.
- Once authenticated, the voter client submits a vote to the voting server, which records the anonymized vote in the voting database.

2. *Data Flow Diagram*

A data flow diagram illustrating the interactions between components is as follows:

- Voter Client -> Voting Server: Public key, and other details (during registration)
- Voter Client -> Voting Server: Signature (during authentication)
- Voting Server -> Voter Client: Authentication result
- Voter Client -> Voting Server: Vote (after successful authentication)
- Voting Server -> Voting Database: Anonymized vote

B. User Interface Design

The user interface for the voter client will be designed to provide a simple and intuitive experience for the users. It will include the following elements:

- Registration form: Collects the user's public key during the registration process.
- Login form: Initiates the authentication process.

- Voting form: Allows authenticated users to cast their votes and submit them to the server.

C. Selection of Technologies and Tools

The following technologies and tools have been selected for the implementation of the secure voting system:

- Programming Language: Python, due to its simplicity, versatility, and the availability of libraries for cryptographic operations.
- Programming Framework: Flask, due to its lightweight and versatile nature and ability to create quick APIs
- Cryptography Library: “elliptic”, an NPM package for elliptic curve digital signatures, and ‘cryptography’, a Python library which will be used to verify the digital signatures.
- Database System: SQLite, a lightweight and easy-to-use relational database system for storing public keys and anonymized votes.

These technologies and tools provide a balance between ease of use, performance, and security, making them suitable choices for the implementation of the secure voting system.

IV. Implementation of a Secure Voting System

This paper aims to implement a secure voting system using elliptic curve digital signatures. The system will leverage a client-server architecture where clients represent the voters and the server represents the voting system or election authority. Voters will authenticate themselves by proving possession of a valid private key associated with a registered public key, without revealing their identity or private key.

Git repository hosting the project
Frontend hosted
Backend hosted

V. Evaluation

A. Security analysis

1. Privacy preservation

The Privacy of the system is not completely robust due to the lack of information and scale. To create a completely anonymized version of this project, further unique identification from the user would be required, for example, Aadhar card. Once a valid Aadhar card is given, the privacy could be preserved as following:

Step 1: Create a public key registry wherein all user public keys are stored
Step 2: Create an aadhar card registry where registered aadhar cards are stored
Step 3: Upon registration, these values are stored in the respective registries, and hence the anonymity of the user would be preserved, as they would not be mapped to a user
Step 4: When a user votes, the user's signature would be checked across the entire public key registry. If the signature can be verified by a public key in the registry, the user’s vote would be stored, and their public key would be removed from the registry.
Step 5: The user cannot vote again, as their secret key would now not have a corresponding public key. Furthermore, they cannot sign up again, as the Aadhar card is being stored in the registry, and they cannot use it to generate a new key.

Hence, the privacy of a user might be preserved through this method.

2. Resistance to attacks

The system’s resistance to attacks would be a feature of the platform it is being hosted on. However, there is sensitive details stored in the registry, although it is not mapped to any user

B. Performance analysis

As compared to a system with no anonymization, the computation time for generation of keys and signing does pose a computational overhead. However, as compared to other ZKP systems, ECDSA is computationally quicker and uses shorter keys for better efficiency

C. Comparison with non-ZKP based systems

Compared to non-ZKP based systems, the ZKP method offers better privacy and security guarantees. Non-ZKP based systems rely on the use of encryption and decryption techniques, which can be vulnerable to attacks, such as side-channel attacks. Additionally, non-ZKP based systems may reveal the voter's identity or their voting choice to the system administrator, which is a breach of privacy.

VII. Conclusion

A. Summary of project findings

In this project, we have designed and implemented a secure voting system based on a zero-knowledge proof scheme using elliptic curve digital signatures. The system provides a client-server architecture where voters can prove their eligibility to vote without revealing their identity or private keys, ensuring both privacy and security during the voting process.

The security analysis of the system shows that it provides privacy preservation and anonymization of votes. The ECDSA protocol's efficiency makes the system computationally less expensive than other ZKP protocols, such as zk-SNARKs and zk-STARKs, which translates to faster execution times crucial for large-scale voting systems.

In terms of performance analysis, the computation time is minimal, making the system highly scalable. Comparing the system with non-ZKP based systems, this secure voting system provides a higher level of security and privacy protection while also being more efficient.

Overall, the project findings show that ECDSA is a suitable choice for building secure voting systems that provide strong privacy guarantees, are resistant to attacks, and are computationally efficient.

B. Potential future improvements

The system can be improved by implementing the steps outlined in the "Privacy Preservation" section. To reiterate:

Step 1: Create a public key registry wherein all user public keys are stored
Step 2: Create an aadhar card registry where registered aadhar cards are stored
Step 3: Upon registration, these values are stored in the respective registries, and hence the anonymity of the user would be preserved, as they would not be mapped to a user
Step 4: When a user votes, the user's signature would be checked across the entire public key registry. If the signature can be verified by a public key in the registry, the user's vote would be stored, and their public key would be removed from the registry.
Step 5: The user cannot vote again, as their secret key would now not have a corresponding publi key. Furthermore, they cannot sign up again, as the Aadhar card is being stored in the registry, and they can't use it to generate a new key.

In a large scale system, this will introduce a large computational overhead. Hence, measures for searching and finding the right key would be required. For example, the keys and IDs of users might be placed into buckets for easier searching.

VIII. References

1. https://www.youtube.com/watch?v=GvwYJDzzI-g
2. https://blog.chain.link/zk-snarks-vs-zk-starks/
3. https://www.youtube.com/watch?v=mV9hXEFUB6A&ab_channel=Theoretically
4. https://www.encryptionconsulting.com/education-center/what-is-ecdsa/

* AI tools were used in the implementation of this project