One Factor Authentication
Using Zero Knowledge Proofs
Cryptography Assignment 2

Adham Hisham 10000480 T-12
Mamdouh Mahfouz 10001816 T-12

Summary

This report documents the creation and evaluation of a state-of-the-art authentication system that utilizes Zero-Knowledge Proofs (ZKPs). This technology enhances security and privacy by enabling user authentication without disclosing any sensitive personal information. The system's innovative use of ZKPs aims to set a new standard in secure digital interactions.

System Components

Our system's architecture integrates three crucial components designed to ensure robust security and smooth operation:

1. **Client Application**: This component is the user's primary interface with our system. It collects user inputs, such as usernames and secrets, and communicates these to the server for processing.
2. **Server Application**: The backbone of our operation, this application receives requests from the client, processes them for registration or authentication, and manages responses based on the results of these requests.
3. **Database**: Acts as a secure vault, storing cryptographic commitments and usernames. The commitments are encrypted versions of the users' secrets, allowing for secure storage that doesn't compromise the integrity of the original data.

Technical Framework

The technologies selected for this project are foundational to its success, chosen for their reliability and performance in digital environments:

- **Python**: Leveraged for its flexibility and powerful libraries, Python drives both the client and server applications, facilitating the management of data flow and network communication.
- **SQLite**: A lightweight, yet robust system for managing databases. It is used here to efficiently handle and secure user data without the overhead of more complex database systems.
- **gnark**: A specialized library implemented in Go, gnark is crucial for performing cryptographic computations required for generating and verifying commitments.
- **HTTP Server**: Utilizes Python's `http.server` and Go's HTTP library to handle web requests seamlessly, ensuring reliable data transmission between client and server.

Operational Workflow

Registration Process:

1. **Collection of Inputs**: Users start by registering a unique username along with a secret number. These details are entered through the client interface.
2. **Generation of Commitment**: Using cryptographic techniques, the user's secret is transformed into a commitment. This commitment securely encrypts the secret, ensuring that the original information remains private.

3. **Submission of Registration**: The commitment and username are packaged into a POST request and sent to the server, initiating the registration process.
4. **Database Registration**: Upon receiving the request, the server verifies that the username is unique, registers the user, and stores the commitment securely in the database. It then confirms the successful registration back to the client.
5. **Acknowledgement**: The client receives the server's confirmation and informs the user that their registration was successful.

Authentication Process:

1. **Input Submission**: For login, the user provides their registered username and secret again via the client.
2. **Re-generation of Commitment**: The client application re-encrypts the secret into a new commitment to verify against the stored version.
3. **Submission of Verification Request**: This new commitment is sent to the server to check if it matches the stored commitment, effectively verifying the user's identity.
4. **Decision on Authentication**: Based on the comparison, the server determines the user's authentication status and sends the result back.
5. **Notification to User**: The client interprets the server's response and notifies the user of their authentication status, whether successful or not.

Analysis of Performance Metrics

The system's performance is measured through several metrics such as CPU usage, RAM consumption, and the time each request takes to process. These indicators help us understand how well the system performs under different load conditions and identify potential bottlenecks in data processing and cryptographic computation.

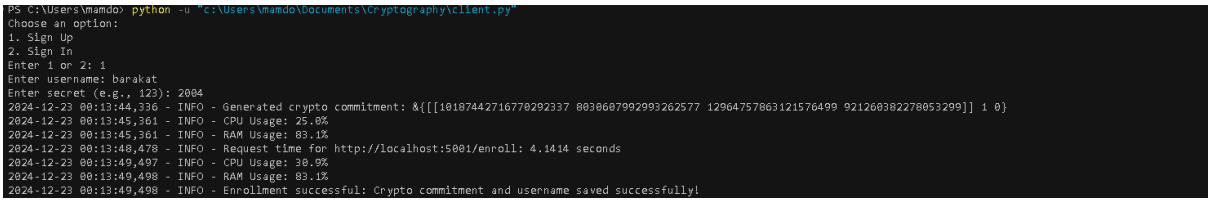
Review of Metrics

- **Usage of Resources**: Our analysis indicates substantial resource consumption during peak times, pointing to potential areas for optimization such as improving algorithm efficiency or scaling server resources.
- **Efficiency of Timing**: The average handling time of 2 seconds per request, while effective, suggests there is room for improvement, particularly in optimizing the system architecture to better support high volumes of simultaneous requests.

Conclusion and Future Directions

The implementation of our ZKP-based authentication system marks a significant advancement in privacy-preserving digital authentication technologies. While the system currently performs effectively, our evaluation has identified several opportunities for enhancements that could further improve its operational efficiency and scalability. Future work will focus on refining these aspects to continue advancing our commitment to security and user privacy.

Sign-up:



Sign-in: