SilentVault: A Password-Based Zero Knowledge File Server

Codreanu Andrei Daniel
Facultatea de Sisteme Informatice si Calculatoare
Academia Tehnica Militara "Ferdinand I"
Bucuresti, Romania
andreidanielcodreanu@gmail.com

***Abstract*—SilentVault is a secure client-server file storage system that employs a password-based Zero Knowledge Proof authentication protocol based by Schnorr identification. By doing away with reusable password verifiers, the system guarantees that user passwords are never sent or kept on the server. Elliptic curve cryptography is used for authentication, which is carried out over a TLS channel and offers robust security guarantees even in the case of server compromise.**

***Index Terms*—Zero Knowledge Proof, Schnorr Protocol, File Server, Authentication, Cryptography**

I. Introduction

Although password-based authentication is still the most popular method for remote access systems, it has intrinsic flaws like credential reuse and offline brute-force attacks. Database compromise is a critical security event because traditional solutions based on salted password hashes still expose reusable verifiers to the server.

SilentVault uses a password-based Zero Knowledge authentication method to get around these restrictions. The server uses algebraic relationships defined over elliptic curve groups to confirm that it knows the password without ever receiving it.

II. System Architecture

SilentVault follows a modular client–server architecture. Communication between the client and server is secured using TLS, while authentication and file operations are handled at the application layer.

Main components include:

- Client module responsible for authentication and file requests
- Server module managing sessions and file storage
- CryptoService implementing the Schnorr-like protocol
- Payload and resolver layers for protocol abstraction

This separation improves maintainability and allows cryptographic components to evolve independently of network logic.

A. Functionalities

In order to provide an example of a ZKP applied to a real-life scenario, SilentVault allows the client to do a couple operations regarding files. He can also manage uploaded files, as follows:

- Uploads files to server.
- Downloads previously uploaded files.
- Checks a list of all the previously uploaded files.
- Demands the server to delete an uploaded file.

The server, besides managing user's files and public data, implements a minimal Command Line Interface, having only the functionality to print a statistic of how many users are currently connected to the server.

Both the client and the server have verbose elements.

III. Zero Knowledge Authentication Protocol

A. Mathematical Model

The protocol uses an element g to generate a cyclic group G of prime order q. In reality, G is instantiated as a group of elliptic curves (P-256).

A user password P is transformed into a secret scalar:

$$x = \mathrm{KDF}(P, \mathrm{salt}) \bmod q \tag{1}$$

The corresponding public value is:

$$Y = g^x \tag{2}$$

B. Authentication Steps

For each authentication session:

1) Client selects random $r \in \mathbb{Z}_q$ and computes $R = g^r$
2) Server sends a random challenge c
3) Client responds with $s = r + cx \bmod q$
4) Server verifies:

$$g^s \stackrel{?}{=} R \cdot Y^c \tag{3}$$

Successful verification proves knowledge of x without revealing it.

C. Security Properties

The protocol satisfies:

- Completeness: honest clients are always accepted
- Soundness: attackers cannot authenticate without knowing x
- Zero Knowledge: transcripts reveal no information about x

IV. Client–Server Sequence

The authentication sequence is as follows:

1) TLS handshake establishment
2) Client sends commitment R
3) Server replies with challenge c
4) Client sends response s
5) Server verifies identity

Upon successful authentication, file operations are enabled over the same secure channel.

V. Comparison with Classical Authentication

A. Hash-Based Authentication

Traditional systems store a verifier of the form:

$$H = \mathrm{Hash}(P||\mathrm{salt}) \tag{4}$$

A compromised database allows offline brute-force attacks, making password security dependent on computational cost alone.

B. Zero Knowledge Approach

Only the public value $Y = g^x$ is stored in SilentVault; this value cannot be used to verify password guesses offline. It is impossible for a fully compromised server to pose as a client.

C. Security Comparison

Property	Hash + Salt	ZKP
Password exposure	Indirect	None
Offline attack	Yes	No
Replay attack	Possible	Impossible
Server compromise	Critical	Limited

VI. Implementation

A. Modules

The application is implemented using a few modules, some common between the client and the server.

1) "payload" module: This module defines the format of the messages sent between the client and the server. It handles serialization and de-serialization, from raw byte strings to concrete classes and vice-versa.

2) "resolver" module: This module is specific only to the server. It handles the resolving of different requests received from the client, in a polymorphic manner, according to *Object Oriented Programming* paradigms.

3) "data" module: It handles the effective management of stored data on the server, like:

- Reading files.
- Writing files.
- Disk organization.
- Executing a double-encryption on the data received from each server.

B. Languages used

The project is written mainly in *C++*, but the cryptographic operations are written in *Python* for easier implementation.

VII. Conclusion

Zero Knowledge authentication can be successfully incorporated into a workable file server architecture, as SilentVault shows. The system greatly lessens the impact of server compromise and offers a solid basis for secure distributed storage systems by doing away with reusable password verifiers.

Future research will focus on end-to-end encrypted file storage and non-interactive ZK protocols.

References

[1] C. Schnorr, "Efficient Identification and Signatures for Smart Cards," *Advances in Cryptology – CRYPTO*, 1989.

[2] D. Hankerson, A. Menezes, S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, 2004.

[3] O. Goldreich, *Foundations of Cryptography*, Cambridge University Press.