TWO FACTOR ZERO KNOWLEDGE PROOF AUTHENTICATION SYSTEM

S.Lekhesh (Roll No.: 22CSB0C03)

Aman Raj (Roll No.: 22CSB0C02)

Ayush Gupta (Roll No.: 22CSB0C01)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL

AY 2024-2025

1 Introduction

1.1 Problem Addressed

In today's digital landscape, securing user authentication is critical, particularly when logins occur from untrusted devices or over compromised networks. Conventional authentication systems are vulnerable to attacks such as hardware and software keyloggers, as well as shoulder surfing. These risks expose sensitive information like passwords, compromising user security.

1.2 Importance of the Problem

While traditional two-factor authentication (2FA) provides an extra layer of security by requiring a secondary device, it still involves transmitting credentials over networks. This transmission leaves credentials susceptible to interception by malicious attackers, further diminishing the effectiveness of 2FA systems.

1.3 Existing Solutions and Limitations

Current solutions primarily involve 2FA systems that rely on password and token verification. However, these approaches lack resilience against sophisticated attacks like man-in-the-middle (MITM) attacks. Additionally, conventional authentication

methods rely heavily on the assumption that networks are secure, which is often not the case.

1.4 Key Contributions of the Project

To address these challenges, this project proposes a **Zero-Knowledge Proof (ZKP)** based two-factor authentication system. The key contributions of the project include:

- **Enhanced Security:** Ensuring no sensitive information is transmitted over the network using ZKP.
- **Resistance to Attacks:** Preventing keylogging, eavesdropping, and MITM attacks by applying ZKP for authentication.
- **Efficient Authentication:** Maintaining a practical, efficient process for users without compromising security.
- **Compatibility:** Ensuring usability on resource-constrained devices.

2 Objectives

- Implement a secure two-factor authentication system using Zero-Knowledge Proof to prevent credential exposure.
- Develop a robust authentication protocol resistant to keylogging, shoulder surfing, and network-based attacks.

3 Implementation and Results Analysis

3.1 Stages Involved and Working of the System

There are two stages involved in the system. They are:

1. **Signup Stage/ Account Creation:** This is the stage in which the user creates an account with a website or other service.
2. **Login Stage:** This stage consists of the steps necessary for the user to log in.

Working of the System

Two-Factor Authentication:

a) Account Creation: Before the user can log into an account with a service, the user must have an account with that service. To begin, the user selects a username and communicates it to the server via the trusted device. If this username is already taken, the user is informed with an error message, and the user must make another attempt. When the user identifies an unused username, the process can continue.

In the next step, the user selects a password and enters it into the trusted device. The trusted device uses that password to generate a secret key. The trusted device then randomly generates and saves a secret key for itself. The public keys associated with each of the two secret keys are then computed. After that, the username and both public keys are sent to the server, which stores this data. Once this process is complete, the trusted device deletes all references to the user's password or the associated public key. This set of steps is summarized in Figure 1.

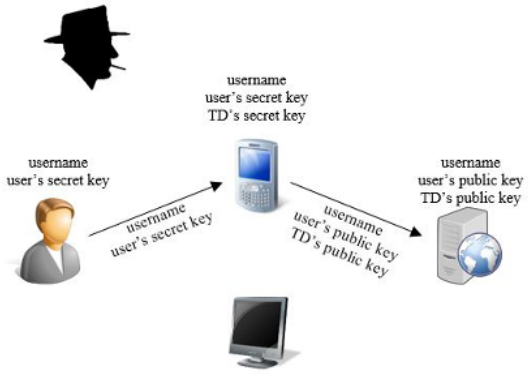

Figure 1: Summarized account creation process

b) Login Stage: The user, having created an account, may find that they wish to log in from an untrusted device. They begin by selecting the service that they wish to log into on the trusted device. The device sends a message, digitally signed with the device's secret key, to the server indicating the intent of the user (as identified by their username) to log in. The server saves in its records that a login attempt has begun. This login attempt expires within a minute if it is not continued. The steps so far have been summarized in Figure 2.

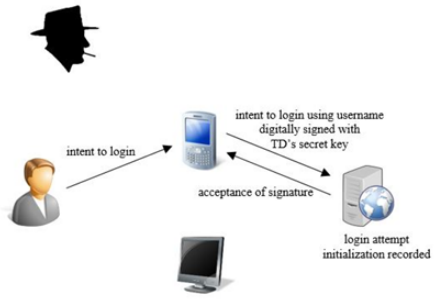

Figure 2: Summarized initial login process

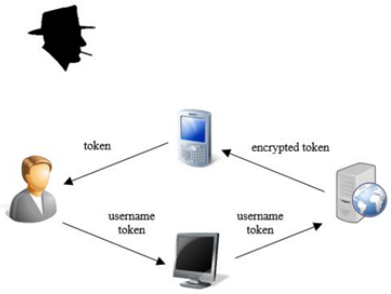

Figure 3: Summarized halfway login process

Next, the trusted device requests that the user enter their password. Once the password is entered, the trusted device computes the user's secret key from the password. Then, the trusted device runs through several rounds of zero-knowledge proof with the server to demonstrate knowledge of the user's secret key. All messages sent are signed with the trusted device's secret key, ensuring their authenticity.

When the server is sufficiently convinced, it generates a random token and encrypts it using the trusted device's public key. This encrypted token is sent to the trusted device, which decrypts it and displays the token on-screen. The user then enters their username and the token into the untrusted device, which sends this information to the server. These steps are shown in Figure 3.

The server checks for a valid login attempt and verifies that the token matches. If it does, the untrusted device is informed of successful halfway login. The untrusted device then displays this information to the user, who confirms it on the trusted device. The trusted device forwards the confirmation to the server. These steps are summarized in Figure 4.

Finally, the server sends another encrypted token. The trusted device decrypts it, and the user copies the token to the untrusted device. After verifying the token, the server completes the login process, granting access. The trusted device deletes all password references. This final step is summarized in Figure 5.

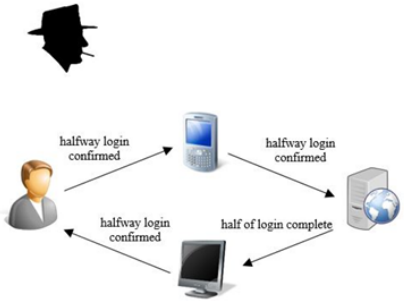

Figure 4: Summarized halfway login confirm process

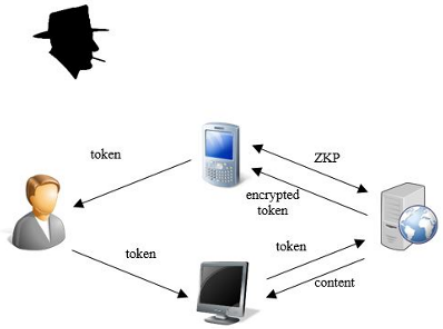

Figure 5: Summarized final login process

3.2 Modification of Diffie-Hellman Key Exchange Algorithm for Zero-Knowledge Protocol

The proposed ZKP based on D-H key exchange algorithm in the sense that both parties (the prover and the verifier) exchange non secret information and did not revealing secrets to get one identical secret key. This means that the prover can prove to the verifier that he knows the secret. The basic D-H key exchange algorithm which is vulnerable to man-in-middle-attack. The proposed version has been developed to resists the man-in-middle attack.

Proposed ZKP Based on Diffie-Hellman Key Exchange Algorithm

To protect the proposed algorithm from a man-in-the-middle attack, encrypted replies (R_1 and R_2) and mutual authentication between the prover (Alice) and the verifier (Bob) are required.

The prover (Alice) proves to the verifier (Bob) that she knows a secret by calculating the key (K) and resending Bob's reply (R_2) to the verifier (Bob) encrypted with the generated secret key (K). Bob encrypts his own reply (R_2) with the generated secret key (K) and matches the two encrypted pieces of information. If matched, then Alice is verified; otherwise, it is rejected.

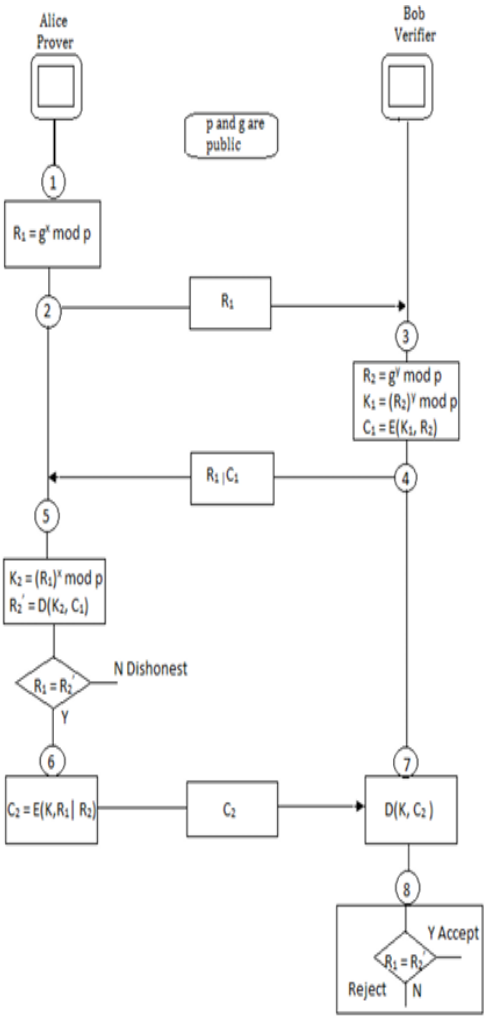

Figure 6: Proposed Zero Knowledge Protocol for key exchange.

The verifier also needs to prove to the prover that he is honest by sending his reply R_1 together with encrypted R_1. Then, the verifier decrypts R_1 using his key and checks if $R_1 = R_1'$. If they match, the verifier is confirmed as honest.

The protocol performed as follows:

3.3 Alice and Bob Protocol

Algorithm 1 Proposed ZKP based on Diffie-Hellman Key Exchange algorithm

Require: Large prime number p, generator g

Ensure: Alice's identity verification

1: Alice chooses a random number x, $0 < x < p$, and calculates $R_1 = g^x \mod p$

2: Alice sends R_1 to Bob

3: Bob chooses a random number y, $0 < y < p$, and calculates $R_2 = g^y \mod p$

4: Bob computes $K_{\text{Bob}} = (R_1)^y \mod p$ and encrypts $C_1 = E(K_{\text{Bob}}, R_2)$

5: Bob sends (R_2, C_1) to Alice

6: Alice computes $K_{\text{Alice}} = (R_2)^x \mod p$, decrypts $R_2' = D(K_{\text{Alice}}, C_1)$

7: Alice verifies $R_2 = R_2'$. If not, Bob is dishonest.

8: Alice encrypts $C_2 = E(K_{\text{Alice}}, R_1, R_2)$ and sends it to Bob

9: Bob decrypts C_2 to retrieve R_1' and R_2'

10: Bob verifies $R_1 = R_1'$. If equal, Alice is verified (Accepted), otherwise rejected.

3.4 Result Analysis

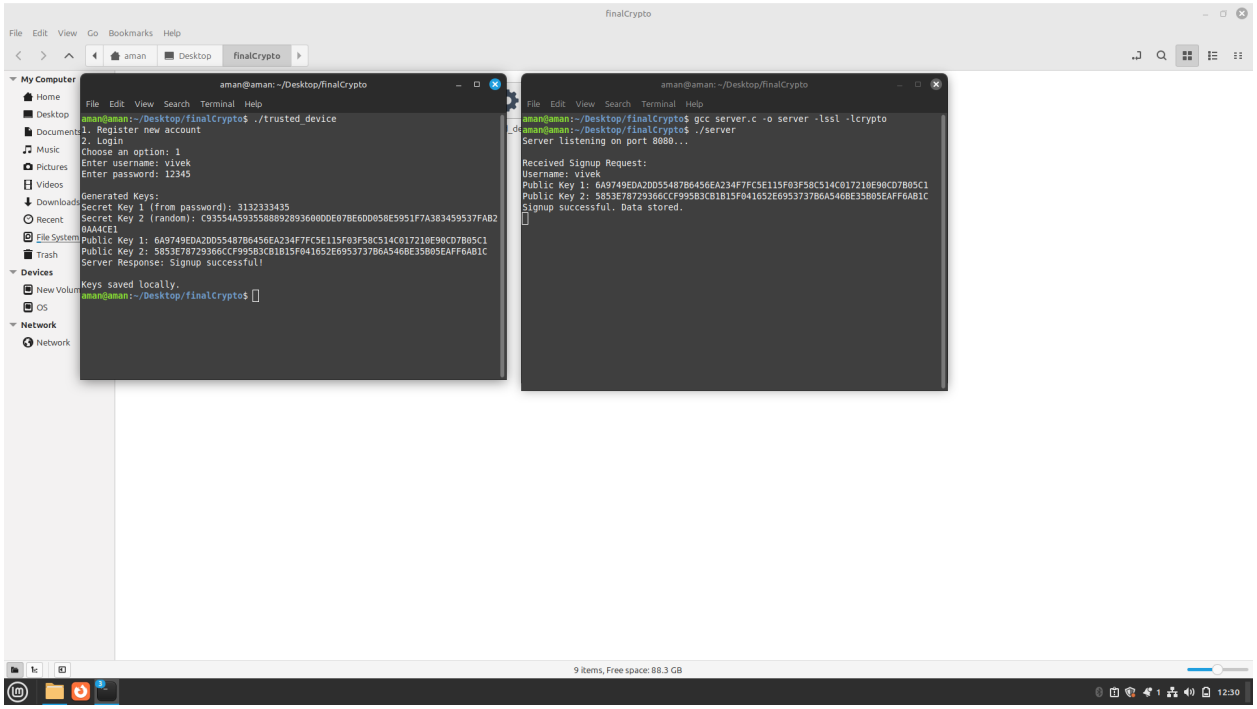

Figure 7: Registration-login.

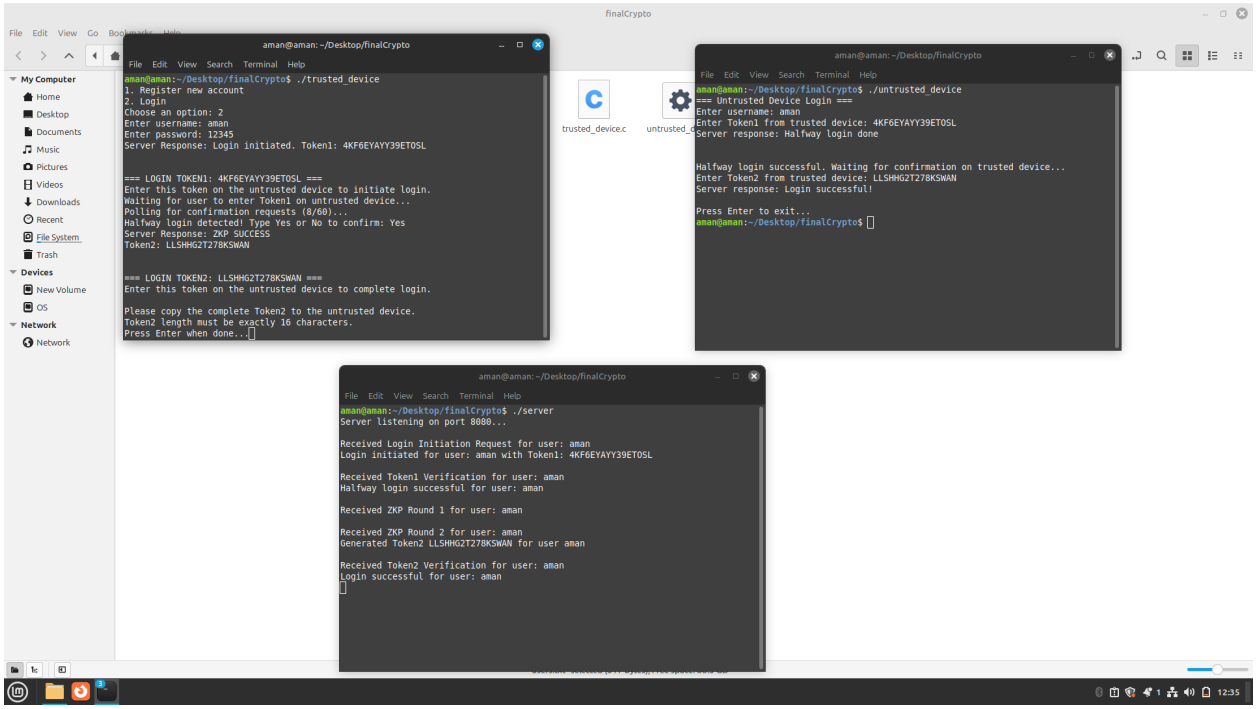

Figure 8: Output of the proposed protocol with untrusted device.

4 Conclusion

Two Factor Authentication is a method commonly used by internet services to provide an extra layer of security in addition to the standard password used as login credentials. It employs a secondary device, such as a phone that the user must have in his or her possession to complete the authentication process. Besides providing a much higher level of security to a web application, there are many other reasons why "Two Factor Zero Knowledge Proof Authentication" is worth implementing. Firstly, it allows for someone with no knowledge on how the protocol works taking advantage of such a concept. Also the system is basically transparent to the user. The two factor authentication provides an extra layer of security. Finally, the simplicity and ease to implement the system is definitely a value-add and a reason to have this in any web application. No additional hardware required to implement this system.Data transmitted through the network is useless to the attackers. Through the use of a one-time token in the hashing function, the information sent over is only valid for once, and thus will not be usable by attackers who intercept the information. Two factor authentication ensures more security.

5 Learning Outcomes

- Gain insight into the principles of zero-knowledge proofs—specifically, their completeness, soundness, and zero-knowledge properties—and how they can be used to authenticate without revealing secret credentials.
- Learn how two-factor authentication can add an extra layer of security by requiring both knowledge (password) and possession (trusted device).
- Understand the vulnerabilities inherent in traditional authentication methods (e.g., keylogging, shoulder surfing, man-in-the-middle attacks).

6 Source Code

You can find the source code on our collaborative GitHub repository:
`https://github.com/ayushg-nitw/2FA_Zero_knowledge_proof`

7 References

Bibliography

[1] David Chaum, Jan-Hendrik Evertse, Jeroen van de Graaf, and Rene Peralta, "Demonstrating Possession of a Discrete Logarithm Without Revealing it," in *Advances in Cryptology — CRYPTO '86*, Lecture Notes in Computer Science, vol. 263, 1987, pp. 200–212.

[2] Jeremy Clark, "Elgamal and CPA-Security, Zero Knowledge," Concordia Institute for Information Systems Engineering, Scribe Notes, March 14, 2013.

[3] Oded Goldreich and Yair Oren, "Definitions and Properties of Zero-Knowledge Proof Systems," *Journal of Cryptology*, 199424, Volume 7, Issue 1, pp. 1–32.

[4] Quan Nguyen, Mikhail Rudoy, and Arjun Srinivasan, "Two Factor Zero Knowledge Proof Authentication System," 6.857 Spring 2014 Project.

[5] International Journal of Advanced Research in Computer Science and Software Engineering, "Research Paper," available online at: `http://www.ijarcsse.com`.

[6] Jitendra Kurmi and Ankur Sodhi, "A Survey of Zero-Knowledge Proof for Authentication," Department of Computer Science & Engineering, Lovely Professional University, Phagwara, Punjab, India.