2D-2FA Software Implementation

Zane Globus-O’Harra, Doug Ure

CS 433 Spring 2023
University of Oregon

Motivations

Firstly, the subject of this project has real-world relevance. Two-factor authentication (2FA) is becoming increasingly more common to authenticate users, and as it becomes more prevalent, attackers will focus more of their efforts on finding ways to break through its layers of security. Our project will help us learn about ways to further increase the security of 2FA by using additional information along with the user's credentials and the server-provided “identifier.”

This is also a learning opportunity for us. Neither of us are very familiar with computer security, and it is something that we are interested in learning about. By completing this project, we will develop valuable technical skills and increase our knowledge base, as well as preparing us for future projects and industry roles.

Lastly, this project could have a real impact on end-users. 2FA enhances users' trust and confidence in online systems by making their personal information and online interactions more secure. This project has the potential to contribute to a larger goal of make a more secure digital environment.

Objectives

- Create a working implementation of 2D-2FA using software
  - Programs for the server and teh device, as described by the paper
  - The implementation will work across multiple devices
  - The implementation will work for multiple users
  - Test cases ensuring the correct functionality of our code
- Documentation describing our design process, the code we have written, and how to use our code. Specifically,
  - Well-commented code
  - A design diagram
  - Installation instructions
  - Usage instructions

References

Shirvanian, M. and Agrawal S., “2D-2FA: A New Dimension in Two-Factor Authentication”. arXiv, CoRR. 2021. (https://arxiv.org/abs/2110.15872)

System Description

The system is composed of two main phases. Our implementation mostly focuses on the Authentication phase of the process, as this phase involves the secure interactions between the user’s device and the server.

- **Registration:** In this phase, the user registers a username and a password with the server, and the server and the device agree on a secret key. The server stores the user’s state, and the device stores the secret key associated with the user account on the server. In our implementation, we assume that this phase has already been completed, and we have created files to store this information for the server and the device.
- **Authentication:** In this phase the parties interact to securely authenticate the user to the server. The user inputs their username and password to the server, and in return, the server displays the user a unique identifier. On the user’s device, the client specifies the server name and inputs the identifier. The device uses the identifier and the current time to generate a PIN to send to the server. The server verifies the PIN and authenticates the user’s session and information. During this process, the server keeps a temporary record of all active sessions and identifiers.

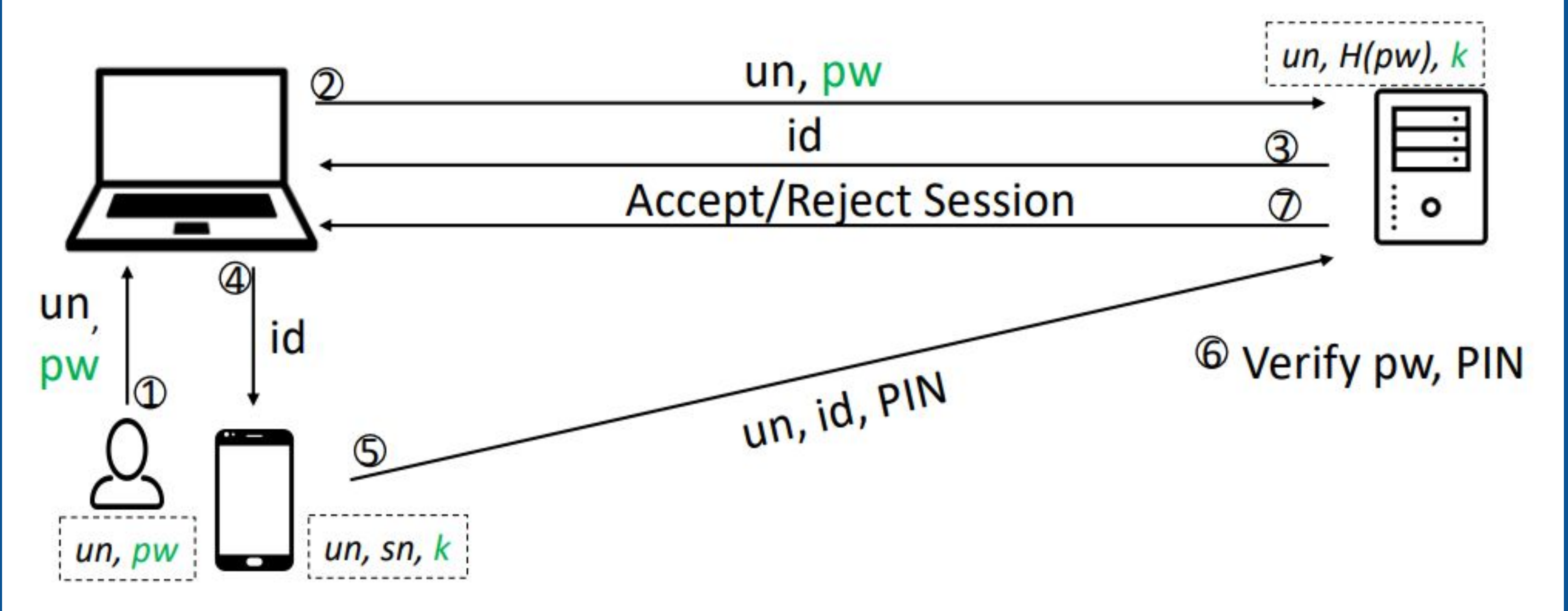
Our Implementation: Our implementation primarily focuses on the authentication phase, as this is where the main meat of the 2D-2FA system is described. We assume that a user has already registered their information with the server, and had a secret key generated. We also use TCP connections instead of TLS recommended by the paper. Additionally, while we have verified that our implementation works across multiple devices, we primarily did our testing by running both the server code and the device code in separate terminal windows on the same machine. The paper’s implementation recommends using a drawing pattern or a QR code as the identifier, however we opted to use a random 6-digit number to reduce the complexity of our implementation.

Security

The 2D-2FA paper lists possible attacks to this system, and provides reasons as to how the system mitigates or eliminates these risks. Those reasons are paraphrased here:

- **Client Compromise:** If the attacker knows the user’s password, they can get an identifier from the server. However, without the user’s device, they don’t have their secret key for the PIN computation.
- **Device Compromise:** The attacker can access the user’s secret key, but they have no information about the user’s password, and thus can not log in to get an identifier from the server.
- **Channel Compromise:** There are three channels, each connecting the client (C), the server (S), and the device (D), as seen in the image below. Similar to the device compromise, the attacker can not get the user’s secret key. While the channel from the device to the server is a regular channel that the attacker can get the identifier and the PIN from, they can not use these to authenticate themselves.
- **User Negligence:** This occurs when a user accidentally approves an attacker’s session. This only happens if a user enters an identifier that doesn’t match the one displayed to them, and does match the identifier given to the attacker. In our implementation, the chances of this occurring is 1 in a million (0.0001%).
- **Attacks on 3rd Parties:** 2D-2FA does not use any 3rd parties or introduce any additional authentication infrastructure into its system beyond those that are already well established (HMAC, SSL/TLS, random number generation, etc.), and thus we do not need to rely on any 3rd parties, eliminating this risk.

This figure shows the steps performed during the authentication phase of 2D-2FA, as well as the communications between the user, the client, the server, and the device. Our code has successfully implemented each of these steps to replicate the functionality of 2D-2FA.


un, H(pw), k
② un, pw
id ③
Accept/Reject Session ⑦
un, pw ①
④ id
⑤ un, id, PIN
⑥ Verify pw, PIN
un, pw
un, sn, k