Two-Factor Physical Authentication Lock System

Dagfinnur Ari Normann
Dept. of Applied Math & Comp. Science.
Technical University of Denmark
s212961@dtu.dk

Diogo Adegas
Department of Electrical Engineering
Technical University of Denmark
s233349@dtu.dk

Jesper Thøgersen
Department of Electrical Engineering
Technical University of Denmark
s203841@dtu.dk

***Abstract*—This paper presents the design and implementation of a two-factor authentication system for unlocking mechanisms. The proposed system integrates a fingerprint scanner and an RFID authenticator as primary authentication methods. The authentication process is executed on separate systems, with outcomes transmitted wirelessly to the locking system. While this initial proof-of-concept involves basic string values for communication, room for encryption features is in place for future iterations. This project report is the collaborative effort of students in *Networked Embedded Systems* at the *Technical University of Denmark*.**

***Index Terms*—dual-factor authentication, esp-idf, rfid-rc522, RFID-Scanner, GT511C3, Fingerprint Scanner, WiFi**

I. Introduction

In response to the imperative for robust security measures to safeguard valuable assets, we propose a two-factor authentication system that amalgamates the dependability of a fingerprint scanner and an RFID authenticator. This paper delineates the technical intricacies and implementation of the system, underscoring its modular architecture.

II. System Architecture

The overall system comprises three ESP32 Ethernet kits [1] (ESP32), an LED, an RFID-RC522 Breakout [2], and a GT511C3 Fingerprint scanner [3]. These components collaboratively orchestrate the two-factor authentication process.

Each subsystem utilizes the ESP IoT Development Framework (ESP-IDF), ensuring a stable and reliable foundation for seamless integration. Within this framework, the primary authentication methods operate on dedicated systems. Source code for all three systems can be found on GitHub [4].

A. Authentication and Transmission

Upon successful authentication via fingerprint or RFID, the system wirelessly transmits messages to the central locking system using UDP. Here, additional authorization checks and logic precede the initiation of the mechanical unlocking process.

B. Locking Mechanism

The locking mechanism resides on the main system, the Server. It consists of a GPIO pin driven to a logic high on unlock or low on lock. In this initial proof-of-concept, an LED, representing the pin, is used to indicate the lock state. The pin is held at logic high for one second before returning to the locked state.

C. Communication Protocol

Wireless communication between authentication systems and the locking mechanism uses the UDP protocol. In the initial project phase, communication includes fundamental string values. While our current focus is not on additional security measures, future steps may explore enhancing security through encryption, reflecting our commitment to continuous improvement and robust data protection.

III. Authentication Sequence

From a high-level perspective, the process is as follows:

Two independent systems have a unique sensor. Each system has a list of authorized tags or fingerprints matched against scan attempts. If the scan is authenticated, the system sends a message to the Server, which has its list of authorized fingerprints and RFID tags. Each authentication system has its MAC address registered on the Server, and each message sent must be associated with the board MAC. The message format must follow <type-mac_address-tag>, with ”-” being the delimiter.

The server authentication is only successful when the finger and the tag are sent from the correct corresponding addresses within less than 3 seconds. When this happens, the dual authentication is successful, the unlock event is triggered, and the lock is open!

A. RFID Client Verification Logic

For this independent logic, we have an ESP32 board and an RFID-RC522 Breakout, which uses SPI as a communication protocol. An external Git Repository [5] was used as boiler plate for handling events in the scanner.

The board’s routine starts by connecting to a predefined WiFi Network. Then, it enters a loop waiting for a tag to read. Every time the Breakout reads a value, there are three possible cases:

If the Breakout reads the master tag (previously saved special tag), it goes into write mode. Write mode will grant access to the Non-Volatile Storage (NVS), either adding or removing a tag from it, depending on whether the following tag is already registered or not.

If the scanner reads an authorized tag, a message containing the authentication type, the board’s MAC address, and the scanned tag to transmit is sent to the Server.

All other tags are ignored if write mode is disabled. No message is sent to the Server. The sequence is displayed in Figure 1 below.

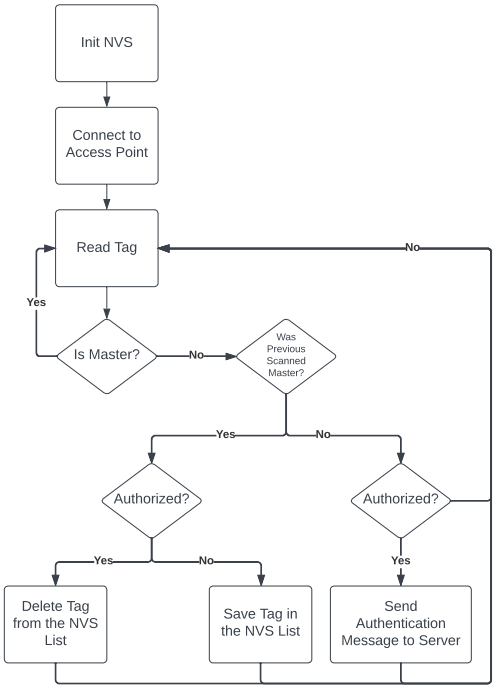

Fig. 1. RFID Authentication Flow Chart

B. Fingerprint Client Verification Logic

The fingerprint verification system consists of an ESP32 and a GT511C3 Fingerprint scanner. The GT511C3 has an onboard ARM Cortex M3 processor, which handles serial communication using UART, fingerprint matching, and storage. Fingerprints are captured using a digital camera with a resolution of 256x256 pixels and fed to the processor, where they are processed by an algorithm and converted to a fingerprint template. The fingerprint template is stored on the flash memory for identification and verification. There is room for 200 fingerprints on the flash memory [3]. Enrolling new fingerprints to the internal flash memory is performed in a different program, which will not be covered.

The fingerprint scanner is idle on power-up and must be instructed to perform fingerprint capture. This is done by turning on the LED of the digital camera and sending a *CaptureFingerprint* command. A scanning attempt is executed, and if successful, the fingerprint will be matched to the templates stored on the flash memory. A message is sent to the Server for cross-validation if there is a match. If there is no match, no message will be sent. This loop will continue till all scan attempts have been carried out. The module will then turn off the LED and wait for 500 ms before resetting the attempt counter and re-initiating the scanning sequence. The wait between scanning sequences is implemented to save power and prevent processor overload.

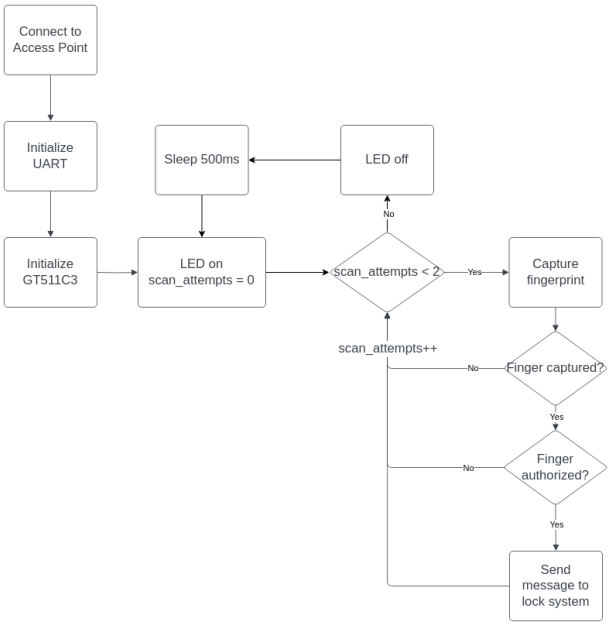

Fig. 2. Fingerprint Authentication Flow Chart

C. Server Cross-Validation Logic

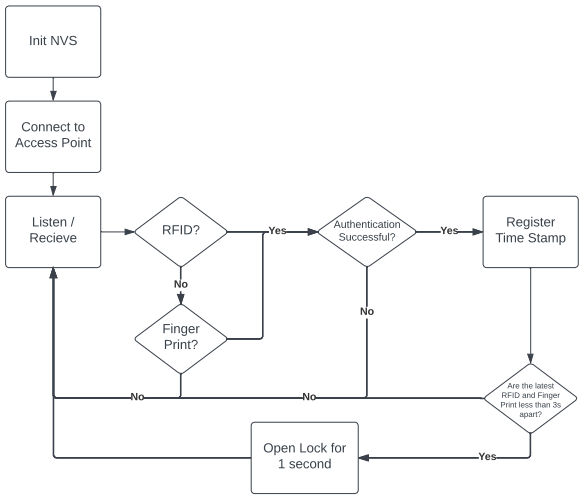

Fig. 3. Server Cross-Validation Flow Chart

This system consists of an ESP32 and an LED. It has two lists of authorized pairs of <mac_address-tag> and <mac_address-fingervalue> for the RFID and the fingerprint scanner, respectively. The Server receives the messages from the two previously described systems. It checks if the incoming message is an RFID or Fingerprint authentication attempt by parsing the message it receives in the aforementioned format by first assessing *type*. After that, the Server checks the corresponding list for pairs of *mac_address-tag* or *mac_address-fingerprint*, depending on the *type*. If the pair is present in the server list of authorized fingerprints/tags, then the timestamp of receiving the message is saved.

Every time there is an incoming message, the Server compares the timestamp of each authentication attempt and checks whether it is smaller than 3 seconds. If so, the authentication is successful. Otherwise, it fails.

IV. Power optimization

Power optimization possibilities were investigated to minimize power consumption. Since the two authentication systems will predominantly operate in an idle state, with sporadic intervals during which they initiate message transmission, it is worth looking into ways to limit idle power consumption. On system start, the system will connect to an Access Point (AP) using WiFi and remain connected. The ESP32 WiFi modem supports three different power-saving (PS) modes: PS_NONE, PS_MIN, and PS_MAX [6]. The power-saving modes allow the WiFi module to sleep only to wake up in fixed intervals to receive WiFi beacon frames. This way, the ESP32 stays connected to the network.

A set of experiments was conducted to measure the power consumption of the ESP32 under varying messaging frequencies and at different WiFi modem power-saving modes. The first series of tests involves sending messages every 1 second, while the second employs a less frequent transmission interval of 10 seconds. The primary objective was to examine the impact of sleep intervals on the total power consumption of the ESP32 while verifying that no messages were lost.

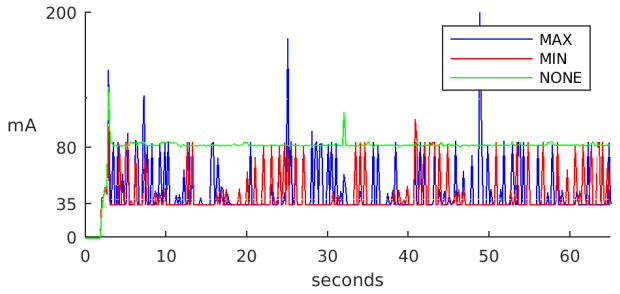

Fig. 4. ESP32 Power Consumption at a 1-Second Message Frequency

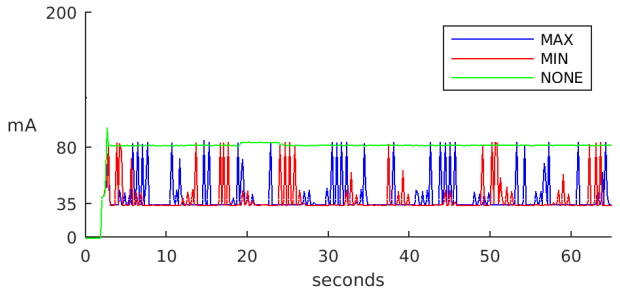

Fig. 5. ESP32 Power Consumption at a 10-Second Message Frequency

The figures 4 and 5 clearly show the difference in power consumption between no power-saving enabled and the MIN/MAX power-saving modes. With no power-saving enabled, the ESP32 continuously consumes approximately 80 mA. With power-saving enabled, the continuous power consumption drops to about 35 mA, with current spikes at around 80 mA in the fixed beacon intervals. No messages were lost in any of the tests.

V. Challenges

Several issues had to be addressed throughout the development of the different systems.

Due to its practicality and extensive online documentation and support, the initial project proposal suggested using ZephyrOS as the Embedded OS. However, after spending considerable time trying to connect to WiFi with no success, we were forced to explore different operation systems. The manufacturer of the ESP32 has its own framework, which we opted for. Using the ESP-IDF, the WiFi issue was immediately surpassed.

Regarding the testing data, the highest frequency at which we were able to record data was five recordings per second, with a sample rate of 40 readings per second. This was the maximum capability of the IDM-8342 multimeter [7] we used. The faster sample rate of 40 readings per second also introduces some uncertainty.

VI. Conclusion

We developed a proof-of-concept for a dual-factor authentication system using two types of sensors: a fingerprint and an RFID scanner. The lock logic drives a digital pin high, indicated by an LED, but can be wired to a solenoid, relay, or the like. Moreover, power optimization testing for the WiFi connection was carried out to limit the power usage without compromising the functionality.

This project lays the ground for applications such as user registration or authorization systems for home IoT. It allowed us to interact with an Embedded OS and apply the concepts we learned in class.

References

[1] Espressif Systems. *Get Started with Ethernet Kit*. 2023. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-ethernet-kit.html.

[2] NXP Semiconductors. *MFRC522 Standard performance MIFARE and NTAG frontend*. 2023. URL: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf.

[3] SparkFun Electronics. *GT-511C1R Datasheet*. 2016. URL: https://cdn.sparkfun.com/datasheets/Sensors/Biometric/GT-511C1R_datasheet_V2-2016-10-25.pdf.

[4] GitHub. *NES project source code*. 2023. URL: https://github.com/dagfinnur/NES.

[5] Alija Bobija. *esp-idf-rc522: C library for interfacing ESP32 with MFRC522 RFID card reader*. 2023. URL: https://github.com/abobija/esp-idf-rc522.

[6] Espressif Systems. *ESP-IDF Programming Guide*. 2023. URL: https://docs.espressif.com/projects/esp-idf/en/latest/.

[7] RS Pro. *IDM-8341/8342 Series Instruction Manual/English*. 2016. URL: https://docs.rs-online.com/8a56/0900766b81734963.pdf.