ET0731 IoT Security

Mini Project Report

Group Members:

Class:	DCPE/FT/3A/25	Date:	15/2/2023
Members		Admin No.	
Christine Wong Tien Lee		P2037295	
Darren Tan Kai Jen		P2032980	
Seng Cheong Long Gabriel		P2032357	

Secured Smart Lock

The “SSL”

1. Introduction

Secured Smart Lock is a next generation smart lock which integrates many advanced security features to only authenticate household members to physically access their homes.

2. Product features

The suite of protections featured in Secured Smart Lock are centred around the Category of Technical Reference 64 (TR64). Cryptographic support, Identification and authentication and Data protection are the main categories that we are focusing Secured Smart Lock on.

There is only one method to unlock the Secured Smart Lock, which consists of a series of pre-determined steps that the user must follow. They are easy to execute for the user, but behind-the-scenes, there is much work done to ensure that all data being transmitted is fully secured, and with proper authorisation. Every feature in the Lock's has its own purpose to protect the system from potential attacks, in this section, each feature will be explored and explained further.

2.1. Tamper-proof

Secured Smart Lock (SSL) casing is designed to be tamper-proof, where physical tampering is tedious and attackers attempting to access the ESP32 or locking mechanism will face difficulty. We have built a cardboard prototype which will be shown later in this report, and it is meant to simulate a life-sized door lock.

2.2. 16 Key-Numeric Keypad

To unlock the door, a six-digit numeric code (-XXXXXX-) needs to be keyed into the keypad. If the wrong passcode is entered into the lock three times, the system will reset code for the keypad and restart the system whole process. (Refer to Section 4)

2.3. Android Authentication

Most high security applications like banking apps use the phone's in-built biometric features to authenticate the user of the phone before deploying any type of service. In the Secured Smart Lock, this security implementation is also present. Before establishing any type of connection to unlock the door, our application requests the user for his biometrics data which has already setup in his phone through the settings app prior, which can be in the form of fingerprints, facial recognition, or a PIN password. Without this authentication, the user is unable to access any of the mobile app services.

2.4. Multi-Factor Authentication

Unlocking the door requires the user to meet certain requirements.

1) The user needs to pass the biometrics page before being able to access the app.
2) Once he has successfully gained access, he must connect to the ESP32 module built into the door lock over Bluetooth Low Energy, and then, a randomly generated 6-digit message will be encrypted and sent over automatically.
3) Simultaneously, the mobile app which must be configured to a Thingspeak channel beforehand, will write the same encrypted message to Thingspeak. The un-encrypted 6-digit message will be displayed on the user's phone for 30 seconds.
4) At the ESP32, upon receiving the first message over BLE, it will read the Thingspeak field to retrieve the second message.
5) The ESP32 compares the message received through BLE with the message retrieved from Thingspeak.
6) If they match, the encrypted message will be decrypted into the six-digit code (-XXXXXX-) and it will be used as the passcode for the keypad on the Secured Smart Lock, which the user must key in.

This process provides the system with Multi-Factor Authentication where a BLE message, 6-digit passcode, correct ciphertext in Thingspeak and screen lock security pass are needed to unlock the lock.

2.5. Bluetooth Low Power

Bluetooth Low Power (BLE) is a wireless communication technology used in the Secured Smart Lock system. Its purpose is to connect the Android Phone with the ESP32 within a short distance or Personal Area Network (PAN). BLE will be used to send the encrypted ciphertext containing the 6-Digits Code to the ESP32 while ensuring there is a user nearby the lock to unlock it due to it's short range.

2.6. Advanced Encryption Standard (AES)

In the operation of Secured Smart Lock, the 6-digit passcode will be transmitted into the cloud (Thingspeak). By using Kali Linux and Wireshark, the HTTP packet containing the 6-digit code can be captured and used for man-in-the-middle attacks resulting in the system to be unsecure. By using a symmetric key and an Initialisation Vector, the code can be encrypted to form a ciphertext using 128-bit Advanced Encryption Standard (AES), cipher block chaining mode.

2.7. Inactivity detection

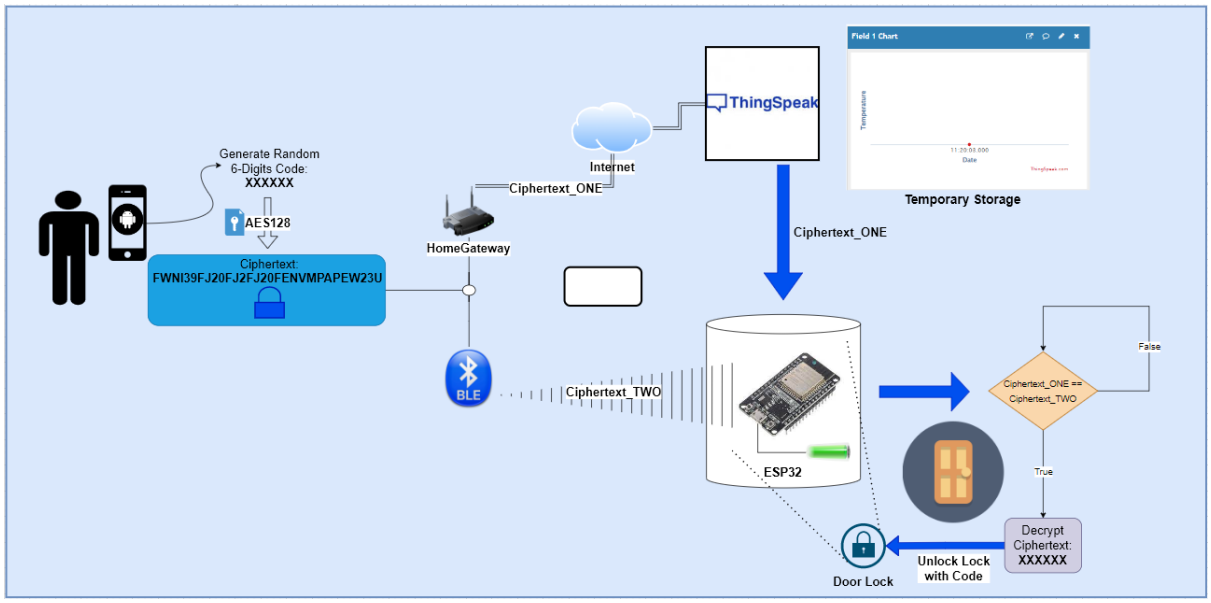
When the app detects a lack of activity by the user (i.e., no touches are made on the screen for approximately 1 to 2 minutes), the app will automatically logout and return back to the page where the user has to reauthenticate himself. This protects against an unauthorised user from comprising the current session.

2.8. Destruction of high-risk data

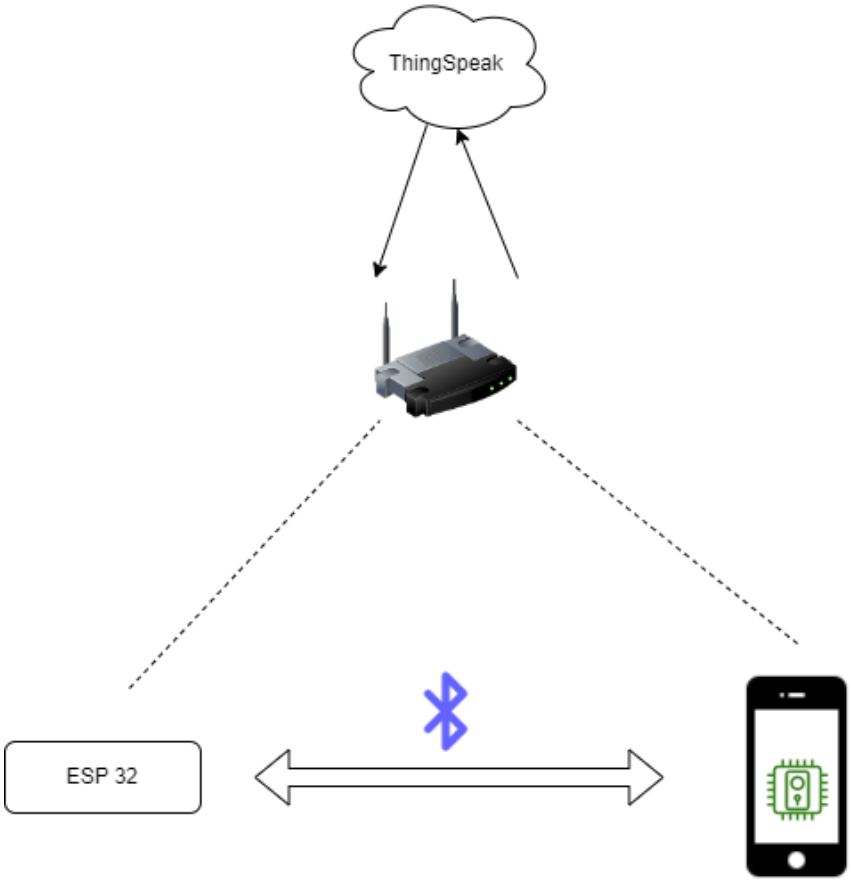
All data that is sent to the Thingspeak cloud will be destructed 10 seconds after generation. Should the data fail to be deleted, the data will be destructed the next time the user re-accesses the Bluetooth unlock page. We aim for there to be absolutely no data stored on the Thingspeak channel at all times, preventing any attackers from retrieving data from the cloud and having sufficient time to figure out the decryption.

3. Diagrams

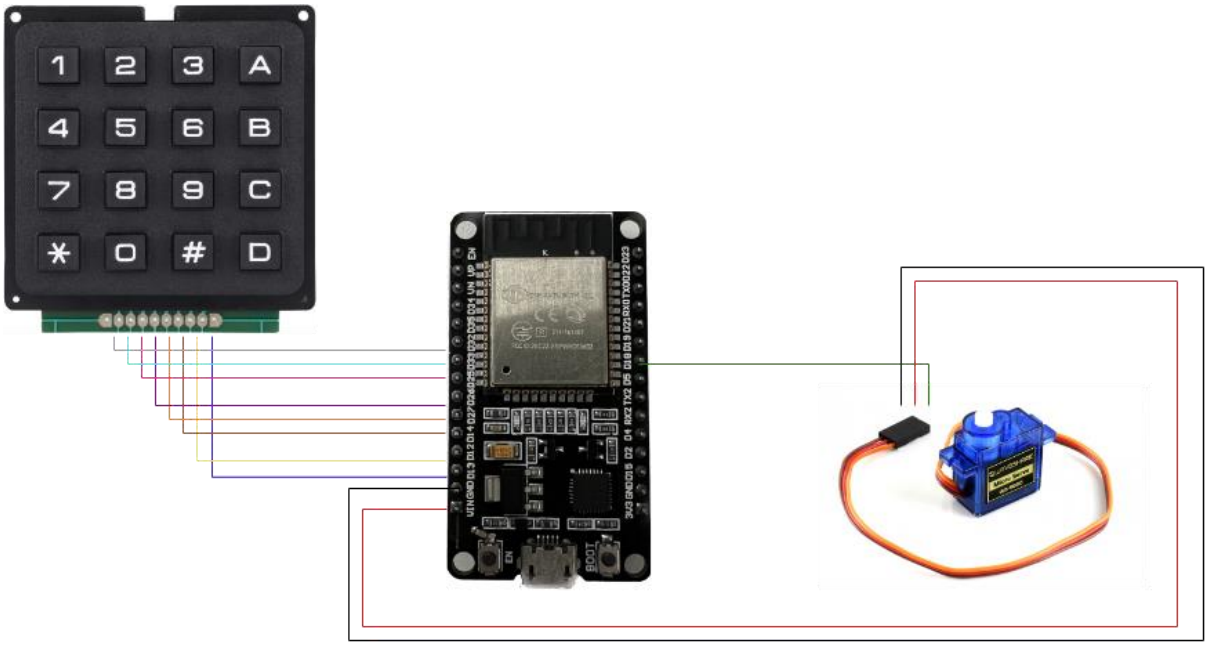
The prototype door lock as part of the Secured Smart Lock



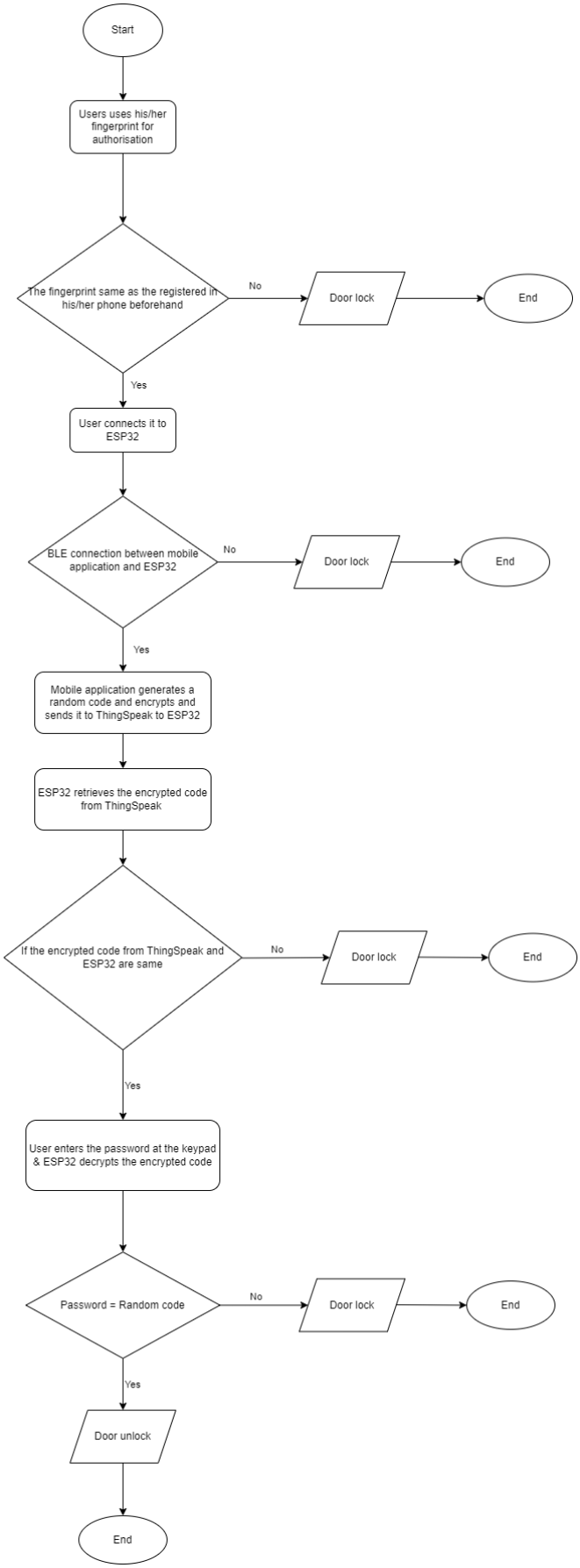
System Diagram



Network Diagram



Schematic Diagram


Start
Users uses his/her fingerprint for authorisation
The fingerprint same as the registered in his/her phone beforehand
No
Door lock
End
Yes
User connects it to ESP32
BLE connection between mobile application and ESP32
No
Door lock
End
Yes
Mobile application generates a random code and encrypts and sends it to ThingSpeak to ESP32
ESP32 retrieves the encrypted code from ThingSpeak
If the encrypted code from ThingSpeak and ESP32 are same
No
Door lock
End
Yes
User enters the password at the keypad & ESP32 decrypts the encrypted code
Password = Random code
No
Door lock
End
Yes
Door unlock
End

Flowchart

4. User Flow

This Section describes the setting up and usage of the Secured Smart Lock.

4.1. Download the Secured Smart Lock (SSL) App

Firstly, the owner must download the Android mobile application for Secured Smart Lock (SSL) to start setting up the SSL. Due to recent changes in Bluetooth permissions required by Google, we have built this app to only work with Android devices running version 12 or later.

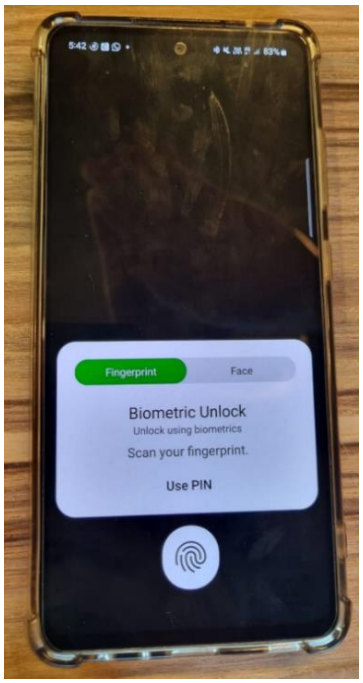
4.2. Set up Secured Smart Lock (SSL)

Before the owner of Secured Smart Lock (SSL) can use the system, they must first key in specific credentials into the application unique to their device. When first using the Secured Smart Lock (SSL), the owner must first authenticate with his biometrics to access the settings page where he can key in his private Thingspeak Write API key. A unique Write API key and Channel ID are provided with each Secured Smart Lock (SSL) device and are censored when keying it into the application, to ensure confidentiality.

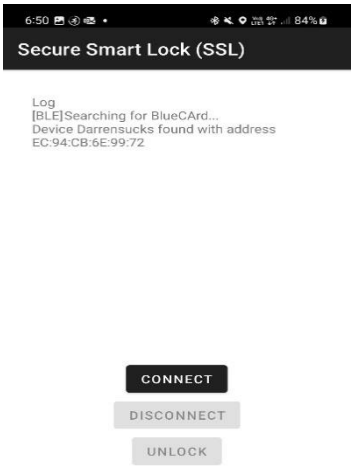
Secondly, the owner needs to provide the Secured Smart Lock with a stable internet connection. This one-time process can be done by connecting the Lock physical interface (micro-USB) to a computer and then enter the owner's Wi-Fi connection credentials. Once this is done, the owner can power the Secured Smart Lock (SSL) using the provided USB cable for usage.

4.3. Usage of Secured Smart Lock (SSL)

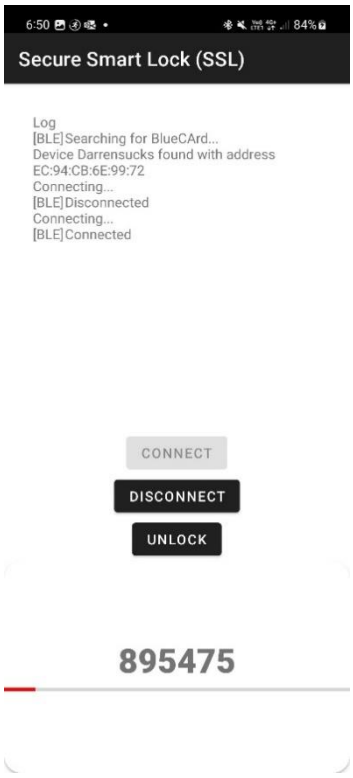
For the usage of Secured Smart Lock (SSL), the Android application is essential to control the lock. If the owner needs to unlock the lock, he must open the application on their phone and unlock the application service by providing their screen lock security credentials. These credentials would be the same used to access their phone.



After authenticating, the owner can navigate to another page that will start scanning for the nearby lock. The application will attempt to connect to the ESP32 inside the Secured Smart Lock (SSL). The BLE's universal unique identifier (UUID) of the owner's lock will appear on the screen if the owner is nearby. The owner can then click "Connect" to establish a BLE connection to the lock.



After successfully establishing the BLE connection, the owner will be given multiple options. First, they can click on the "Disconnect" button to disconnect from the Secured Smart Lock. Secondly, they can click on the "Unlock" button to prompt the application to generate a random 6-digit numeric passcode to be used in the keypad to unlock the Secured Smart Lock (SSL). The 6-digit passcode will be displayed on the application and a timer limit begins counting down for the owner to enter the passcode displayed into the keypad. Once the displayed passcode is entered, the Secured Smart Lock (SSL) will be unlocked.



After the correct code is entered, the Smart Lock automatically disconnects from the owner's phone and relocks the door after the time limit extinguishes.

5. Encrypting the Message

To ensure confidentiality and integrity, the 6-digit passcode is encrypted before getting transmitted out into the cloud or through BLE, preventing any attackers from deciphering any data-in-transit and accessing the passcode for the lock. For the encryption technique, Advanced Encryption Standard (AES) was picked as the most

suitable in our application. AES is a symmetric block cipher encryption method that relies on a single key to encrypt a specific plaintext, which in this case is the 6-digit numeric passcode. By comparing the usage of an Asymmetric encryption technique to AES in the Secured Smart Lock (SSL), AES is more appropriate as there is no need for two keys (public and private) for other third party members to access the passcode. Only the user unlocking the door can access the passcode. Furthermore, RSA is less efficient in terms of encryption/decryption speed, whereas AES is much faster making it more fitting for a door lock.

6. TR64 Compliance

Security Feature	Attack Surface	Checklist	TR 64 Category	Description
Random Code Generation	Physical device	Passcode does not remain the same for each usage	CS-01	Proper random number generation is employed for passcode.
Screen Lock Authorisation	User Android Phone	Authenticate user of phone, preventing impersonation or unauthorized connection by stolen devices.	IA-02 IA-04 DP-01	Phone Screen Lock Authentication is requested before establishing any connection.
AES Encryption	Thingspeak, BLE Connection	Transmitted Passcode is AES encrypted, data in transit is secured with cryptology algorithm.	CS-02 CS-03	Cryptographic Algorithm, AES, is used for code transmission.
Bluetooth Low Power Connection	BLE Connection	Lock only read first received BLE packet then disconnect from client, after which a 10s delay occurs.	RS-03	Does not allow malicious spam BLE packets to transmitted to lock, preventing DDOS attacks.

Three times Passcode Failure	Physical Device	If the wrong passcode is entered into the lock 3 times, the lock disconnects from client and restart process.	AP-01	Existing passcode will be extinguished after repeated incorrect passcode, which will need a new random code generated.
Secured Channel Data	Android	Screen lock security is required to configure the Thingspeak WriteAPIkey and channel ID.	FP-02	Channel Data is censored with "*" and cannot be copied, protecting its integrity.
Tamper-Proof	Physical Device	Casing is designed to be tamper-proof, No exposed joints or physical interface.	AP-04 AP-03 DP-01	Tamper-resistant hardware may be used to prevent physical attacks.
Multi-Factor Authentication	General System	Multi-factor authentication using Thingspeak, BLE, Passcode and Screen Lock Security.	AP-02	Multi-factor authentication is employed for a better overall security.
Threat Modelling	General System	STRIDE and DREAD models are used to evaluate threats.	LP-01 LP-07	Threat modelling is conducted to identify and mitigate threats.

7. Security Testing

7.1. Man-in-the-middle Attack (MITM)

Tools:

- Kali Linux
- Wireshark
- Nmap
- Ettercap

1) Nmap:

“sudo nmap -sn 172.16.110.0/24”

This command is used to scan the network and find the IP addresses of devices connected.

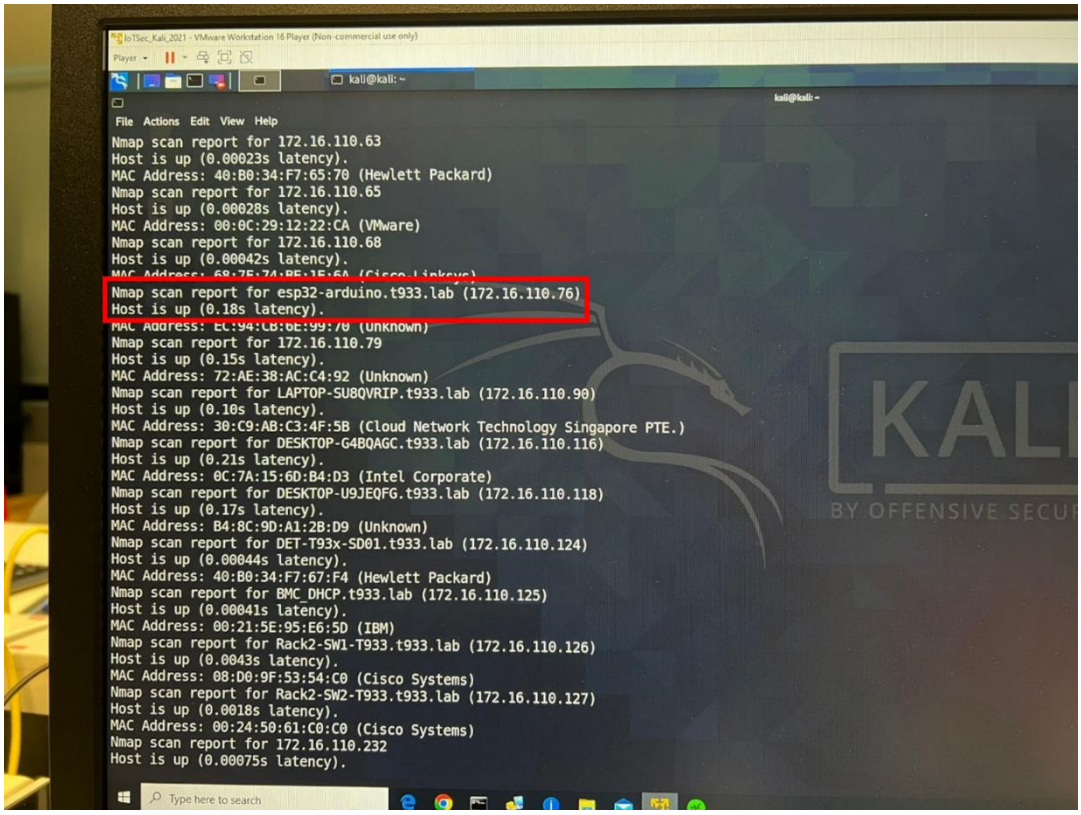
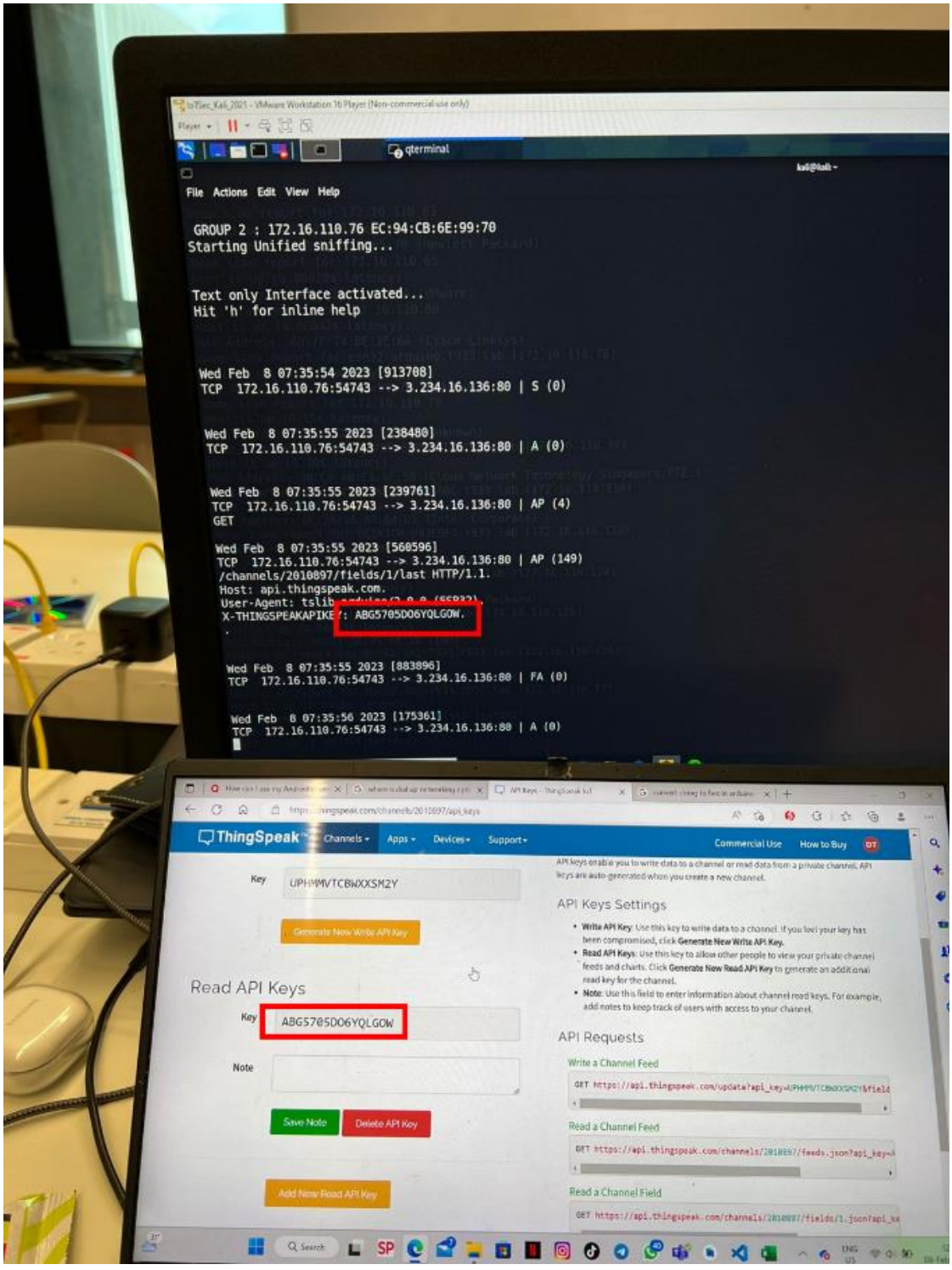


Diagram above shows the IP address of ESP32.

2) Ettercap:

"sudo ettercap -T -S -i wlan0 -M arp:remote /172.16.110.1// /172.16.110.76//"

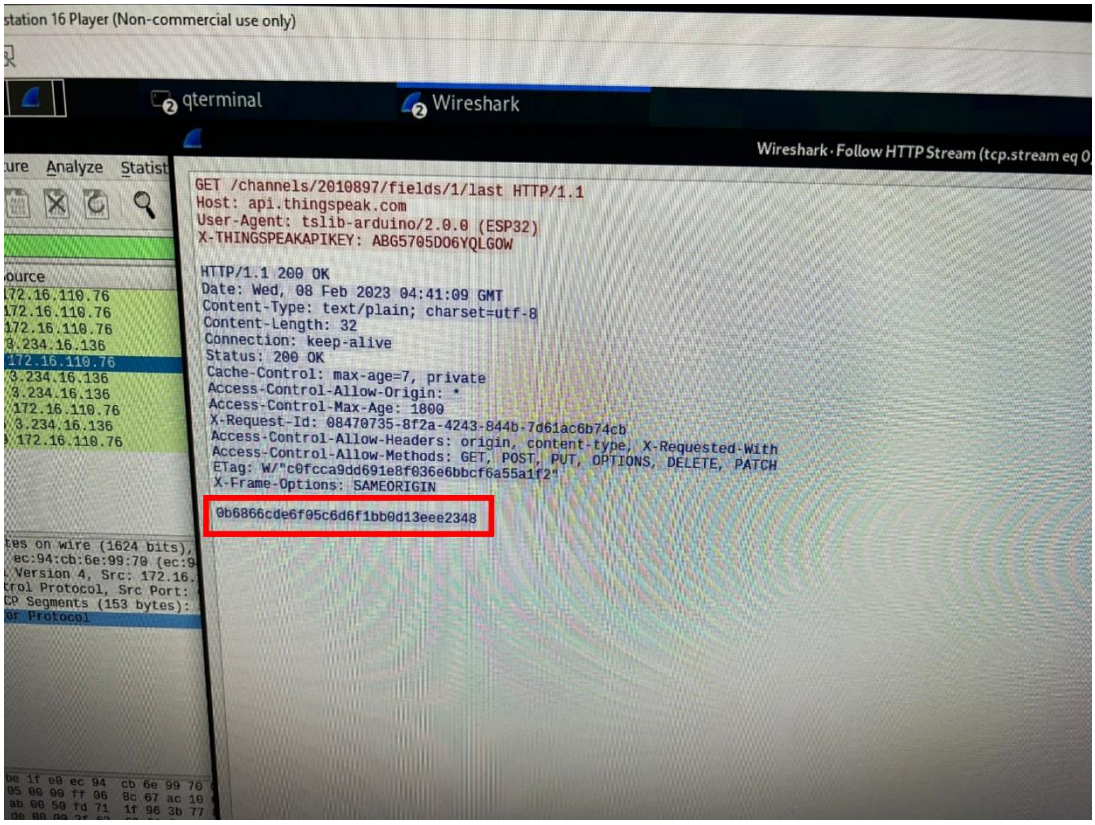
This command is used to sniff the ESP32's traffic.

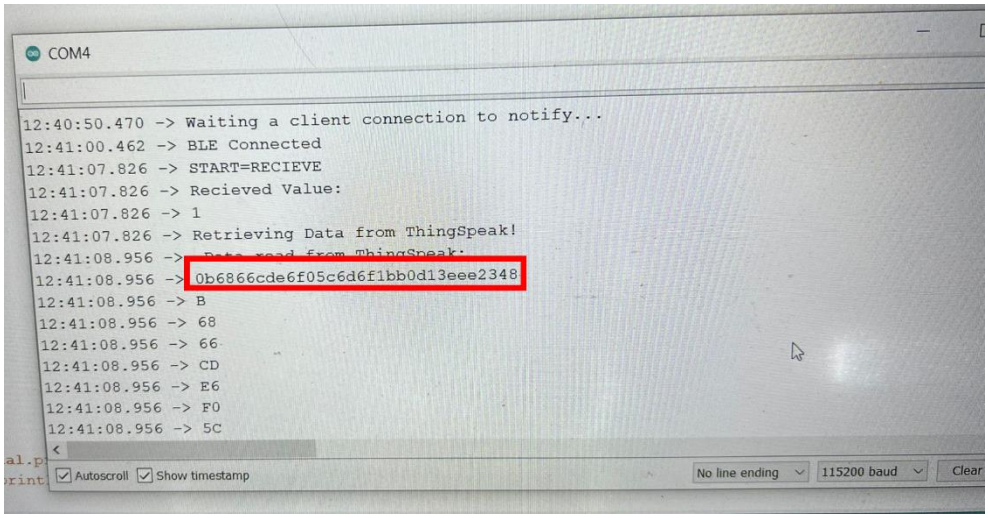


This shows that ettercap is able to retrieve the read API key by sniffing the traffic of ESP32.

3) Wireshark:

“ip.addr == 172.16.110.76 && http”





Unfortunately, the encrypted message can be retrieved by using Wireshark.

8. Threat Modelling

In this Section, we will be using STRIDE and DREAD to identify certain vulnerabilities and mitigate them using different techniques.

8.1. STRIDE

Category	Mitigation
Spoofing	• Mobile App requires screen lock to authenticate actual user of Android Phone before providing services, preventing unauthorised connection with stolen devices or impersonation. • Lock uses home Wi-Fi internet connection to access Thingspeak, without access to home Wi-Fi network credentials, man-in-the-middle attack is difficult. • Data in transit to Thingspeak and through BLE are encrypted, preventing replay attacks.
Tampering	• Physical Interfaces of ESP32 is covered by Tamper-Proof casing. • Lock USB physical connection is only exposed on the other side of the door. • Integrity of messages in transit is protected with AES encryption.
Repudiation	• BLE Connection from Android phone and lock acts as a proximity authentication mechanism to ensure user location is nearby the lock.
Information Disclosure	• Eavesdropping is mitigated with AES encryption and home Wi-Fi network (If attacker does not have Wi-Fi credentials). • Sensitive data such as API keys and encryption keys are not sent out into the network.
Denial of Service	• DDOS attack with BLE to the ESP32 are ineffective as the BLE connection is terminated after every usage or after a timeout period.
Elevation of Privileges	• Sensitive physical interface can only be accessed after unlocking the lock.

8.2. DREAD

Using OWASP scoring system for the DREAD risk assessment model to evaluate this project system.

https://owasp.org/www-community/Threat_Modeling_Process#subjective-model-dread

Damage Potential	**10**
Reproducibility	**0**
Exploitability	**5**
Affected Users	**10**
Discoverability	**5**
Base Score	**6**

9. Potential Improvements

Despite the multiple security implementations in Secured Smart Lock (SSL), there are further improvements that can be added to the lock, to make it more secure, user-friendly and reliable as product. We will be listing some of the potential improvements that Secured Smart Lock (SSL) lack:

Improvements	**Reason**
HTTPS packets accessing Thingspeak	Prevent interception of signal between ESP32 and Thingspeak website.
Provide a user log in credentials in application	Authenticate user account credentials in application to ensure user identity.
Use different IV for AES encryption	Dynamic Initialisation vector for AES CBC mode increases the difficulty for attackers to determine encryption key.
Encrypt sensitive data at rest	Encrypt sensitive data such as API Key and Encryption key prevent physical attack to obtain data.
More Secure locking mechanism	Current prototype lock mechanism can easily be bypassed with a thin object, a more advanced physical lock mechanism would make it more difficult against physical tampering.