PROJECT No. ISNE P706-1/2568

Design and Implementation of a Lightweight Secure Communication System for IoT Using LoRa and MicroPython

Ritthanupahp Sitthananun 650615030
Piyawut Buncharoen 650615024
Siripa Aungwattana 650615032

A Report Submitted in Partial Fulfillment of Project Survey Course
as Required by the Degree of Bachelor of Engineering

Department of Computer Engineering
Faculty of Engineering
Chiang Mai University

2025

Project Title	: Design and Implementation of a Lightweight Secure Communication System for IoT Using LoRa and MicroPython
Name	: Ritthanupahp Sitthananun 650615030
	Piyawut Buncharoen 650615024
	Siripa Aungwattana 650615032
Department	: Computer Engineering
Project Advisor	: Asst. Prof. Kampol Woradit, Ph.D.
Degree	: Bachelor of Engineering
Program	: Information Systems and Network Engineering
Academic Year	: 2025

The Department of Computer Engineering, Faculty of Engineering, Chiang Mai University has approved this project to be part of the degree of Bachelor of Engineering (Information Systems and Network Engineering)

..... Department chair
(Assoc. Prof. Santi Phithakkitnukoon, Ph.D.)

Project examination committee:

..... Main advisor / Chair
(Asst. Prof. Kampol Woradit, Ph.D.)

..... Committee member
(Assoc. Prof. Anya Apavatjrut, Ph.D.)

..... Committee member
(Sasin Janpuangtong, Ph.D.)

Contents

Chapter 1
Introduction

1.1 Project rationale

LoRa (Long Range) is a low-power wireless communication technology widely used in Internet of Things (IoT) applications such as environmental monitoring, infrastructure systems, and smart home appliances. Despite its advantages, LoRa lacks strong built-in security mechanisms. This leaves signals exposed to attacks such as eavesdropping, reply attacks , and message injections [1], [2]. Traditional encryption methods can mitigate these risks but are often too resource-intensive for low-power devices like the ESP32.

1.2 Objectives

To address this challenge, this project avoids static, pre-stored keys and instead generates encryption keys that are dynamic and constantly changing. RSSI (Received Signal Strength Indicator) values are used as the basis for key generation. Since RSSI values are unique between two devices and vary with environmental conditions, the resulting keys are never fixed. Unlike a static key, RSSI-based keys change over time, making them more difficult for an attacker to predict or reuse.

This project presents a lightweight secure communication framework for IoT systems using LoRa and MicroPython, focusing on encryption and dynamic key management. Session keys are updated using RSSI values. Both devices share a predefined keyword and an RSSI interval for adjustment. The communication process begins with the sender transmitting a verification message while the receiver measures the RSSI and iteratively adjusts its interpretation within the agreed interval until the message is correctly decrypted and the keyword matches. This ensures that both devices converge on the same session key despite natural fluctuations in signal strength. Furthermore, by using RSSI values with two-decimal precision and combining measurements across multiple LoRa channels, the system significantly increases resistance to interception and eavesdropping. Overall, the framework provides a practical balance between security and performance for securing IoT communication [3],[4].

1.3 Project scope

1.3.1 Hardware scope

1.3.2 Software scope

1.4 Expected outcomes

1.5 Technology and tools

1.5.1 Hardware technology

1.5.2 Software technology

1.6 Project plan

1.7 Roles and responsibilities

1.8 Impacts of this project on society, health, safety, legal, and cultural issues

Chapter 2
Background Knowledge and Theory

2.1 Internet of Things (IoT)

The Internet of Things (IoT) refers to a network of interconnected devices that can sense, process, and exchange data with minimal human intervention. These devices are widely deployed in applications such as smart homes, healthcare monitoring, environmental sensing, transportation systems, and industrial automation. While IoT enables efficiency and automation, it also introduces security challenges. Many IoT devices are resource-constrained in terms of processing power, memory, and energy supply, making it difficult to implement traditional, computation-heavy cryptographic techniques. As a result, IoT networks are often vulnerable to attacks such as eavesdropping, replay, and message injection.

2.2 LoRa Technology

LoRa (Long Range) is a low-power wide-area network (LPWAN) communication technology that enables devices to transmit data over several kilometers while consuming minimal energy. This makes it suitable for IoT applications where devices must operate on battery power for extended periods. LoRa achieves long-distance communication using Chirp Spread Spectrum (CSS) modulation, which provides robustness against noise and interference. However, LoRa's primary limitation is its lack of built-in security mechanisms. The physical layer itself does not provide strong confidentiality or integrity protection. While LoRaWAN (the higher-layer protocol) introduces some security features, lightweight LoRa implementations—such as those used in simple ESP32 + LoRa projects—are highly vulnerable to interception, spoofing, and key extraction if insecure practices (e.g., hardcoded keys) are used.

2.3 Received Signal Strength Indicator (RSSI)

The Received Signal Strength Indicator (RSSI) measures the power level of a received wireless signal, typically expressed in decibels relative to one milliwatt (dBm). In LoRa systems, RSSI is automatically measured at the receiver whenever a packet is received. Normally, RSSI is used to evaluate link quality or assist in adaptive communication strategies. In the context of security, RSSI can also be leveraged as a source of entropy for key generation. Since RSSI values fluctuate depending on distance, obstacles, interference, and environmental conditions, they are inherently dynamic and difficult for an attacker to predict without being physically co-located in the communication channel. By using RSSI values as a basis for session key generation, IoT devices can avoid reliance on static, pre-shared keys that are easy to compromise.

2.4 Lightweight Cryptography

Lightweight cryptography refers to cryptographic techniques designed specifically for devices with limited computational and memory resources. Unlike traditional algorithms such as AES-256 or RSA, which require significant processing power, lightweight algorithms are optimized for efficiency while maintaining an acceptable level of security. Typical strategies include reducing key sizes, minimizing memory overhead, or designing algorithms tailored for 8-bit or 32-bit microcontrollers. For IoT applications using devices like the ESP32, lightweight cryptography is essential to balance security with system performance. The challenge is to implement schemes that protect data confidentiality and integrity without exceeding constraints such as 10 KB of program memory or 300 bytes of RAM.

2.5 Key Management in IoT

Key management is one of the most critical aspects of securing IoT networks. Traditional approaches rely on pre-shared static keys, which pose significant risks: once an attacker obtains the key, they can decrypt all subsequent communications. Dynamic key management schemes provide stronger protection by regularly updating session keys. One promising approach is to derive session keys from physical-layer metrics, such as RSSI. Since each device independently measures RSSI during communication, keys can be generated locally without transmitting sensitive information over the air. To ensure synchronization, both devices agree on an adjustment interval and use a shared keyword for verification. This method significantly reduces the risk of interception and makes it difficult for attackers to reuse keys, thereby enhancing overall IoT security.

2.5.1 Subsection heading goes here

Subsection 1 text

Subsubsection 1 heading goes here

Subsubsection 1 text

Subsubsection 2 heading goes here

Subsubsection 2 text

2.6 Third section

Section 3 text. The dielectric constant at the air-metal interface determines the resonance shift as absorption or capture occurs is shown in Equation (2.1):

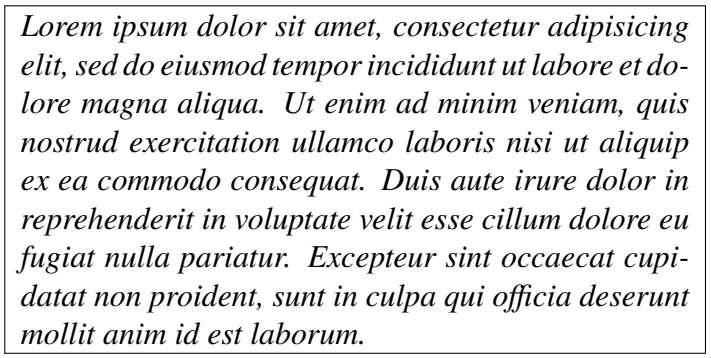

Figure 2.1: This figure is a sample containing lorem ipsum, showing you how you can include figures and glossary in your report. You can specify a shorter caption that will appear in the List of Figures.

$$k_1 = \frac{\omega}{c(1/\varepsilon_m + 1/\varepsilon_i)^{1/2}} = k_2 = \frac{\omega \sin(\theta)\varepsilon_{air}^{1/2}}{c} \tag{2.1}$$

where ω is the frequency of the plasmon, c is the speed of light, ε_m is the dielectric constant of the metal, ε_i is the dielectric constant of neighboring insulator, and ε_{air} is the dielectric constant of air.

2.7 About using figures in your report

Using `\label` and `\ref` commands allows us to refer to figures easily. If we can refer to Figures 3.1 and 2.1 by name in the LaTeX source code, then we will not need to update the code that refers to it even if the placement or ordering of the figures changes.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor

Table 2.1: Sample landscape table

Year	A	B
1989	12	23
1990	4	9
1991	3	6

in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

2.8 Overfull hbox

When the `semifinal` option is passed to the `cpecmu` document class, any line that is longer than the line width, i.e., an overfull hbox, will be highlighted with a black solid rule:

juxtaposition

2.9 ISNE knowledge used, applied, or integrated in this project

2.10 Extracurricular knowledge used, applied, or integrated in this project

Chapter 3
Project Structure

3.1 Methodology

3.1.1 System Architecture

The proposed framework consists of two ESP32 devices, each connected to a LoRa SX1276 module. One device operates as the sender (initiator), and the other as the receiver (responder). The devices communicate over LoRa to exchange verification and application messages. The overall architecture ensures that both devices can generate identical session keys without directly transmitting key material.

3.1.2 RSSI-Based Key Generation

To synchronize RSSI values, one device (the initiator) first transmits a known verification message. The receiving device measures the RSSI of this signal and attempts to decode the message using its reading. If decryption fails, the receiver iteratively adjusts its interpretation by shifting the RSSI value up or down within a pre-agreed tolerance interval (e.g., ±0.5 dBm) until the message is correctly decrypted. Unlike static pre-shared keys, which remain fixed and are vulnerable once compromised, session keys in this framework are dynamically derived from RSSI. Since RSSI is measured uniquely at the receiver and varies naturally with the environment, it is difficult for an attacker to predict or replicate. Security is further enhanced by concatenating the RSSI values and sampling them across multiple frequency channels into an array.

3.1.3 Keyword Verification

To ensure correctness, both devices share a pre-defined keyword. After decryption, the receiver compares the result against this keyword. A match confirms that both devices are synchronized on the same session key. If the keyword is not matched, the process repeats until a valid session key is established.

3.1.4 Implementation Details

The framework is implemented on two ESP32 boards, each connected to LoRa SX1276 modules and powered via USB. Testing is carried out on prototype devices only, within a controlled laboratory environment. No end-user testing or real-world deployment is included. The software stack consists of MicroPython firmware, Visual Studio Code, Thonny IDE, and the mpremote command-line tool. Simulated IoT messages are used as test data to evaluate the system. The variables measured include computation time, CPU usage, and memory consumption.

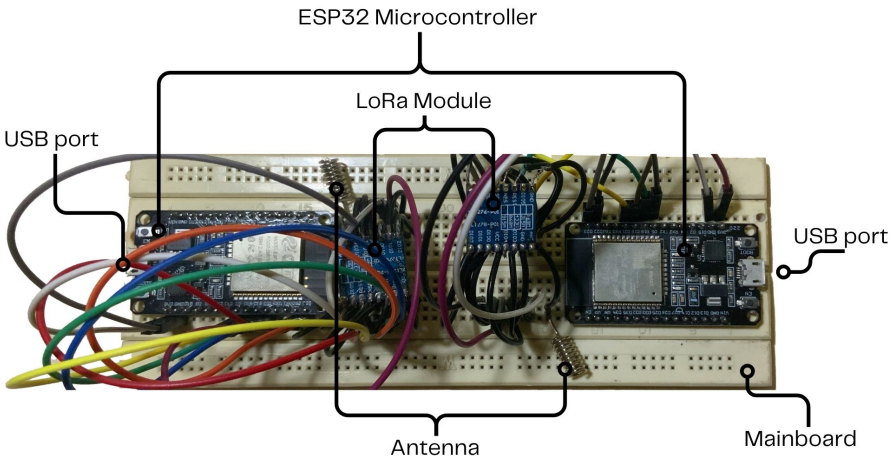

Figure 3.1: Design Architecture

Chapter 4
System Evaluation

4.1 Evaluation Metrics

If implemented, the proposed framework would be evaluated using the following criteria:

4.1.1 Performance

the time required for key generation, encryption, and decryption on ESP32 devices.

4.1.2 Resource Usage

memory and CPU consumption, ensuring the design remains lightweight.

4.1.3 Security Assessment

resistance against common IoT attacks, including replay and man-in-the-middle (MITM) attacks.

4.2 Evaluation Method

4.2.1 Correctness Testing

Repeatedly transmit verification messages and record the percentage of times the receiver derives the correct key and matches the predefined keyword.

4.2.2 Performance Measurement

Measure execution time of key generation and encryption/decryption functions using built-in timers on ESP32.

4.2.3 Resource Usage

Monitor program memory footprint and RAM usage to ensure they remain under the target limits (10 KB flash, 300 bytes RAM).

4.2.4 Security Testing

Simulate potential attack scenarios such as replaying old packets or attempting to intercept communication, to evaluate whether the dynamic key mechanism prevents message reuse or prediction.

4.3 Evaluation Method

The evaluation is expected to show that the RSSI-based key generation approach achieves high correctness in key agreement, maintains lightweight resource usage suitable for ESP32 devices, and provides improved resilience against interception compared to static key methods.

References

[1] Lewis Carroll. *Alice's Adventures in Wonderland*. George MacDonald, 1865.