

**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 Apavatjirut, Ph.D.)

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

## Contents

Contents . . . . .	b
<b>1 Introduction</b>	<b>1</b>
1.1 Project rationale . . . . .	1
1.2 Objectives . . . . .	1
1.3 Project scope . . . . .	1
1.4 Expected outcomes . . . . .	2
1.4.1 Secure IoT Communication Prototype . . . . .	2
1.4.2 Dynamic Key Generation Mechanism . . . . .	2
1.5 Technology and tools . . . . .	2
1.5.1 Hardware technology . . . . .	2
1.5.2 Software technology . . . . .	3
1.6 Project plan . . . . .	3
1.7 Roles and responsibilities . . . . .	3
1.8 Impacts of this project on society, health, safety, legal, and cultural issues .	4
<b>2 Background Knowledge and Theory</b>	<b>5</b>
2.1 Internet of Things (IoT) . . . . .	5
2.2 LoRa Technology . . . . .	5
2.3 Received Signal Strength Indicator (RSSI) . . . . .	5
2.4 Lightweight Cryptography . . . . .	6
2.5 Key Management in IoT . . . . .	6
<b>3 Project Structure</b>	<b>7</b>
3.1 Methodology . . . . .	7
3.1.1 System Architecture . . . . .	7
3.1.2 RSSI-Based Key Generation . . . . .	7
3.1.3 Keyword Verification . . . . .	7
3.1.4 Implementation Details . . . . .	7
<b>4 System Evaluation</b>	<b>9</b>
4.1 Evaluation Metrics . . . . .	9
4.1.1 Performance . . . . .	9
4.1.2 Resource Usage . . . . .	9
4.1.3 Security Assessment . . . . .	9
4.2 Evaluation Method . . . . .	9
4.2.1 Correctness Testing . . . . .	9
4.2.2 Performance Measurement . . . . .	9
4.2.3 Resource Usage . . . . .	9
4.2.4 Security Testing . . . . .	9
4.3 Evaluation Method . . . . .	10
<b>References</b>	<b>11</b>

# 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

The scope of this project focuses on designing and developing a secure communication framework based on LoRa technology using MicroPython for low-power IoT devices such as the ESP32. The project emphasizes generating dynamic encryption keys from signal strength values (RSSI) to enhance communication security, as well as conducting communication experiments between at least two devices.

## **1.4 Expected outcomes**

### **1.4.1 Secure IoT Communication Prototype**

A working prototype of a lightweight secure communication framework implemented on ESP32 boards with LoRa modules, demonstrating reliable data exchange using MicroPython.

### **1.4.2 Dynamic Key Generation Mechanism**

Successful implementation of RSSI-based session key generation, ensuring that encryption keys are dynamic, unpredictable, and resistant to reuse by attackers.

## **1.5 Technology and tools**

### **1.5.1 Hardware technology**

The hardware used in this project consists of the following components:

- **ESP32 Development Board**

- Serves as the main microcontroller for implementing the secure communication framework.
- Provides sufficient processing power, memory, and Wi-Fi/Bluetooth capabilities while maintaining low energy consumption.
- Runs MicroPython firmware for ease of prototyping and testing.

- **LoRa Transceiver Module**

- Enables long-range, low-power wireless communication between IoT devices.
- Used to transmit and receive data packets.

- **Jumper Wires and Breadboard**

- Used to connect ESP32 boards to LoRa modules for prototyping purposes.
- Provides flexibility for testing different circuit configurations in a laboratory environment.

- **Power Supply (USB)**

- Each ESP32 board is powered via USB from a computer or adapter.
- Provides stable voltage and allows simultaneous programming and monitoring.

### 1.5.2 Software technology

- **MicroPython**

- Lightweight Python implementation designed for microcontrollers such as ESP32.
- Provides built-in support for hardware interfaces (GPIO, SPI, UART) and efficient execution on resource-constrained devices.
- Used to implement the encryption, decryption, and RSSI-based key generation mechanisms.

- **Visual Studio Code (VS Code)**

- Serves as the primary code editor and development environment.
- Offers extensions for MicroPython, debugging, and serial communication with ESP32 devices.

- **Thonny IDE**

- Beginner-friendly IDE optimized for MicroPython programming.
- Used for quick prototyping, uploading scripts, and monitoring device outputs.

- **mpremote (MicroPython Remote Tool)**

- Command-line utility for interacting with MicroPython devices over USB.
- Supports file transfer, script execution, and device management.

- **Serial Monitor Tools**

- Used to display and debug communication between ESP32 devices in real time.
- Allows monitoring of RSSI values, encryption/decryption outputs, and keyword verification results.

## 1.6 Project plan

## 1.7 Roles and responsibilities

This project is made possible by 3 students and 1 advisor

- Ritthanupahp Sitthananun: Responsible for coding and demo
- Piyuwut Buncharoen: Responsible for implementing hardware
- Siripa Aungwattana: Responsible for implementing hardware
- Kampol Woradit: Advisor providing suggestions and support.

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

This project focuses on developing secure and reliable IoT communication. Users can transmit and receive data with a reduced risk of leakage or attacks. The system is low-power, minimizing health impacts, and contains no illegal or inappropriate content or functions. Therefore, it does not cause significant adverse effects on society or culture.

## **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.

## **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.,  $\pm 0.5$  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.

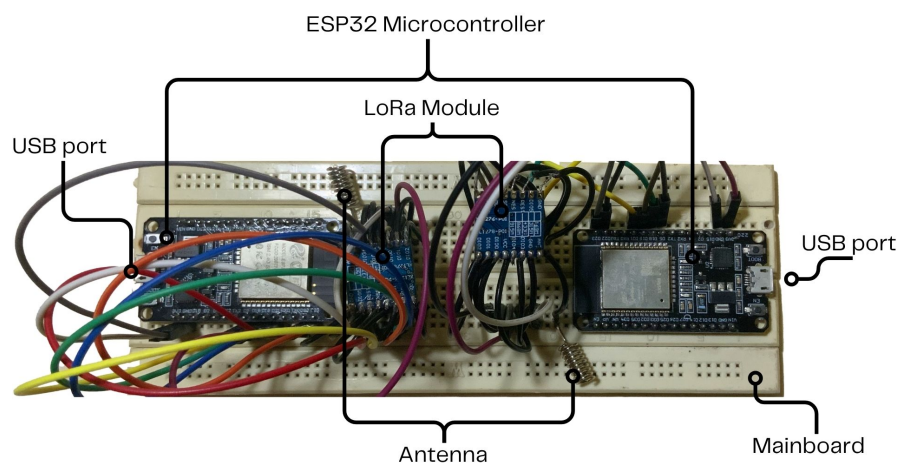


Figure 3.1: Design Architecture

## **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.