Design and Implementation of a Secure BLE Beacon System with Advanced Encryption Using ESP32

A. R. Zakaria Talukdar
Computer Science and Engineering
Assam University, Silchar
Contacts:
E-mail: zakariatalukdar123@gmail.com
GitHub: https://github.com/arrhenius975

January 28, 2025

Contents

1 Introduction

This document provides a comprehensive overview of a secure BLE beacon implementation using the ESP32-WROOM-32 module. The system features advanced encryption, persistent storage, and configurable parameters, making it suitable for secure location-based services and asset tracking applications.

2 Hardware Overview

2.1 Key Components

- **Processor**: ESP32 with dual-core Xtensa LX6 processor
- **Memory**:
  - SRAM: 520 KB
  - Flash: External flash support
  - NVS: Non-volatile storage for configuration
- **Wireless**: Bluetooth 4.2 (BLE) support
- **Security**: Hardware encryption acceleration

3 Software Architecture

3.1 Core Components

- **Beacon Manager**:
  - UUID, Major, Minor value management
  - Advertisement interval control
  - TX power configuration
- **Security Module**:
  - AES-256 encryption
  - Secure key generation
  - Protected storage
- **Storage Manager**:
  - NVS flash management
  - Configuration persistence
  - Key storage

3.2 Key Features

- Configurable beacon parameters
- Encrypted payload transmission
- Persistent configuration storage
- Comprehensive logging system
- Power-efficient operation

4 Implementation Details

4.1 Beacon Configuration

Listing 1: Beacon Configuration Structure

```c
typedef struct {
    uint8_t uuid[16];
    uint16_t major;
    uint16_t minor;
    int8_t power;
    uint16_t adv_int_min;
    uint16_t adv_int_max;
    bool encryption_enabled;
} beacon_config_t;
```

4.2 Encryption Implementation

Listing 2: Encryption Function

```c
esp_err_t beacon_crypto_encrypt(
    const uint8_t *data,
    size_t len,
    uint8_t *out_data,
    size_t *out_len
) {
    // AES-256 encryption implementation
    mbedtls_aes_setkey_enc(&aes_ctx,
                           current_key.key, 256);
    // ... encryption logic
}
```

4.3 Storage Management

Listing 3: Configuration Storage

```c
esp_err_t beacon_storage_save_config(
    const beacon_config_t *config
) {
    // Save configuration to NVS
    nvs_set_blob(storage_handle,
                 NVS_KEY_UUID,
                 config->uuid,
                 sizeof(config->uuid));
    // ... storage logic
}
```

5 Build and Deployment

5.1 Prerequisites

- ESP-IDF framework installed
- CMake build system
- ESP32 development board
- USB cable for programming

5.2 Build Instructions

1. Set up ESP-IDF environment:

   ```
   . $IDF_PATH/export.sh   # Linux/macOS
   %IDF_PATH%\export.bat   # Windows
   ```

2. Configure the project:

   ```
   idf.py menuconfig
   ```

3. Build the project:

   ```
   idf.py build
   ```

4. Flash to ESP32:

   ```
   idf.py -p (PORT) flash
   ```

6 Testing and Verification

6.1 Test Cases

- **Encryption Tests**:
  - Key generation
  - Encryption/decryption
  - Key storage security

- **Beacon Tests**:
  - Advertisement intervals
  - Signal strength
  - Battery efficiency
- **Storage Tests**:
  - Configuration persistence
  - NVS reliability
  - Error handling

6.2 Verification Tools

- nRF Connect for Mobile
- LightBlue Explorer
- ESP-IDF Monitor

7 Power Management

7.1 Power Optimization

- BLE-only mode
- Configurable TX power
- Optimized advertising intervals
- Sleep mode support

8 Security Considerations

8.1 Security Features

- AES-256 encryption
- Secure key storage
- Protected configuration
- Regular key rotation

8.2 Security Recommendations

- Regular firmware updates
- Secure key management
- Physical access protection
- Monitoring for unauthorized access

9 Future Enhancements

- Over-the-Air (OTA) updates
- Enhanced encryption schemes
- Battery monitoring
- Remote configuration interface
- Integration with asset tracking systems

10 Conclusion

The implemented secure BLE beacon system provides a robust foundation for building secure location-based services. With its advanced encryption, configurable parameters, and efficient power management, it meets the requirements for both security and functionality in modern IoT applications.