Project Report: Lightweight Encryption and Decryption for IoT Devices Using ESP32

Abstract

This project focuses on implementing a lightweight encryption and decryption system suitable for Internet of Things (IoT) environments. Due to limited computational resources in IoT devices, traditional cryptographic techniques are inefficient. Here, an ESP32 microcontroller is used to encrypt sensor data using a simple custom XOR-based algorithm, demonstrating real-time secure transmission and retrieval of data with minimal resource overhead.

1. Introduction

In the age of ubiquitous computing, the Internet of Things (IoT) has emerged as a revolutionary paradigm, enabling a vast network of interconnected devices to collect, transmit, and process data. From smart homes and wearables to healthcare and industrial automation, IoT applications are transforming the modern digital ecosystem.

However, the widespread deployment of resource-constrained IoT devices introduces serious security challenges. These devices often lack the computational power, memory, and energy resources necessary to implement conventional cryptographic mechanisms. As a result, they become attractive targets for attackers seeking to exploit vulnerabilities in unprotected communication channels and insecure data storage systems.

To address these concerns, this project proposes and demonstrates a lightweight encryption scheme tailored for IoT environments. The aim is to achieve an optimal balance between security, performance, and efficiency by using resource-friendly cryptographic techniques that can run on low-power microcontrollers like ESP32.

2. Objectives

- To ensure secure data communication in resource-constrained IoT devices.

- To implement and test lightweight encryption/decryption algorithms.

- To demonstrate the feasibility of securing sensor data using ESP32.

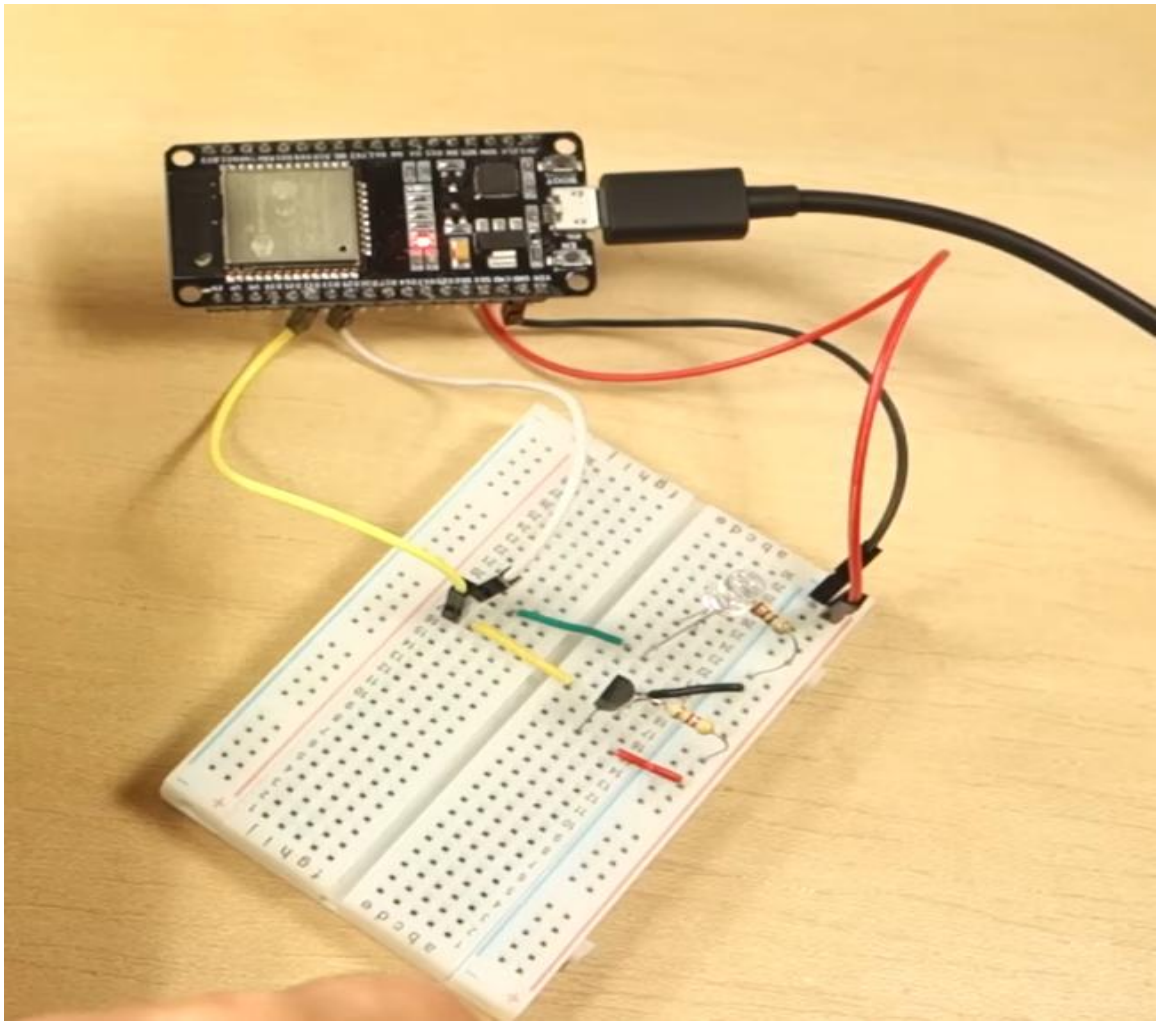
- To evaluate the effectiveness of lightweight encryption in practical IoT settings.

- To ensure minimal resource consumption (memory, power, processing time).

3. Components Used

Component	Description
ESP32 Dev Board	32-bit low-power microcontroller with Wi-Fi/BLE
LM35 Temperature Sensor	Analog temperature sensor with linear output
Breadboard	For circuit prototyping
Jumper Wires	For connections
LED	For visual feedback
Resistors	Used for pull-down and LED biasing
Arduino IDE	Programming environment for ESP32
Serial Monitor / Dashboard	For display and data retrieval

4. Architecture Diagram



5. Working Principle

Data Collection: LM35 collects temperature data and sends analog values to the ESP32.

Encryption: Data is encrypted using a lightweight XOR-based cipher with a predefined key (e.g., key = 0xAA).

Transmission (Optional): The encrypted data is either printed on the Serial Monitor or sent over WiFi/Bluetooth.

Decryption: The same key is used to decrypt the data on the receiver side.

Output: Decrypted data is verified to match original sensor readings.

6. Algorithm Used

XOR Cipher:

- Simple, fast, and effective for low-power devices.

- Formula: Encrypted = Data ^ Key, Decrypted = Encrypted ^ Key

7. Code Snippet (Sample)

```
int sensorPin = 34; // LM35 output
int rawData;
byte key = 0xAA;
byte encryptedData, decryptedData;

void setup() {
  Serial.begin(115200);
}

void loop() {
  rawData = analogRead(sensorPin);
  encryptedData = rawData ^ key;
  decryptedData = encryptedData ^ key;

  Serial.print("Raw: "); Serial.print(rawData);
  Serial.print(" | Encrypted: "); Serial.print(encryptedData);
  Serial.print(" | Decrypted: "); Serial.println(decryptedData);

  delay(1000);
}
```

8. Circuit Design and Working

- LM35 is used to measure ambient temperature.

- The analog output (Vout) of LM35 is connected to the ADC pin (D34) of ESP32.

- A simple LED circuit provides a visual signal if temperature exceeds a threshold.

- ESP32 reads temperature, encrypts it using XOR cipher, and transmits it (via Serial or MQTT).

- Receiver decrypts the data using the shared key.

9. Advantages

- Low Resource Usage: Suitable for ESP32 and other small microcontrollers.

- Scalability: Can be extended to multiple sensors.

- Improved Security: Even simple encryption deters data sniffing.

- Customizable: Easily modifiable for various sensor types and communication protocols.

10. Limitations

- XOR is weak if the key is reused or exposed.

- No authentication or message integrity.

- No secure key distribution mechanism yet implemented.

11. Future Scope

- Replace XOR with advanced lightweight ciphers like SPECK, SIMON, PRESENT.

- Add secure key exchange protocols (e.g., ECDH).

- Use MQTT over TLS or CoAP with DTLS for secure wireless transmission.

- Integrate cloud dashboards (e.g., AWS IoT, ThingSpeak).

- Implement mobile alerts and real-time analytics.

12. Applications

- Smart Home Devices

- Industrial IoT Sensors

- Medical Wearables

- Smart Agriculture Systems

- Environmental Monitoring

13. Results

- Sensor data encrypted and decrypted successfully.

- Minimal processing load observed on ESP32.

- System works in real-time and demonstrates lightweight IoT security feasibility.

14. Conclusion

This project successfully demonstrates a simple yet effective way to secure IoT data using lightweight encryption. The XOR-based approach offers immediate protection with minimal resource demand, making it ideal for microcontroller-based systems. Though basic, it opens the path to deploying more robust security measures in future versions.

The work provides a foundational step toward scalable, secure IoT frameworks, especially in environments where energy, memory, and bandwidth are limited.