ECE40862: Software for Embedded Systems

Fall 2019

Lab 6 – Secure Communication between 2 ESP32-Sensor FeatherWing Assembly

To be done IN GROUPS OF 2; Due by 11:59pm, Wednesday, October 30, 2019.

1. Overview

In this assignment, you will be working in Teams of TWO. Your first task is to find a suitable partner to work with. This assignment deals with *MQTT communication protocol*, *Encryption and Decryption using AES* and *Authentication using HMAC SHA*. These protocols and standards are used in most Embedded and IoT industries for secure data communication.

Both team members should use the "spinner" i.e., the *ESP32 Feather board* and *Adafruit Sensor FeatherWing* assembly, you created in Lab5 and exchange data in a **secure/encrypted** method using MQTT communication protocol. Fig. 1 shows a high-level overview of the application the team should build.

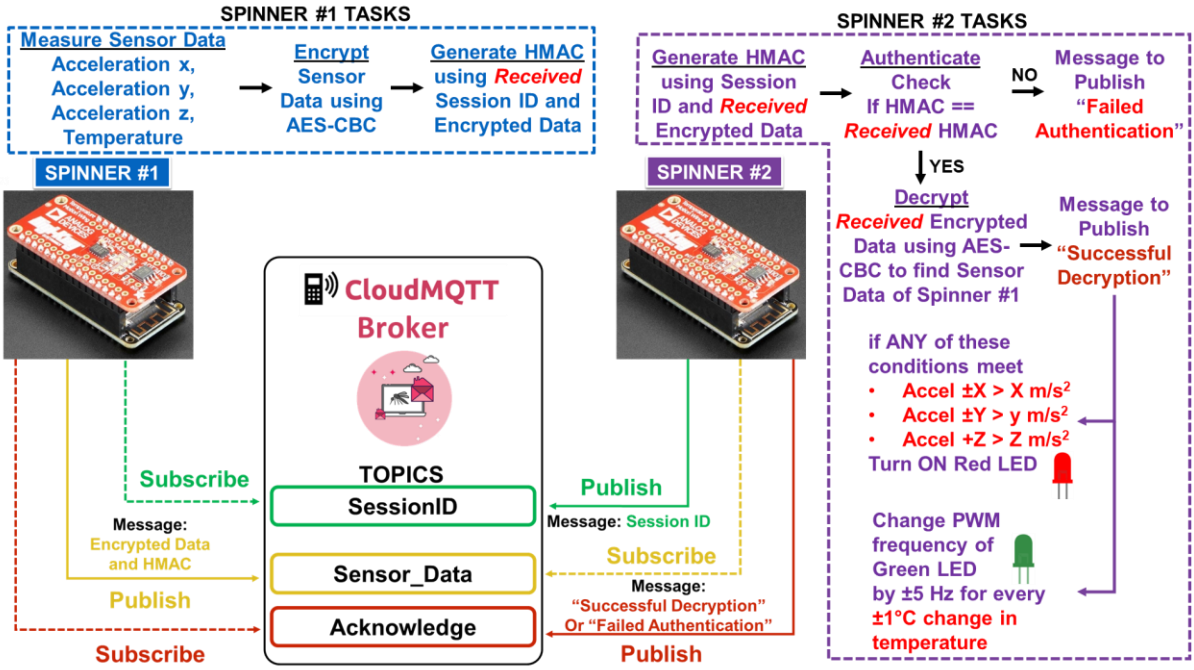

Figure 1. High level Overview of Application

MQTT stands for **M**essage **Q**ueuing **T**elemetry **T**ransport. It is a lightweight **publish** and **subscribe** system where you can publish and receive messages as a client. In this assignment, you will be configuring your spinners as two MQTT Clients. Each of them will publish messages on some topics while subscribe to other topics to receive messages. To enable this two-way communication, you will be using the CloudMQTT service, which hosts Mosquitto servers/brokers in the cloud. The broker is primarily responsible for **receiving** all messages from the publishing clients, **filtering** the messages, **decide** who is interested in them and then **publishing** the message to all subscribed clients.

1.1. Overall Application Workflow (Fig. 1)

The flow of tasks executed by spinner #1 and spinner #2 are described. Your application should follow this for the **Secure Spinner Demo** after initialization and calibration is completed.

1. Spinner #2 publishes message on the ***SessionID*** topic. Generate a random number every 1 second and publish the generated number on this topic every 1 second. We will call this random number as '**SessionID**'. So, every 1 second, a new SessionID is created.
2. Spinner #1 is subscribed to the ***SessionID*** topic. When it receives the *SessionID* from spinner #2, it starts the following series of tasks. After their completion, spinner #1 waits until it receives any message on the Acknowledgement topic.
    a. *Measure* acceleration in X, Y, Z axis from the ADXL343 accelerometer sensor and temperature from the ADT7410 temperature sensor.
    b. Use AES-CBC (Cipher Block Chaining) method to *encrypt* the sensor data.
    c. *Generate HMAC* (hash-based message authentication code) using the encrypted data and received SessionID.
3. Spinner #1 publishes message on the ***Sensor_Data*** topic. Create your message or payload by using the encrypted data and generated hmac and convert to JSON format and publish as soon as the previous step is completed.
4. Spinner #2 is subscribed to the ***Sensor_Data*** topic. When it receives any message from spinner #1, it starts the following series of tasks:
    a. *Generate HMAC* using the received encrypted data and original SessionID at this time step it had generated in step 1.
    b. Check if the generated HMAC matches the one it has received as a part of the message to *verify authenticity of sender*. If there is a mismatch, spinner #2 publishes message "Failed Authentication" on another new topic, *Acknowledgement* and goes back to step 1.
    c. On the other hand, if the HMACs match, then use AES-CBC to decrypt the received encrypted data to figure out sensor readings of spinner #1. Publish message "Successful Decryption" on the topic, *Acknowledgement*. After completion of the following two tasks, spinner #2 goes back to step 1.
        i. Check the decrypted sensor readings and **turn on external red LED** to observe if ANY of these conditions meet
            1. Accel ±X > X m/s^2
            2. Accel ±Y > y m/s^2
            3. Accel +Z > Z m/s^2

ii. Additionally, **change the PWM frequency of Green LED** by ±5 Hz for every ±1°C change in temperature.

5. Spinner #1 is subscribed to the *Acknowledgement* topic. Once it receives any message in this topic, it waits to receive message on SessionID topic.

NOTE 1: In Fig. 1, and the overall application workflow described above, **encrypted data** refers to combination of **encrypted nodeid** (unique ESP32 id), **encrypted initialization vector** (random number generated by Spinner #1 for each cycle of *measure-encrypt-generate HMAC*), and **encrypted sensor data** (Accel X, Y, Z, Temperature). Refer to the provided *crypt.py* file.

NOTE 2: You need to move the ESP32 and FeatherWing assembly by hand to mimic the motion of the spinner. You are free to choose the values for X, Y, Z m/s^2 as the maximum acceleration in 3 axes. However, make sure that the values chosen should distinguish the given states (e.g. the difference of acceleration below +X m/s^2 and acceleration above +X m/s^2 should be clearly visible to the observer's eyes. To increase temperature on the sensor, you can physically place your finger over the sensor chip.

2. Programming Exercises

2.1. Hardware Interfacing

Interface the following components to the ESP32 board:

- One external Red **LED** as GPIO OUTPUT.
- One external Green **LED** as PWM OUTPUT.
- Two external **push buttons** as GPIO (DIGITAL) INPUTs.

NOTE: The FeatherWing already has pull up resistors attached to the SCL and SDA pins. So, you do not need to attach separate resistors. This lab assumes you have successfully interfaced the featherwing with the ESP32 in Lab5. In case you were not able to read data from the featherwing successfully earlier, make sure to do that before progressing ahead.

2.2. Software Implementation

2.2.1. Setup MQTT on ESP32 and Cloud

2.2.1.1. Download MQTT and HMAC MicroPython Library

Connect the ESP32 to the Internet (same as Lab 3 Section 2.2.1) and stay connected. Use **upip** package manager to install the packages '*micropython-umqtt.simple*' and then '*micropython-umqtt.robust*' on your ESP32 board. umqtt is a simple MQTT client for MicroPython. It consists of two submodules: **umqtt.simple** and **umqtt.robust**. umqtt.robust is built on top of umqtt.simple and adds auto-reconnect facilities for some of networking errors. Check the following GitHub links for more details. You will be using the MQTT client class for any communication using the MQTT protocol. Additionally, install '*micropython-hmac*' for authentication.

- https://github.com/micropython/micropython-lib/tree/master/umqtt.simple
- https://github.com/micropython/micropython-lib/tree/master/umqtt.robust
- https://github.com/micropython/micropython-lib/tree/master/hmac

2.2.1.2. Setup Account on CloudMQTT as MQTT Broker

CloudMQTT is a globally distributed MQTT broker which will allow you to achieve the two-way communication between the two ESP32. You can follow this link to create the "Cute Cat" plan: https://medium.com/@suci/build-your-mqtt-broker-in-cloudmqtt-f95ba04270c3.

2.2.2. Initialization of Sensors (*almost same as Lab5*)

Use the I^2C driver in MicroPython (**machine.I2C**) to communicate with both of the sensors by constructing I^2C bus on your ESP32. Perform the following operations:

- Initialize LEDs, Switches, Interrupts, Timers (if you need any)
- Use the schematics and ESP32 manuals to create the I^2C bus using proper pins.
- Set the I^2C communication speed to 400 kHz.
- Detect switch presses using **interrupts**.
    - If you press SWITCH1, use **onboard LED** (NOT external LEDs) as GPIO output and turn it ON and implement the actions outlined in section 2.2.3 (*Interfacing Sensors*). LED should stay ON unless you press SWITCH2.
    - If you press SWITCH2, turn off **the onboard LED**, and implement the actions outlined in section 2.2.4 (Secure Spinner Demo).

2.2.3. Interfacing Sensors

Interface the sensors ADXL343 and ADT7410 with your ESP32 board and calibrate them.

2.2.3.1. Initialize Accelerometer (*almost same as Lab5*)

Start the accelerometer initialization process by checking the device ID (*find out from the corresponding datasheet*). If the device ID is inaccessible or incorrect, your program should return an error message. Upon successful checking, configure the following settings in the ADXL343 using I^2C. *Your program should display adequate messages on the terminal after completion of all the initialization steps.*

- Set to **10-bit full-resolution mode** for output data (X-, Y-, and Z-axis)
- Set range to **±2g**
- Set output data rate (ODR) to **800 Hz**

NOTE: You can also use higher range and resolution but take necessary precautions to read data properly.

2.2.3.2. Calibrate Accelerometer (*same as Lab5*)

The accelerometer data needs to be calibrated before use. For instance, if your ESP32 and Featherwing assembly *remains flat and stands still*, output data for X and Y should be **0 m/s^2** or **0 g** and for Z should be **9.8 m/s^2** or **1 g** which is the default acceleration of gravity. Check

datasheet for more details on **offset calibration**. *Your program should display appropriate message on the terminal after completion of the calibration.*

2.2.3.3. Initialize Temperature Sensor (*same as Lab5*)

Start the temperature sensor initialization process by checking the device ID (*find out from the corresponding datasheet*). If the device ID is inaccessible or incorrect, your program should return an error message. Upon successful checking, configure the following settings in the ADT7410 using memory/register operations using I^2C. *Your program should display adequate message on the terminal after completion of the initializations.*

- Set to **16-bit high resolution** for temperature data

2.2.4. Secure Spinner Demo

- Setup the individual Spinner#1 and Spinner#2 so that they are able to publish and subscribe to different topics using MQTT as described in overall application workflow.
- When required *as indicated in overall workflow*, spinner #1 should read accelerometer and temperature sensor data from the featherwing using I^2C protocol. YOU DO NOT NEED TO CALCULATE VELOCITY OR TILT like Lab5. Measure only raw accelerometer data in all 3 different axes of motion: X, Y, Z.
- You will be provided with a skeleton python file: **crypt.py** for **encryption** and **decryption.** Analyze the file to figure out the steps needed to perform encryption and decryption of any data. You will observe that the file has a single class 'CryptAes' and different functions each with detailed comments. You should use appropriate functions from this class to **encrypt sensor data, generate hmac** in Spinner #1, and **verify hmac, decrypt sensor data** in Spinner #2.

2.2.5. Complete System

Fig. 2 explains the state transition diagram you need to follow for this assignment. Start your program with the initialization step (2.2.2). This section needs to run only once. Afterwards, use switch1 and switch2 presses to change your state between interfacing sensors to secure spinner demo. It is obvious that you need to interface your sensors before demo for the first time. Ideally, you only would need to interface sensors once and then stay in the spinner demo.

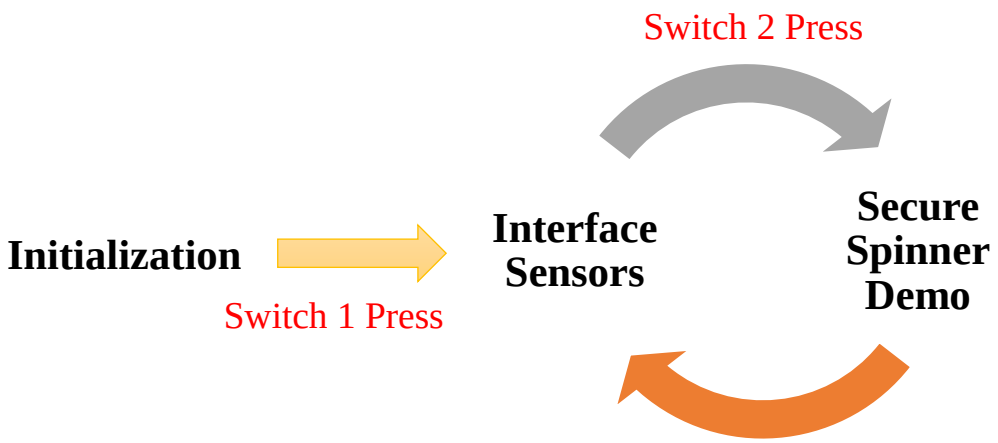

Figure 2. State Transition Diagram of Secure Spinner using ESP32 and Sensor FeatherWing

3. Submission

Make sure you follow these instructions precisely. Points will be deducted for any deviations. You need to turn in your code on Blackboard. Please create a directory named ***username*_lab6**, where username is your CAREER account login ID. *This directory should contain only the following files, i.e., no executables, no temporary files, etc.*

1. ***spinner1.py***: your program for Spinner#1.
2. ***spinner2.py***: your program for Spinner#2.
3. ***crypt.py***: completed encryption file.

Zip the files and name it as *username*_lab6.zip and **upload the .zip file to Blackboard**.

4. Grading

20 teams (40 students) will be randomly selected for lab demo, they will be informed on Thursday, 31st October, 10 AM *via your Purdue email id*. If you are selected for the demonstration, you will need to login to any of the computers in the **EE217 lab**, download your submitted source codes from Blackboard, copy them to your ESP32 board *using rshell* and show the TA that your code works correctly. You will also be asked to show the TA your code listing and answer conceptual questions about your code. Marks are only awarded if you satisfy all requirements.

In addition, automatic evaluation scripts will be executed on your uploaded source codes on Blackboard and you will be graded according to the evaluation results.

NOTE: Follow the lab document strictly when using different peripherals/modules/packages. Points will be deducted if you fail to follow the lab instructions. If anything is NOT mentioned explicitly, you can use package/module to write your program.

REFERENCES

[1] Getting started with MicroPython on the ESP32
https://docs.micropython.org/en/latest/esp32/tutorial/intro.html

[2] ESP32 WROOM-32 Datasheet
https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[3] ESP32 Technical Reference Manual
https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[4] Adafruit HUZZAH32 – ESP32 Feather Online Manual
https://learn.adafruit.com/adafruit-huzzah32-esp32-feather

[5] Adafruit ESP32 Feather Schematics https://cdn-learn.adafruit.com/assets/assets/000/041/630/original/feather_schem.png?1494449413

[6] MicroPython GitHub https://github.com/micropython/micropython

[7] ESP32 specific functionalities in MicroPython http://docs.micropython.org/en/latest/library/esp32.html

[8] Learn how to talk to I^2C devices with MicroPython: https://learn.adafruit.com/micropython-hardware-i2c-devices/i2c-master

[9] ADXL343 triple-axis Accelerometer: https://www.adafruit.com/product/4097

[10] ADXL343 datasheet: https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL343.pdf

[11] ADT7410 precision Temperature Sensor: https://www.adafruit.com/product/4089

[12] ADT7410 datasheet: https://www.analog.com/media/en/technical-documentation/data-sheets/adt7410.pdf

[13] Cloud MQTT: https://www.cloudmqtt.com/

[14] AES Wiki: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[15] More on AES encryption: https://www.comparitech.com/blog/information-security/what-is-aes-encryption/

[16] HMAC Wiki: https://en.wikipedia.org/wiki/HMAC

[17] More on HMAC: https://www.jscape.com/blog/what-is-hmac-and-how-does-it-secure-file-transfers