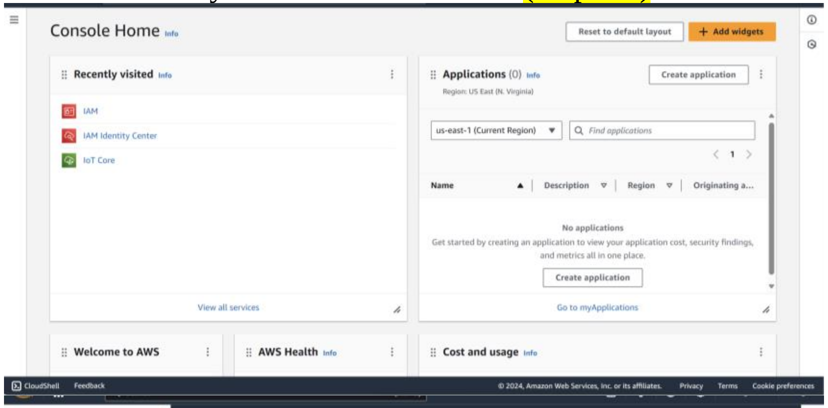
IoT Security and Privacy

Lab 5 – Network Security on ESP32

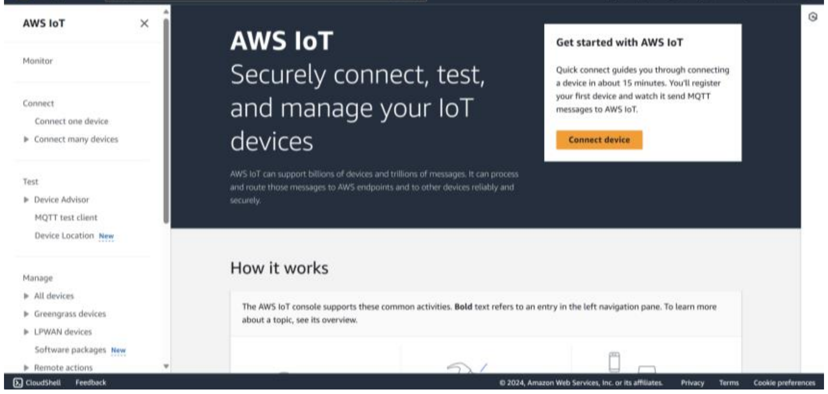
Questions

1. Please visit [Set up your AWS account] to create your AWS account. Please visit [AWS account root user credentials and IAM user credentials] to see the difference between root user and IAM user. The root user can be used for this assignment to log into AWS console. Please provide a screenshot of your AWS IoT console. (10 point)



Get started by creating an application to view your application cost, security findings, and metrics all in one place.

© 2024, Amazon Web Services, Inc. or its affiliates.



AWS IoT

Securely connect, test, and manage your IoT devices

AWS IoT can support billions of devices and trillions of messages. It can process and route those messages to AWS endpoints and to other devices reliably and securely.

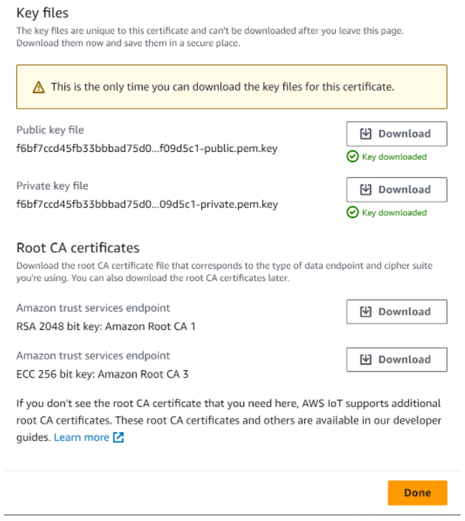
Quick connect guides you through connecting a device in about 15 minutes. You'll register your first device and watch it send MQTT messages to AWS IoT.

The AWS IoT console supports these common activities. **Bold** text refers to an entry in the left navigation pane. To learn more about a topic, see its overview.

© 2024, Amazon Web Services, Inc. or its affiliates.

2. Please visit [Create AWS IoT resources] and follow steps in this article.
   a. *Create a Thing object*. This Thing should be called ($YOUR_GROUPNUMBER)_ESP. (10 point)

i. Be sure to download the keys and certificate.



Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

⚠ This is the only time you can download the key files for this certificate.

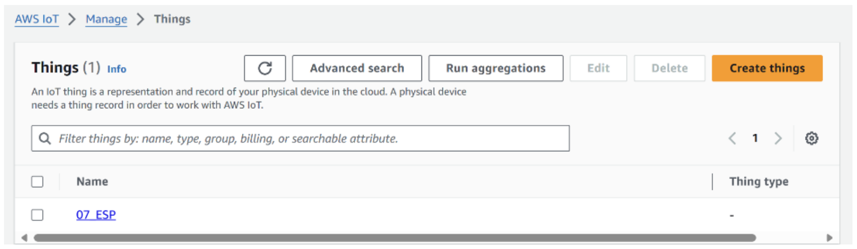
Public key file f6bf7ccd45fb33bbbad75d0...f09d5c1-public.pem.key	Download ✓ Key downloaded
Private key file f6bf7ccd45fb33bbbad75d0...09d5c1-private.pem.key	Download ✓ Key downloaded

Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint RSA 2048 bit key: Amazon Root CA 1	Download
Amazon trust services endpoint ECC 256 bit key: Amazon Root CA 3	Download

If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available in our developer guides. Learn more



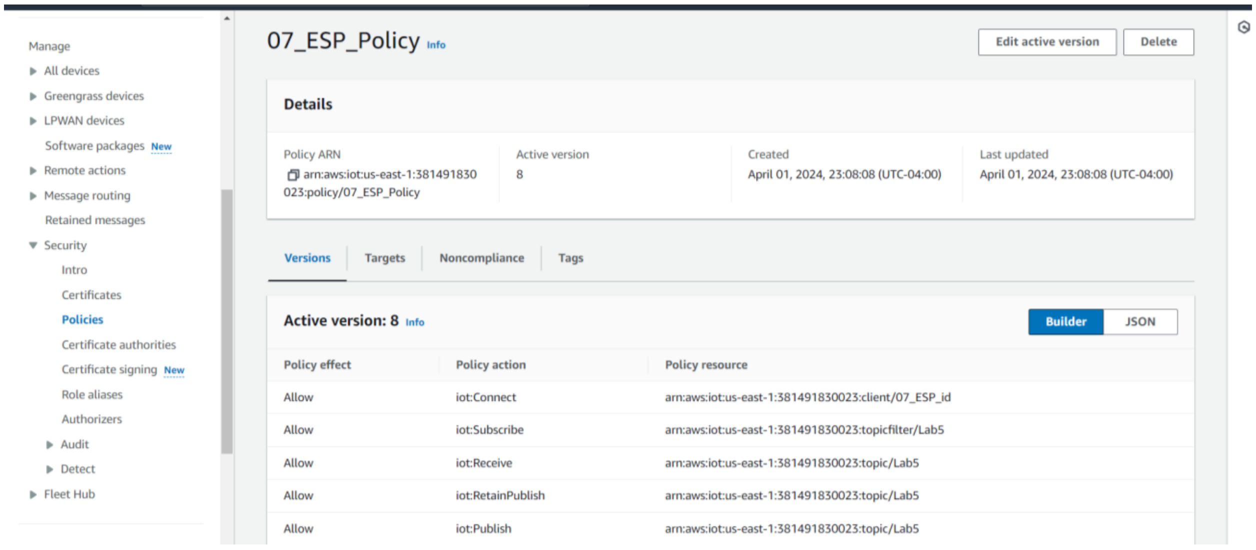
Things (1) Info

An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

Filter things by: name, type, group, billing, or searchable attribute.

Name	Thing type
07_ESP	-

b. *Create an AWS IoT policy*. This should be a secure policy used for this assignment. This means that the allowed actions and resources should be specific to what is being asked here. Your ESP Thing should be allowed to connect with the client ID "($YOUR_NUMBER)_ESP_id" and publish to the "Lab5" topic. Please provide a screenshot of the created policy. (10 point)



07_ESP_Policy Info

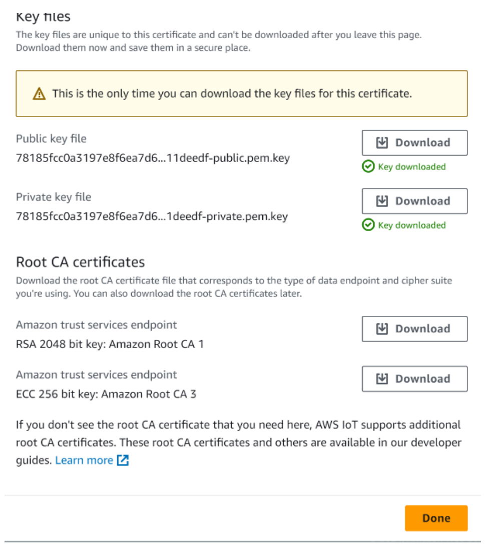
Details

Policy ARN	Active version	Created	Last updated
arn:aws:iot:us-east-1:381491830023:policy/07_ESP_Policy	8	April 01, 2024, 23:08:08 (UTC-04:00)	April 01, 2024, 23:08:08 (UTC-04:00)

Active version: 8 Info

Policy effect	Policy action	Policy resource
Allow	iot:Connect	arn:aws:iot:us-east-1:381491830023:client/07_ESP_id
Allow	iot:Subscribe	arn:aws:iot:us-east-1:381491830023:topicfilter/Lab5
Allow	iot:Receive	arn:aws:iot:us-east-1:381491830023:topic/Lab5
Allow	iot:RetainPublish	arn:aws:iot:us-east-1:381491830023:topic/Lab5
Allow	iot:Publish	arn:aws:iot:us-east-1:381491830023:topic/Lab5

c. *Create a second Thing object*. Please provide a screenshot of the listed keys and certificate which you can download. This Thing should be called ($YOUR_NUMBER)_mosquitto. (10 point)



Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

⚠ This is the only time you can download the key files for this certificate.

Public key file
78185fcc0a3197e8f6ea7d6...11deedf-public.pem.key

Private key file
78185fcc0a3197e8f6ea7d6...1deedf-private.pem.key

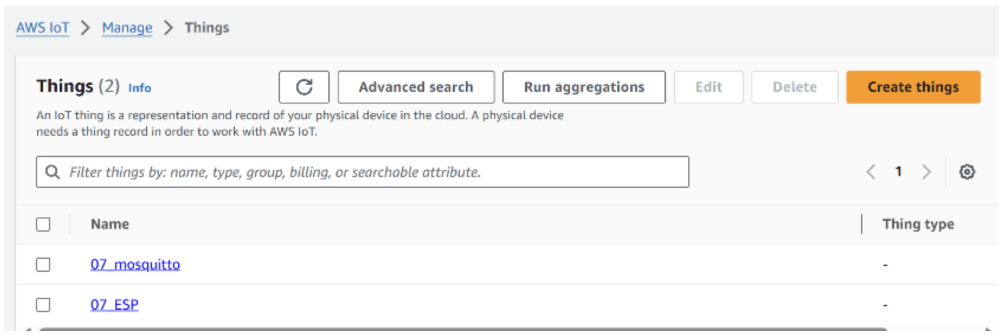
Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint
RSA 2048 bit key: Amazon Root CA 1

Amazon trust services endpoint
ECC 256 bit key: Amazon Root CA 3

If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available in our developer guides. Learn more



AWS IoT > Manage > Things

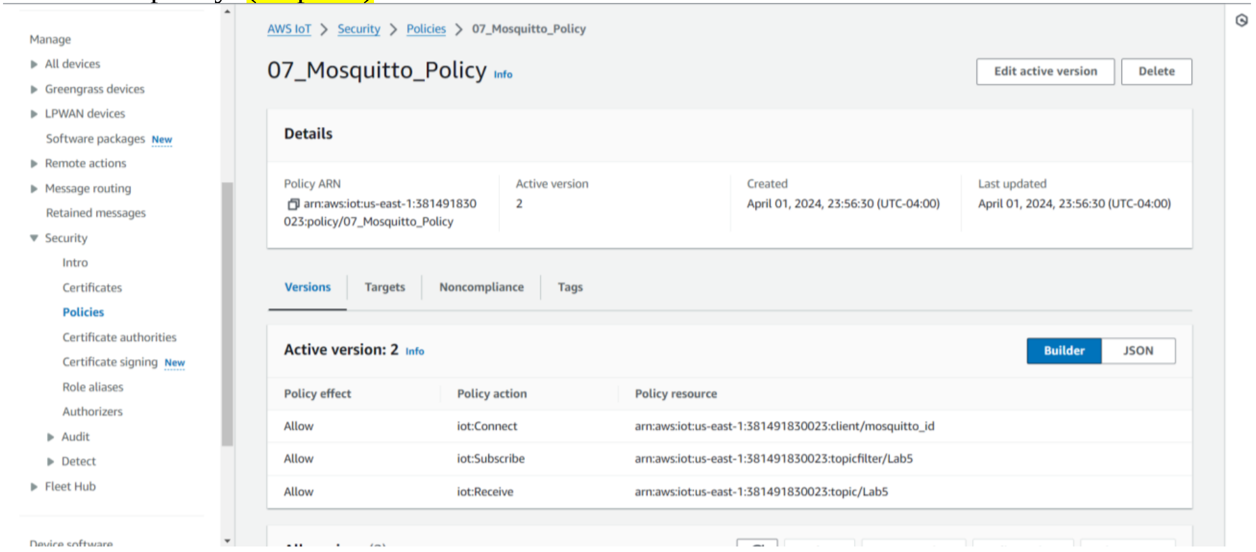
Things (2) Info

An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

Filter things by: name, type, group, billing, or searchable attribute.

Name	Thing type
07_mosquitto	-
07_ESP	-

d. *Create an AWS IoT policy*. This should be a secure policy used for this assignment. This means that the allowed actions and resources should be specific to what is being asked here. Your mosquitto Thing should be allowed to connect with the client ID “mosquitto_id” and subscribe to the “Lab5” topicfilter. Please provide a screenshot of the created policy. (10 point)



AWS IoT > Security > Policies > 07_Mosquitto_Policy

07_Mosquitto_Policy Info

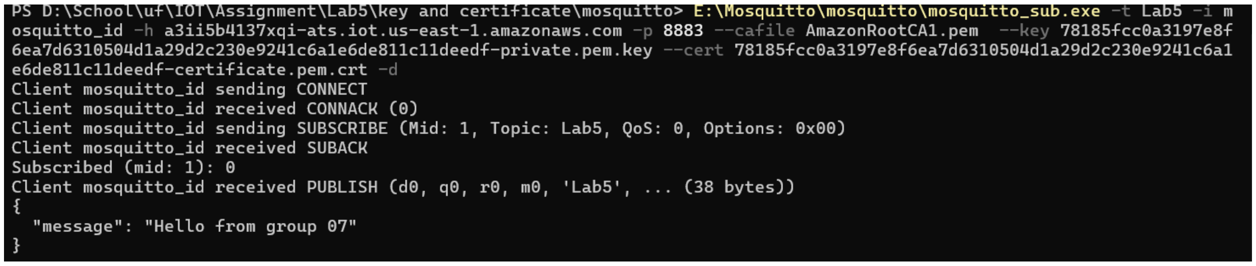
Details

Policy ARN	Active version	Created	Last updated
arn:aws:iot:us-east-1:381491830023:policy/07_Mosquitto_Policy	2	April 01, 2024, 23:56:30 (UTC-04:00)	April 01, 2024, 23:56:30 (UTC-04:00)

Active version: 2 Info

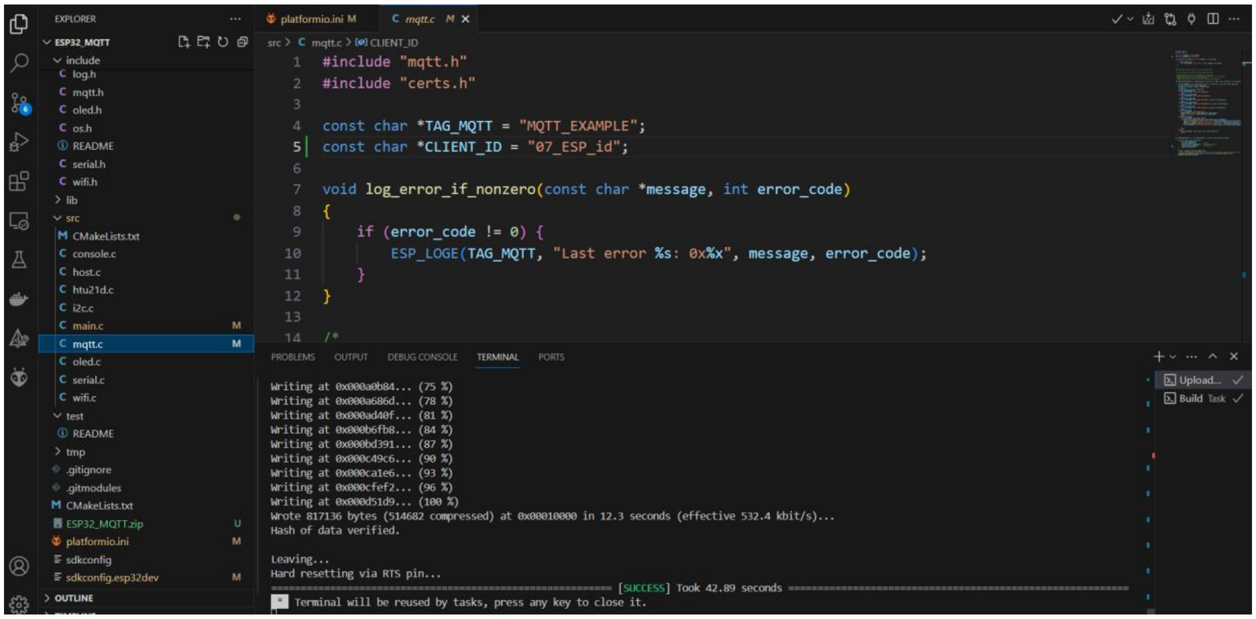
Policy effect	Policy action	Policy resource
Allow	iot:Connect	arn:aws:iot:us-east-1:381491830023:client/mosquitto_id
Allow	iot:Subscribe	arn:aws:iot:us-east-1:381491830023:topicfilter/Lab5
Allow	iot:Receive	arn:aws:iot:us-east-1:381491830023:topic/Lab5

3. Use the downloaded keys and certificates to have mosquitto subscribe to the “Lab5” topic on your AWS account. Send a message with you. r group number from the IoT MQTT console to that topic and show that mosquitto was able to receive it with a screen shot. (10 point)



```
PS D:\School\uf\IOT\Assignment\Lab5\key and certificate\mosquitto> E:\Mosquitto\mosquitto\mosquitto_sub.exe -t Lab5 -i m
osquitto_id -h a3ii5b4137xqi-ats.iot.us-east-1.amazonaws.com -p 8883 --cafile AmazonRootCA1.pem  --key 78185fcc0a3197e8f
6ea7d6310504d1a29d2c230e9241c6a1e6de811c11deedf-private.pem.key --cert 78185fcc0a3197e8f6ea7d6310504d1a29d2c230e9241c6a1
e6de811c11deedf-certificate.pem.crt -d
Client mosquitto_id sending CONNECT
Client mosquitto_id received CONNACK (0)
Client mosquitto_id sending SUBSCRIBE (Mid: 1, Topic: Lab5, QoS: 0, Options: 0x00)
Client mosquitto_id received SUBACK
Subscribed (mid: 1): 0
Client mosquitto_id received PUBLISH (d0, q0, r0, m0, 'Lab5', ... (38 bytes))
{
  "message": "Hello from group 07"
}
```

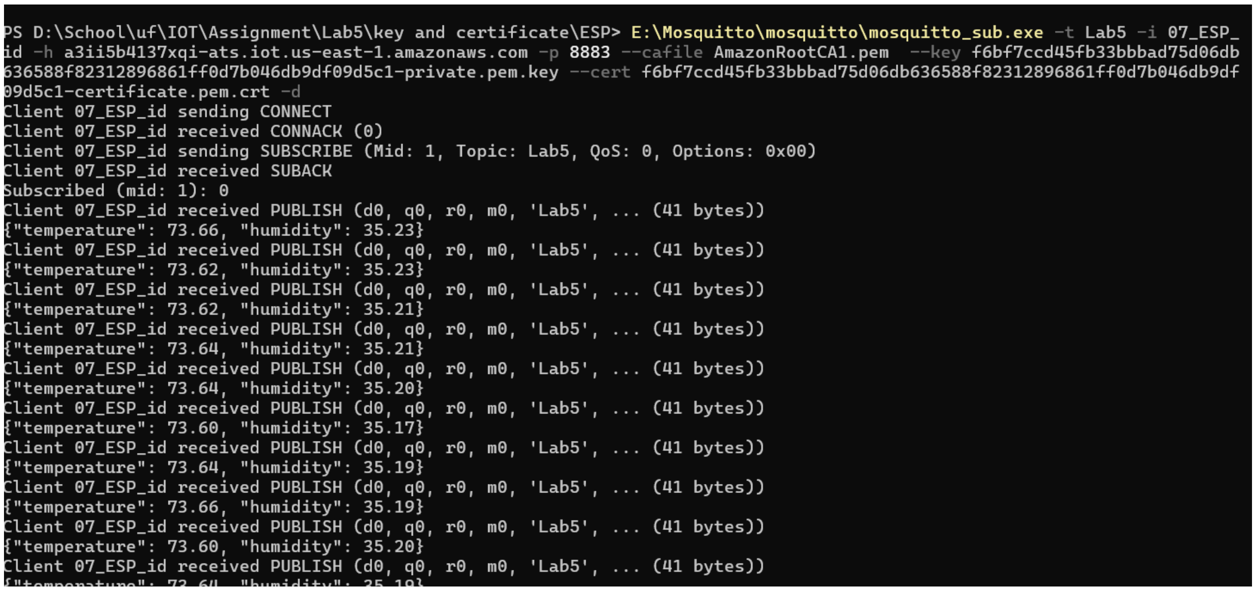
4. Modify the provided code to have your ESP client id match the one allowed to connect. Add your certificate, private key, and root ca to the code. Modify the topic the ESP should publish to be “Lab5” Compile and upload to your ESP. (10 point)



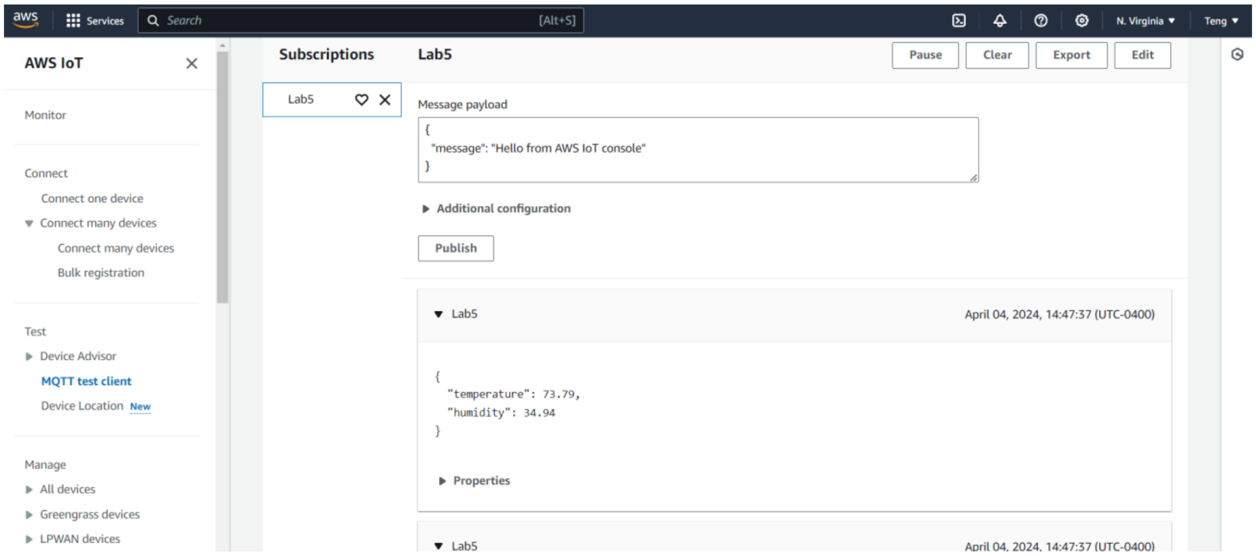
```
#include "mqtt.h"
#include "certs.h"

const char *TAG_MQTT = "MQTT_EXAMPLE";
const char *CLIENT_ID = "07_ESP_id";

void log_error_if_nonzero(const char *message, int error_code)
{
    if (error_code != 0) {
        ESP_LOGE(TAG_MQTT, "Last error %s: 0x%x", message, error_code);
    }
}

/*
```

```
Writing at 0x000a0b84... (75 %)
Writing at 0x000a686d... (78 %)
Writing at 0x000ad40f... (81 %)
Writing at 0x000b6fb8... (84 %)
Writing at 0x000bd391... (87 %)
Writing at 0x000c49c6... (90 %)
Writing at 0x000ca1e6... (93 %)
Writing at 0x000cfef2... (96 %)
Writing at 0x000d51d9... (100 %)
Wrote 817136 bytes (514682 compressed) at 0x00010000 in 12.3 seconds (effective 532.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
========================================================= [SUCCESS] Took 42.89 seconds =========================================================
 *  Terminal will be reused by tasks, press any key to close it.
```

5. Show that your mosquitto subscriber is receiving messages from the ESP with a screen shot. (10 point)



```
PS D:\School\uf\IOT\Assignment\Lab5\key and certificate\ESP> E:\Mosquitto\mosquitto\mosquitto_sub.exe -t Lab5 -i 07_ESP_
id -h a3ii5b4137xqi-ats.iot.us-east-1.amazonaws.com -p 8883 --cafile AmazonRootCA1.pem  --key f6bf7ccd45fb33bbbad75d06db
636588f82312896861ff0d7b046db9df09d5c1-private.pem.key --cert f6bf7ccd45fb33bbbad75d06db636588f82312896861ff0d7b046db9df
09d5c1-certificate.pem.crt -d
Client 07_ESP_id sending CONNECT
Client 07_ESP_id received CONNACK (0)
Client 07_ESP_id sending SUBSCRIBE (Mid: 1, Topic: Lab5, QoS: 0, Options: 0x00)
Client 07_ESP_id received SUBACK
Subscribed (mid: 1): 0
Client 07_ESP_id received PUBLISH (d0, q0, r0, m0, 'Lab5', ... (41 bytes))
{"temperature": 73.66, "humidity": 35.23}
Client 07_ESP_id received PUBLISH (d0, q0, r0, m0, 'Lab5', ... (41 bytes))
{"temperature": 73.62, "humidity": 35.23}
Client 07_ESP_id received PUBLISH (d0, q0, r0, m0, 'Lab5', ... (41 bytes))
{"temperature": 73.62, "humidity": 35.21}
Client 07_ESP_id received PUBLISH (d0, q0, r0, m0, 'Lab5', ... (41 bytes))
{"temperature": 73.64, "humidity": 35.21}
Client 07_ESP_id received PUBLISH (d0, q0, r0, m0, 'Lab5', ... (41 bytes))
{"temperature": 73.64, "humidity": 35.20}
Client 07_ESP_id received PUBLISH (d0, q0, r0, m0, 'Lab5', ... (41 bytes))
{"temperature": 73.60, "humidity": 35.17}
Client 07_ESP_id received PUBLISH (d0, q0, r0, m0, 'Lab5', ... (41 bytes))
{"temperature": 73.64, "humidity": 35.19}
Client 07_ESP_id received PUBLISH (d0, q0, r0, m0, 'Lab5', ... (41 bytes))
{"temperature": 73.66, "humidity": 35.19}
Client 07_ESP_id received PUBLISH (d0, q0, r0, m0, 'Lab5', ... (41 bytes))
{"temperature": 73.60, "humidity": 35.20}
Client 07_ESP_id received PUBLISH (d0, q0, r0, m0, 'Lab5', ... (41 bytes))
```



Message payload

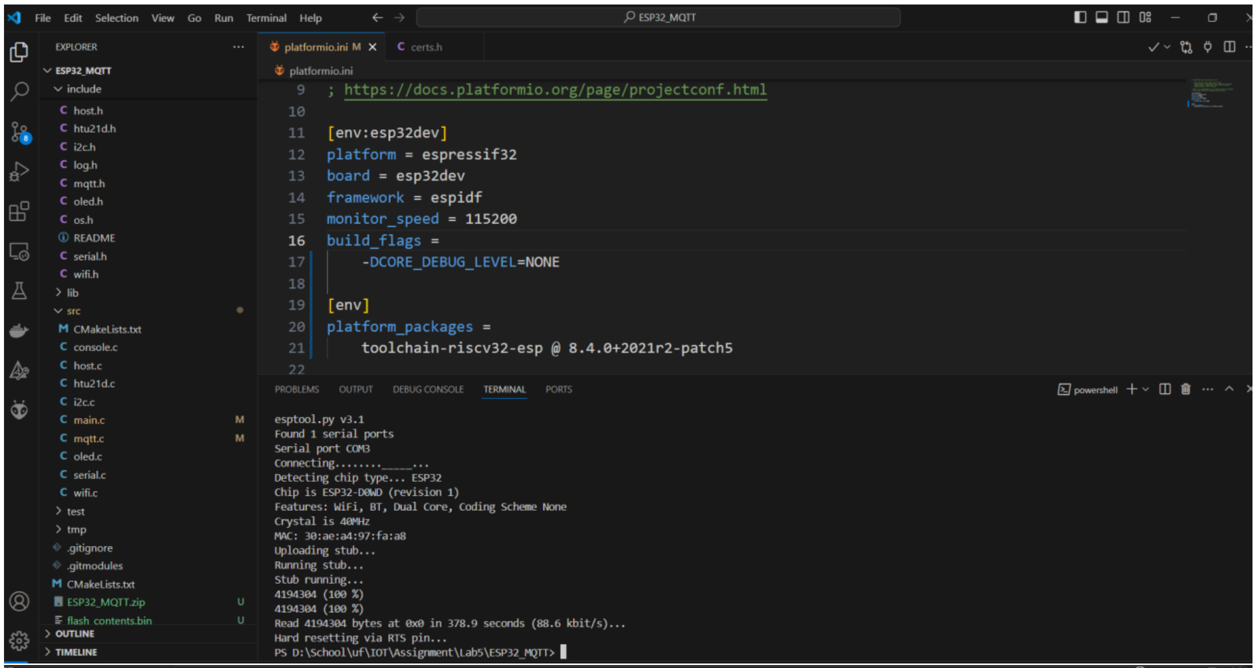
```
{
  "message": "Hello from AWS IoT console"
}
```

Lab5 April 04, 2024, 14:47:37 (UTC-0400)

```
{
  "temperature": 73.79,
  "humidity": 34.94
}
```

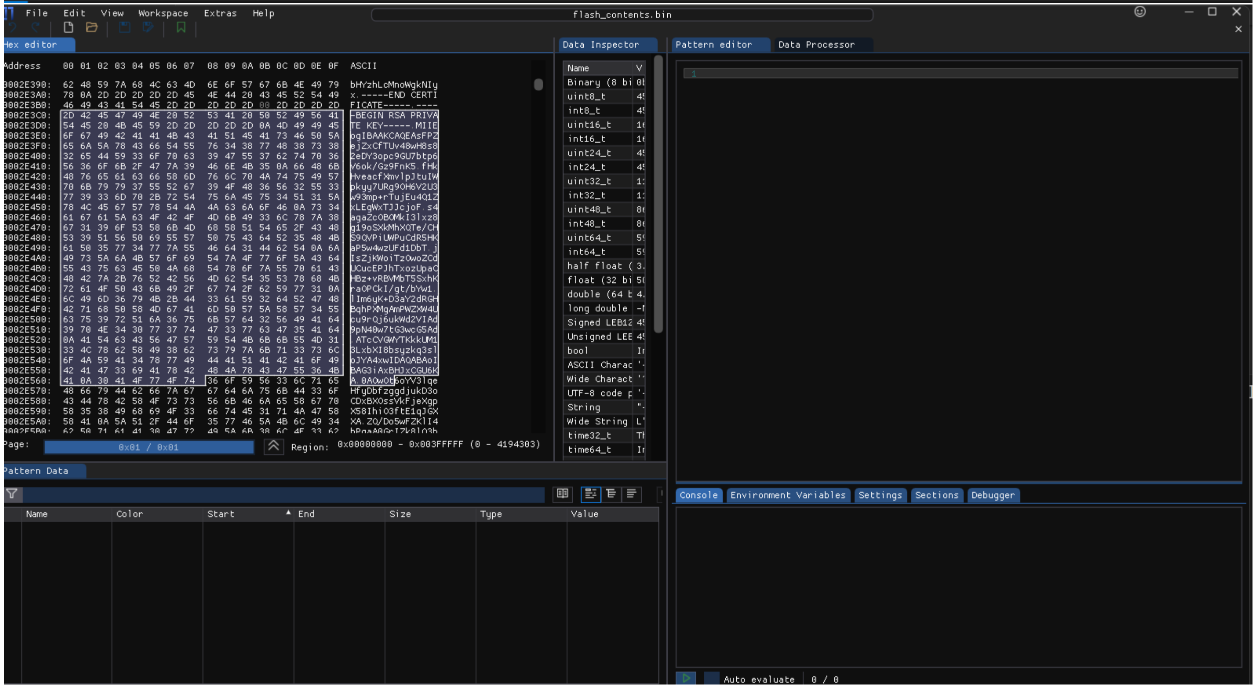
Lab5 April 04, 2024, 14:47:37 (UTC-0400)

6. Please dump your IoT kit's flash with *esptool.py* and search the flash for the private key of the IoT kit.



```
; https://docs.platformio.org/page/projectconf.html

[env:esp32dev]
platform = espressif32
board = esp32dev
framework = espidf
monitor_speed = 115200
build_flags =
    -DCORE_DEBUG_LEVEL=NONE

[env]
platform_packages =
    toolchain-riscv32-esp @ 8.4.0+2021r2-patch5
```

```
esptool.py v3.1
Found 1 serial ports
Serial port COM3
Connecting........_____...
Detecting chip type... ESP32
Chip is ESP32-D0WD (revision 1)
Features: WiFi, BT, Dual Core, Coding Scheme None
Crystal is 40MHz
MAC: 30:ae:a4:97:fa:a8
Uploading stub...
Running stub...
Stub running...
4194304 (100 %)
4194304 (100 %)
Read 4194304 bytes at 0x0 in 378.9 seconds (88.6 kbit/s)...
Hard resetting via RTS pin...
PS D:\School\uf\IOT\Assignment\Lab5\ESP32_MQTT>
```



```
Address    00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  ASCII

0002E390:  62 48 59 7A 68 4C 63 4D  6E 6F 57 67 6B 4E 49 79  bHYzhLcMnoWgkNIy
0002E3A0:  78 0A 2D 2D 2D 2D 2D 45  4E 44 20 43 45 52 54 49  x.-----END CERTI
0002E3B0:  46 49 43 41 54 45 2D 2D  2D 2D 2D 00 2D 2D 2D 2D  FICATE-----.----
0002E3C0:  2D 42 45 47 49 4E 20 52  53 41 20 50 52 49 56 41  -BEGIN RSA PRIVA
0002E3D0:  54 45 20 4B 45 59 2D 2D  2D 2D 2D 0A 4D 49 49 45  TE KEY-----.MIIE
0002E3E0:  6F 67 49 42 41 41 4B 43  41 51 45 41 73 46 50 5A  ogIBAAKCAQEAsFPZ
0002E3F0:  65 6A 5A 78 43 66 54 55  76 34 38 77 48 38 73 38  ejZxCfTUv48wH8s8
0002E400:  32 65 44 59 33 6F 70 63  39 47 55 37 62 74 70 36  2eDY3opc9GU7btp6
0002E410:  56 36 6F 6B 2F 47 7A 39  46 6E 4B 35 0A 66 48 6B  V6ok/Gz9FnK5.fHk
0002E420:  48 76 65 61 63 66 58 6D  76 6C 70 4A 74 75 49 57  HveacfXmvlpJtuIW
0002E430:  70 6B 79 79 37 55 52 67  39 4F 48 36 56 32 55 33  pkyy7URg9OH6V2U3
0002E440:  77 39 33 6D 70 2B 72 54  75 6A 45 75 34 51 31 5A  w93mp+rTujEu4Q1Z
0002E450:  78 4C 45 67 57 78 54 4A  4A 63 6A 6F 46 0A 73 34  xLEgWxTJJcjoF.s4
0002E460:  61 67 61 5A 63 4F 42 4F  4D 6B 49 33 6C 78 7A 38  agaZcOBOMkI3lxz8
0002E470:  67 31 39 6F 53 58 6B 4D  68 58 51 54 65 2F 43 48  g19oSXkMhXQTe/CH
0002E480:  53 39 51 56 50 69 55 57  50 75 43 64 52 35 48 4B  S9QVPiUWPuCdR5HK
0002E490:  61 50 35 77 34 77 7A 55  46 64 31 44 62 54 0A 6A  aP5w4wzUFd1DbT.j
0002E4A0:  49 73 5A 6A 4B 57 6F 69  54 7A 4F 77 6F 5A 43 64  IsZjKWoiTzOwoZCd
0002E4B0:  55 43 75 63 45 50 4A 68  54 78 6F 7A 55 70 61 43  UCucEPJhTxozUpaC
0002E4C0:  48 42 7A 2B 76 52 42 56  4D 62 54 35 53 78 68 4B  HBz+vRBVMbT5SxhK
0002E4D0:  72 61 4F 50 43 6B 49 2F  67 74 2F 62 59 77 31 0A  raOPCkI/gt/bYw1.
0002E4E0:  6C 49 6D 36 79 4B 2B 44  33 61 59 32 64 52 47 48  lIm6yK+D3aY2dRGH
0002E4F0:  42 71 68 50 58 4D 67 41  6D 50 57 5A 58 57 34 55  BqhPXMgAmPWZXW4U
0002E500:  63 75 39 72 51 6A 36 75  6B 57 64 32 56 49 41 64  cu9rQj6ukWd2VIAd
0002E510:  39 70 4E 34 30 77 37 74  47 33 77 63 47 35 41 64  9pN40w7tG3wcG5Ad
0002E520:  0A 41 54 63 43 56 47 57  59 54 4B 6B 6B 55 4D 31  .ATcCVGWYTKkkUM1
0002E530:  33 4C 78 62 58 49 38 62  73 79 7A 6B 71 33 73 6C  3LxbXI8bsyzkq3sl
0002E540:  6F 4A 59 41 34 78 77 49  44 41 51 41 42 41 6F 49  oJYA4xwIDAQABAoI
0002E550:  42 41 47 33 69 41 78 42  48 4A 78 43 47 55 36 4B  BAG3iAxBHJxCGU6K
0002E560:  41 0A 30 41 4F 77 4F 74  36 6F 59 56 33 6C 71 65  A.0AOwOt6oYV3lqe
0002E570:  48 66 79 44 62 66 7A 67  67 64 6A 75 6B 44 33 6F  HfyDbfzggdjukD3o
0002E580:  43 44 78 42 58 4F 73 73  56 6B 46 6A 65 58 67 70  CDxBXOssVkFjeXgp
0002E590:  58 35 38 49 68 69 4F 33  66 74 45 31 71 4A 47 58  X58IhiO3ftE1qJGX
0002E5A0:  58 41 0A 5A 51 2F 44 6F  35 77 46 5A 4B 6C 49 34  XA.ZQ/Do5wFZKlI4
0002E5B0:  62 50 71 61 41 30 47 72  49 5A 6B 38 6C 4F 33 62  bPqaA0GrIZk8lO3b
Page:  0x01 / 0x01    Region:  0x00000000 - 0x003FFFFF (0 - 4194303)
```

a. Please discuss the security implications of the fact that the private key of the IoT kit can be stolen from the dumped flash. (10 point)

If the IoT kit's private key can be stolen from the dumped flash, it means that an attacker can steal important security credentials. Private keys are a core component of cryptography used to secure data transfers, primarily for authentication and encrypted communication. If an attacker obtains the private key, they can forge the identity of the device, enabling the attacker to communicate with the system or other devices as the device. Decrypt encrypted communications between devices or between devices and servers, potentially leading to the disclosure of sensitive information. Perform a man-in-the-middle attack (MITM), inserted between the device and the communicating counterpart to eavesdrop or tamper with the transmitted data. These security breaches can lead to data leakage, loss of control of the device, and security threats to the entire system.

b. Our IoT kit contains a crypto co-processor ATECC608A, which has an internal secure storage and hardware crypto acceleration. ATECC608A can be used to hold the private key, which cannot be extracted from outside of ATECC608A and does not leave the secure storage. All necessary crypto operations are done inside of ATECC608A. Please explain why the private key is needed by our IoT kit so that we need to protect it. (10 point)

Our IoT suite requires the protection of private keys, which are key to enabling encrypted communication and data security. Private keys are used to generate signatures to prove the authenticity of a message or data, ensuring the integrity of the data and the trustworthiness of the source. In addition, the private key is used to decrypt the message so that only the device or individual with the corresponding private key can decrypt the data encrypted by the public key.The ATECC608A, as a cryptographic coprocessor, provides a secure storage environment for storing sensitive information such as private keys. Security is greatly enhanced by the fact that the ATECC608A has internal secure storage and does not allow external access to the stored private key. All necessary encryption operations are done internally in the ATECC608A, which means that the private key does not leave the secure environment at any time, thus ensuring that the private key remains protected even if the rest of the device is breached. This design reduces the risk of the private key being stolen, ensures the security of the encryption operation, and improves the security level of the entire system.