AN12805
Establish Secure Connection with Private Cloud

Rev. 0 — 26 March 2020

Application Note

Contents

1 Introduction

Security is necessary as many devices connect to cloud. Cloud manufacturers like amazon/Ali cloud have more to ensure that the connection is secure. But if you want to establish private cloud, see this application note.

AN12258 of IoT Device Secure Connection with LoRa explains how to establish IoT secure connection with LoRa between gateway and end device. This application note explains how to establish secure connection with the private cloud. It describes the mechanisms and credentials involved to create a secure TLS connection between an IoT device and the private cloud.

The hardware is based on LPCXpresso55S69 Development Board with SX-ULPAN-2401 Wi-Fi adaptor board. The embedded software is based on LPCXpresso55S69 SDK 2.6.2. Mosquitto broker is installed on server/cloud.

2 Overview

The LPC55S6x MCU family builds the world's first general-purpose Cortex-M33-based microcontroller. It includes many security features like Arm, TrustZone, AES-256 encryption/decryption engine, SHA2 engine, Physical Unclonable Functon(PUF), Random Number Generator(RNG). The SDK for LPC55S6x MCU provides peripheral drivers, FreeRTOS, various other middleware, and an extensive and rich set of example applications.

For this application note, based on the SDK for LPC55S6x, the secure embedded software project is created rapidly.

Eclipse Mosquitto is an open source message broker that implements the MQTT protocol version 5.0, 3.3.1 and 3.1. Mosquitto is lightweight and is suitable for use on all devices from low-power single board computers to full servers. It is easy to create private cloud using Mosquito broker on Windows, Mac, Linux.

Transport Layer Security(TLS) is cryptographic protocol designed to provide communication security over a computer network. The TLS protocol aims primarily to provide privacy and data integrity between two or more communication computer applications. For this application note, TLS provides a secure communication.

3 Hardware platform

Figure 1. is system block diagram, LPC55s69 EVK board with SX-ULPAN-2401 Wi-Fi adaptor board is as device end. Technical and functional specifications are:

- LPC55S69 dual-core Arm Cortex-M33 microcontroller runs at up to 150 MHz.
- Micro SD card slot.
- UART and SPI port bridging from LPC55S69 target to USB via the onboard debug probe. High-speed and full speed USB ports with micro A/B connector for host or device functionality.
- Onboard high-speed USB, Link2 debug probe with CMSIS-DAP and SEGGER J-Link protocol options.
- NXP MMA8652FCR1 accelerometer.
- RGB user LED, plus Reset, ISP, Wake, and user buttons.

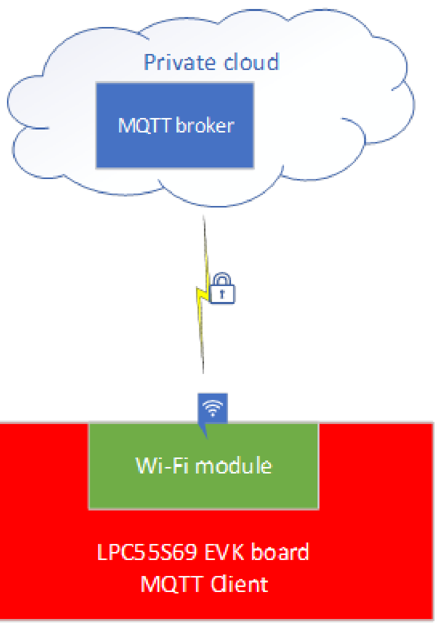


Figure 1. System block diagram

4 Software platform

Transport layer security (TLS) protocol is used to establish secure connection to private cloud.

So key pairs and certificates are must establish a secure channel.

4.1 Create key pairs and certificates

To establish secure channel, digital key pairs and certificates are needed between device and private cloud. OpenSSL tool is used to create digital key pairs and certificates.

```
// create root key pair and root CA is self-signed
#./openssl genrsa -out ca.key 2048
#./openssl req -new -x509 -days 1826 -key ca.key -out ca.crt
// create cloud key pair and cloud CA is signed by root CA
#./openssl genrsa -out server.key 2048
#./openssl req -new -out server.csr -key server.key
#./openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days 360

// create device key pair and device CA is signed by root CA
#./openssl genrsa -out client.key 2048
#./openssl req -new -out client.csr -key client.key
#./openssl x509 -req -in client.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out client.crt -days 360
```

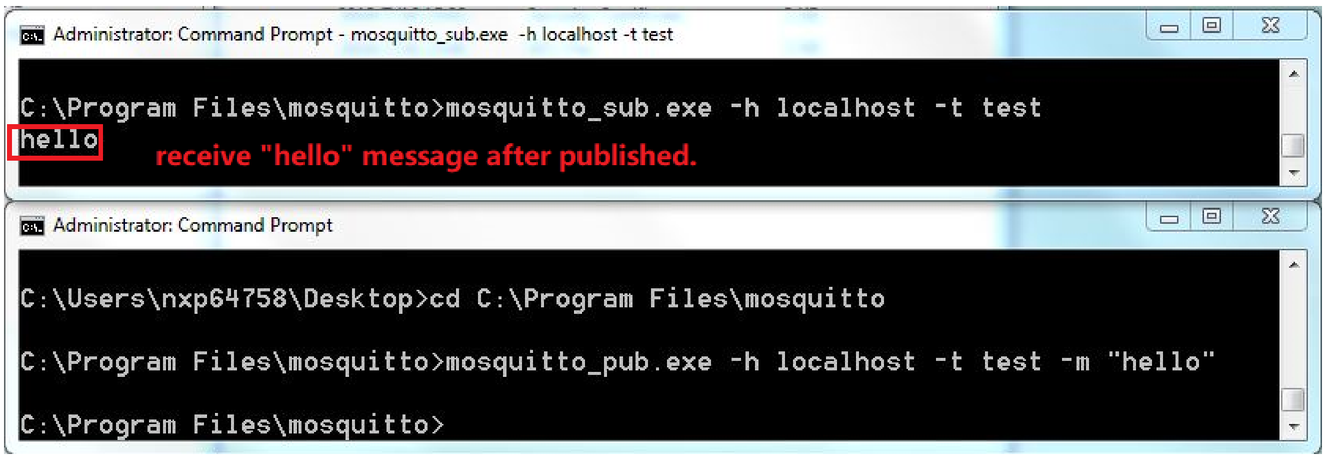
In the DEMO, another device key pair and CA created are used by another device.

NOTE

When creating csr file, Common Name should be your ethernet IP address, for example 192.168.0.170

4.2 Cloud end software

Cloud end must install mosquitto that can be downloaded from the web https://mosquitto.org/download/. After installation, a test can be done as shown in Figure 2. to confirm if moquitto works fine.



```
Administrator: Command Prompt - mosquitto_sub.exe  -h localhost -t test

C:\Program Files\mosquitto>mosquitto_sub.exe -h localhost -t test
hello
```

```
Administrator: Command Prompt

C:\Users\nxp64758\Desktop>cd C:\Program Files\mosquitto

C:\Program Files\mosquitto>mosquitto_pub.exe -h localhost -t test -m "hello"

C:\Program Files\mosquitto>
```

Figure 2. Mosquitto test

4.3 Embedded software

The embedded software in the device end based on SDK_2.6.2_LPCXpresso55s69 can be downloaded from the web https://mcuxpresso.nxp.com/. Mbedtls library, FreeRTOS, and Wi-Fi library are included in SDK.

MQTT code is from https://github.com/LiamBindle/MQTT-C. MQTT-C is an MQTT v3.1.1 client written in C. MQTT is a lightweight publisher-subscriber-based messaging protocol, commonly used in IoT and networking applications where high-latency and low data-rate links are expected. MQTT-C provides a transparent platform abstraction layer(PAL) that ports to new platforms easy. In the mqtt_pal.c file of the project, mqtt_pal_sendall() and mqtt_pal_recvall() are rewritten to match Wi-Fi library.

FreeRTOS creates two main tasks, one is "Atheros_Wifi_Task" that is to process the Wi-Fi data and the other. Another task is “task_main” that is to connect the router to Wi-Fi, to make TLS1.2 handshake, to connect MQTT broker and to subscribe/publish messages. See Figure 3.

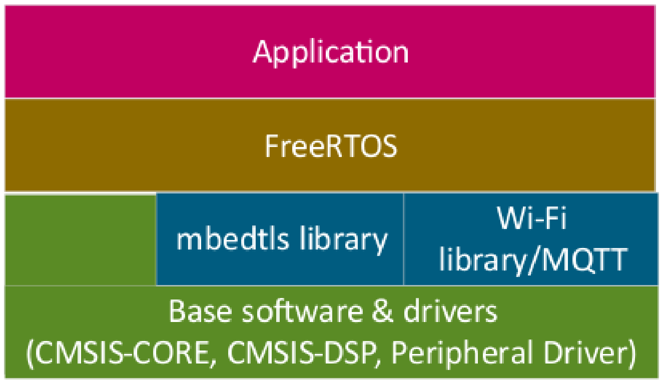


Figure 3. Software architecture

5 Hands-on

5.1 Set up Hardware

Follow the steps given in advance .

1. Plug the SX-ULPAN-2401 adaptor board to stackable headers(P16, P17, P18, P19) of LPC55S69 EVK board.
2. Connect personal computer to the wireless router and confirm that the network is fine. See Figure 4.

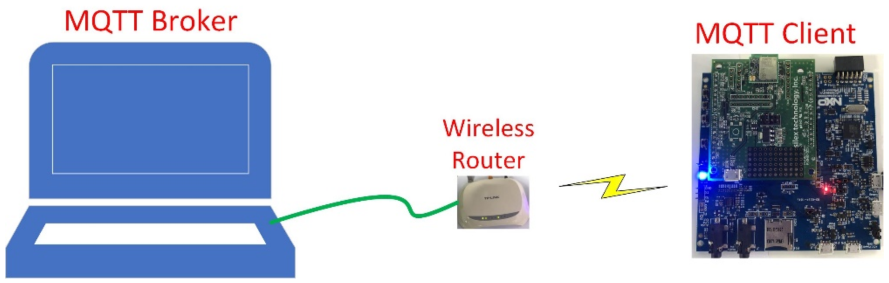


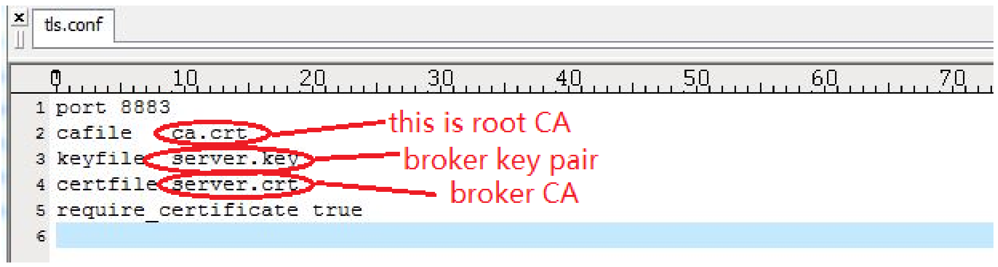
Figure 4. Hardware set up

5.2 Set up software

5.2.1 Mosquitto broker setting

Before startup of Mosquitto broker,configure as follows:.

1. Copy Root CA, broker key, and broker CA to installation directory(it is C:\Program Files\mosquitto in my personal computer).
2. Create mosquitto broker configuration file (it is named of tls.conf in my PC) and put it in installation directory, see Figure 5.



```
port 8883
cafile ca.crt
keyfile server.key
certfile server.crt
require_certificate true
```

Figure 5. Mosquitto broker configuration file

5.2.2 Device end setting

Download LPC55S69 EVK board SDK from https://mcuxpresso.nxp.com/. Remember to add middleware mbedtls, wifi_qca and amazon-freeRTOS, and to select toolchain Keil MDK. Copy the contents of the folder “mosquitto_client_code_LPC55S69” to LPC55S69 SDK project. The folder “mosquitto-tls-client” will be found in the directory “.\boards \lpcxpresso55s69\wifi_qca_examples”. Open the embedded project qca_demo.uvmpw in \boards \lpcxpresso55s69\wifi_qca_examples\mosquitto-tls-client\cm33_core0\mdk folder with MDK V5.25.2 to configure it.

1. In certificate.c file, to use CA created in 4.1 to replace mbedtls_mosquitto_test_ca_crt[] array. Use device key created in 4.1 to replace mbedtls_mosquitto_test_client_crt[] array. Use device CA created in 4.1 to replace mbedtls_mosquitto_test_client_key[] array.
2. In config.h file, to configure Wi-Fi network with “AP_SSID” and “AP_PASSPHRASE” to connect your Wi-Fi router.

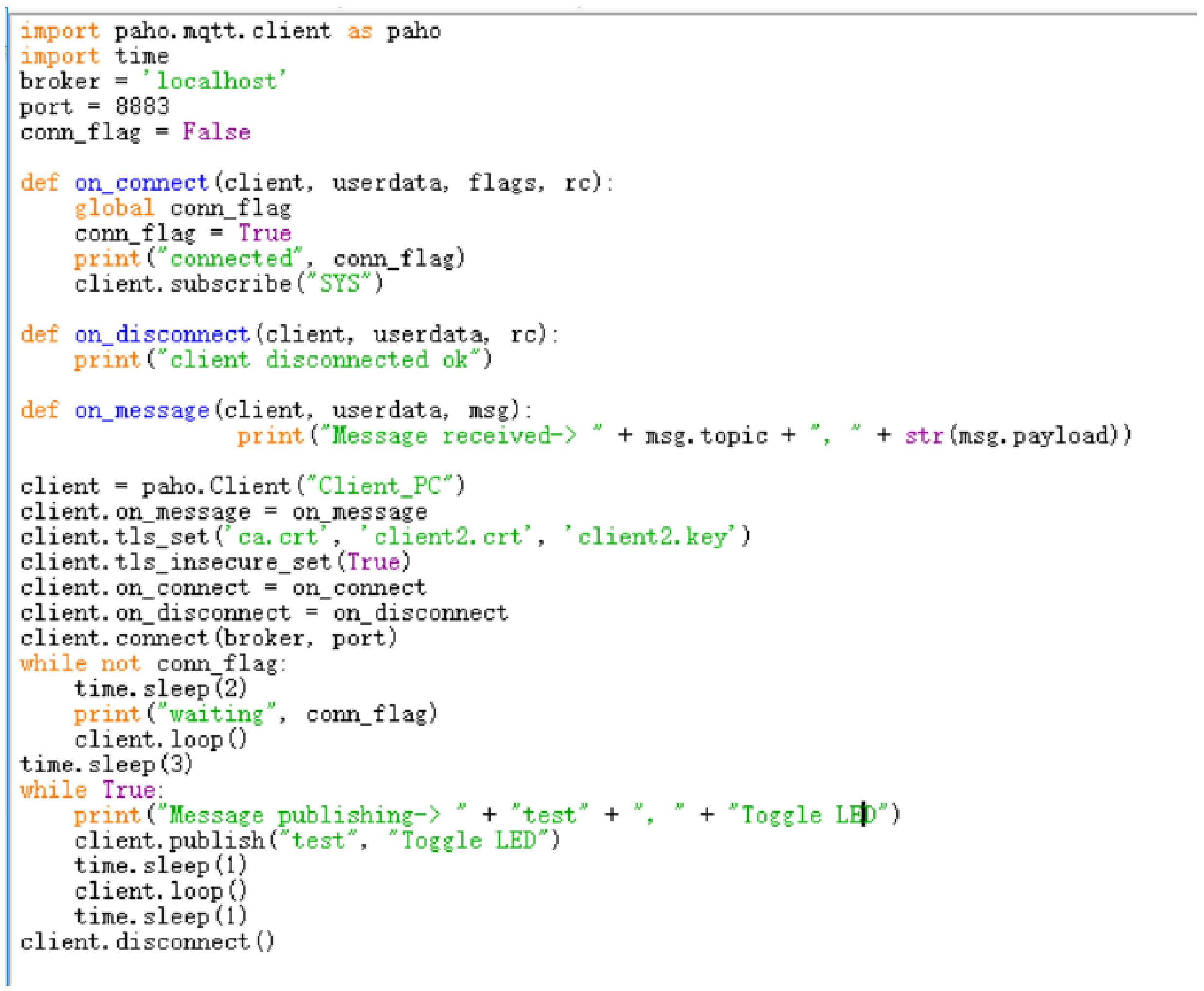
3. In config.h file, to configure broker name and IP with “CONFIG_BROKER_HOST_NAME” and “CONFIG_BROKER_HOST_IP”.
4. Build the project and download image file to LPC55S69.

5.2.3 Another MQTT client on PC

In order to demonstrate, create another MQTT client on the same PC with mosquitto broker.

1. Create client key pair and CA as described in section 4.1.
2. Edit the client code using python language. It can be run in Python 2.7.15. See Figure 6.

After connecting to MQTT broker, the MQTT client will subscribe a topic “SYS”, and publish the topic “test” and message “Toggle LED” continuously.

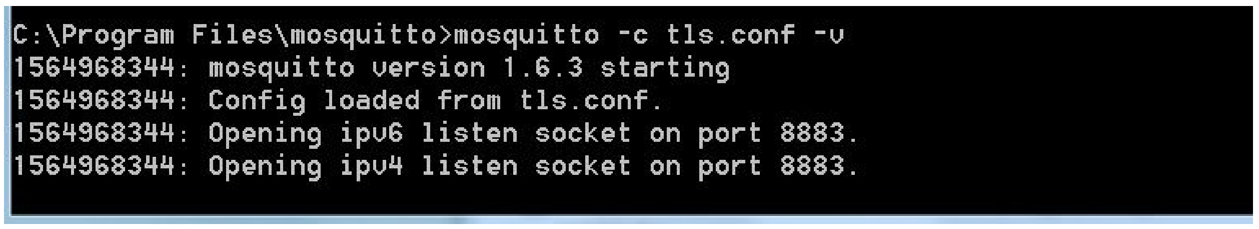


```
import paho.mqtt.client as paho
import time
broker = 'localhost'
port = 8883
conn_flag = False

def on_connect(client, userdata, flags, rc):
    global conn_flag
    conn_flag = True
    print("connected", conn_flag)
    client.subscribe("SYS")

def on_disconnect(client, userdata, rc):
    print("client disconnected ok")

def on_message(client, userdata, msg):
                print("Message received-> " + msg.topic + ", " + str(msg.payload))

client = paho.Client("Client_PC")
client.on_message = on_message
client.tls_set('ca.crt', 'client2.crt', 'client2.key')
client.tls_insecure_set(True)
client.on_connect = on_connect
client.on_disconnect = on_disconnect
client.connect(broker, port)
while not conn_flag:
    time.sleep(2)
    print("waiting", conn_flag)
    client.loop()
time.sleep(3)
while True:
    print("Message publishing-> " + "test" + ", " + "Toggle LED")
    client.publish("test", "Toggle LED")
    time.sleep(1)
    client.loop()
    time.sleep(1)
client.disconnect()
```

Figure 6. MQTT client code with Python language

5.3 Run the DEMO

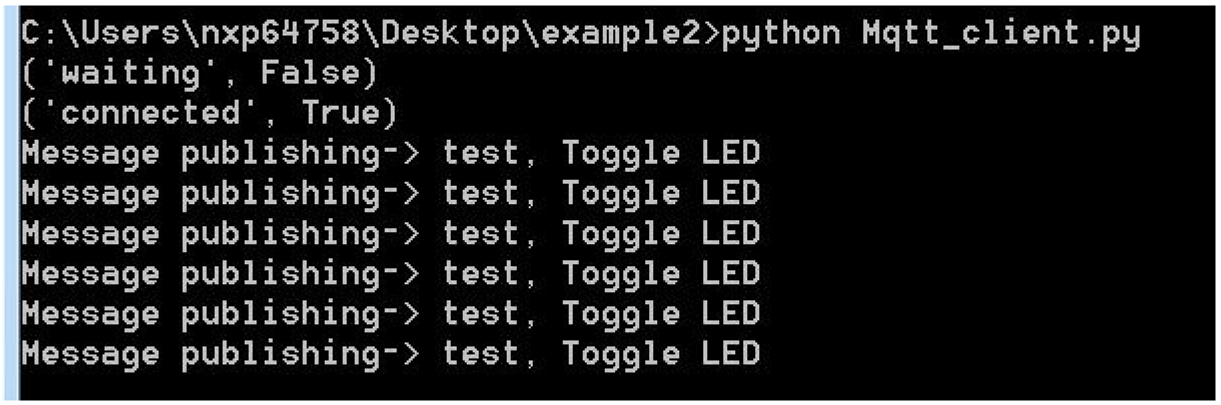
1. Run mosquitto broker like Figure 7. The MQTT broker is in the listening state.



```
C:\Program Files\mosquitto>mosquitto -c tls.conf -v
1564968344: mosquitto version 1.6.3 starting
1564968344: Config loaded from tls.conf.
1564968344: Opening ipv6 listen socket on port 8883.
1564968344: Opening ipv4 listen socket on port 8883.
```

Figure 7. Startup mosquitto broker

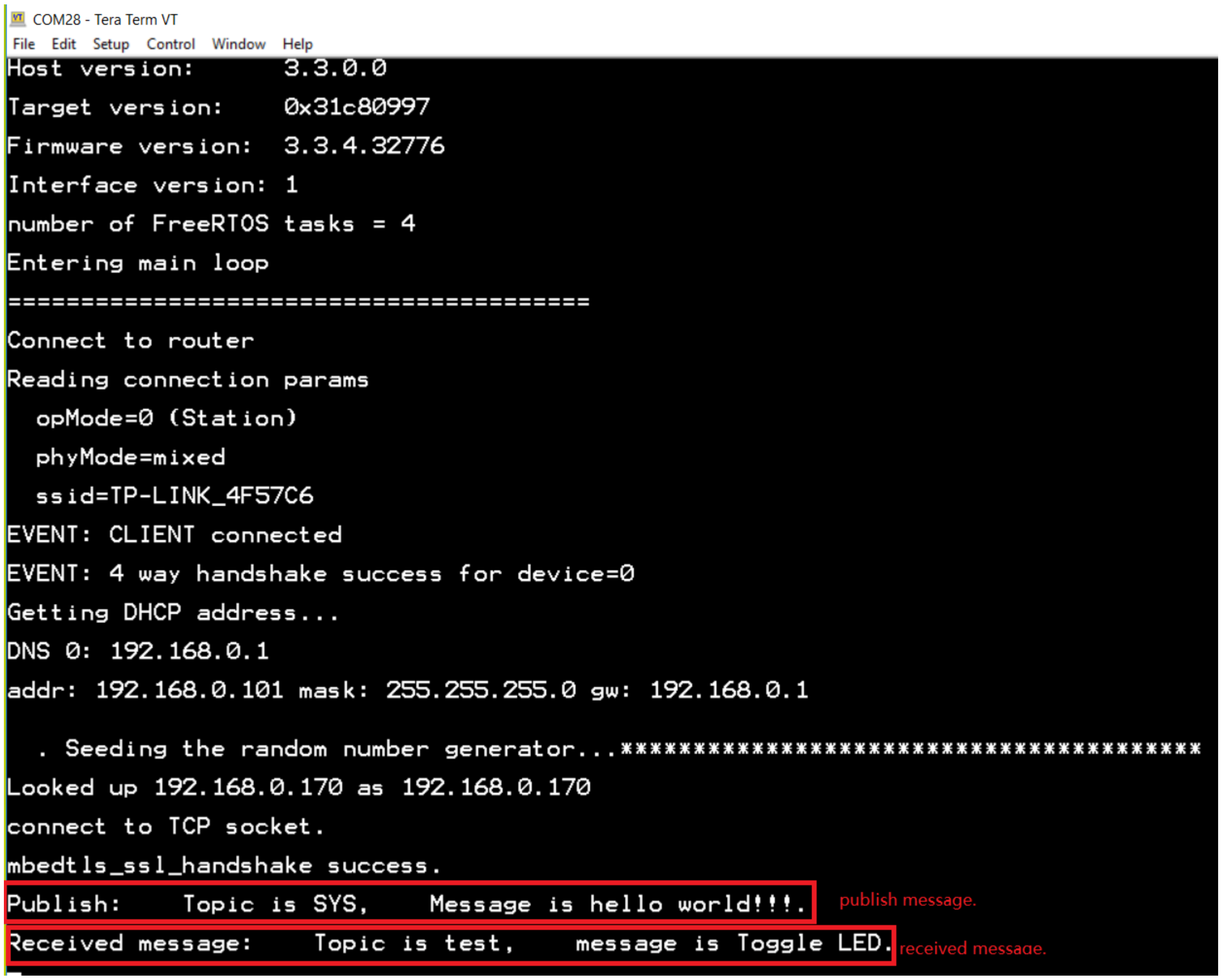
2. Run MQTT client on PC. See Figure 8.



```
C:\Users\nxp64758\Desktop\example2>python Mqtt_client.py
('waiting', False)
('connected', True)
Message publishing-> test, Toggle LED
Message publishing-> test, Toggle LED
Message publishing-> test, Toggle LED
Message publishing-> test, Toggle LED
Message publishing-> test, Toggle LED
Message publishing-> test, Toggle LED
```

Figure 8. Run MQTT client on PC

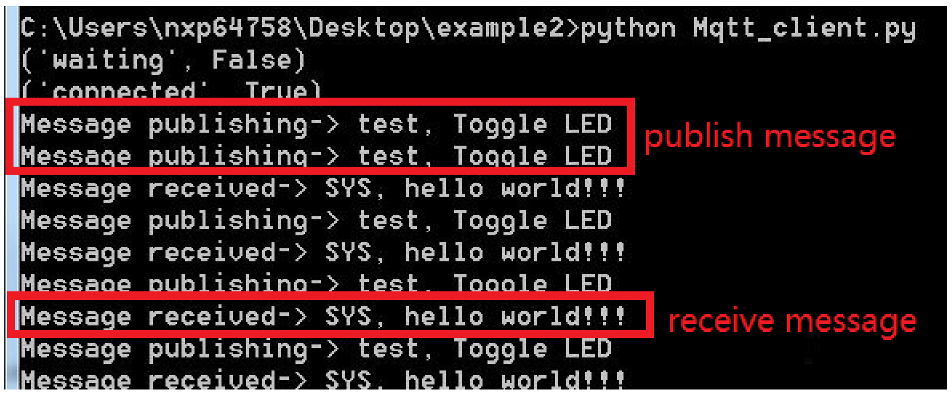
3. Connect a micro USB cable between the PC and P6 port of LPC55S69 EVK. Open a serial terminal with special settings(115200 baud rate, 8 data bits, No parity, One stop bit, No flow control). Press “reset” button to reset LPC55S69 EVK. Some information will be printed, see Figure 9.



```
Host version:       3.3.0.0
Target version:     0x31c80997
Firmware version:   3.3.4.32776
Interface version: 1
number of FreeRTOS tasks = 4
Entering main loop
========================================
Connect to router
Reading connection params
  opMode=0 (Station)
  phyMode=mixed
  ssid=TP-LINK_4F57C6
EVENT: CLIENT connected
EVENT: 4 way handshake success for device=0
Getting DHCP address...
DNS 0: 192.168.0.1
addr: 192.168.0.101 mask: 255.255.255.0 gw: 192.168.0.1

  . Seeding the random number generator...****************************************
Looked up 192.168.0.170 as 192.168.0.170
connect to TCP socket.
mbedtls_ssl_handshake success.
Publish:     Topic is SYS,     Message is hello world!!!.
Received message:     Topic is test,     message is Toggle LED.
```

Figure 9. LPC55S69 EVK log

After connecting with TLS1.2, the message is encrypted. LPC55S69 EVK receives the topic “test” from MQTT client and toggles LED. On the MQTT client, publish messages continuously and receive the topic “SYS” messages. See Figure 10.



```
C:\Users\nxp64758\Desktop\example2>python Mqtt_client.py
('waiting', False)
('connected', True)
Message publishing-> test, Toggle LED
Message publishing-> test, Toggle LED
Message received-> SYS, hello world!!!
Message publishing-> test, Toggle LED
Message received-> SYS, hello world!!!
Message publishing-> test, Toggle LED
Message received-> SYS, hello world!!!
Message publishing-> test, Toggle LED
Message received-> SYS, hello world!!!
```

Figure 10. MQTT client running

6 Conclusion

This application note describes how to set up secure connection between device and private cloud. It is easy to create a private cloud as this application note. Customer can download the related project to set up the whole secure connection.

7 References

- LPC55S6x/LPC55S2x/LPC552x User manual.
- https://mosquitto.org/
- https://mcuxpresso.nxp.com/
- https://en.wikipedia.org/wiki/Transport_Layer_Security

Date of release: 26 March 2020

Document identifier: AN12805