



Projet TMC

Réalisé par :

Amadou Ourry DIALLO

Wajdi KILANI

Moetaz RABAI

Table des matières

1	Problématique	2
2	Mises en place des communications	2
2.1	WiFi et MQTT	2
2.1.1	De l'ESP8266 vers le Raspberry Pi	2
2.1.2	La connexion du client MQTT de l'ESP8266 vers le serveur MQTT du Raspberry Pi	2
2.2	Lora : entre les deux Raspberry	3
3	Chiffrement	3
3.1	Par courbe elliptique tirant parti de l'ATECC508	3
3.1.1	Authentification du serveur MQTT depuis l'ESP8266 .	4
3.1.2	Authentification du client ESP8266 auprès du serveur MQTT	5
3.2	Par AES	6
3.2.1	Échange des valeurs entre les deux Raspberry Pi par LoRa & Utilisation du format JWT	6

1 Problématique

Le but de ce projet est de créer un réseau de capteurs (ESP8266) connectés par WiFi vers un concentrateur (un Raspberry Pi) où chaque capteur va exploiter un circuit dédié à la cryptographie sur courbe elliptique (un ATECC508) connecté à l'ESP8266 qui à travers Mongoose publie à intervalle régulier la donnée capturée sur un serveur MQTT sécurisé par l'utilisation de certificats et du protocole TLS, cette donnée sera ensuite chiffrée et transmise entre deux concentrateurs à travers le protocole LoRa. La Réalisation, les explications et les configurations qui ont permis de mettre en place les différentes étapes du projet sont disponibles sur le github accessible sur le lien suivant : https://github.com/Amadimk/UNILIM_TMC/

2 Mises en place des communications

2.1 WiFi et MQTT

2.1.1 De l'ESP8266 vers le Raspberry Pi

Pour permettre aux capteurs de joindre le concentrateur, il est nécessaire de mettre en place un access point wifi sur le Raspberry Pi en installant les paquets `hostapd` et `dnsmasq`. `Hostapd` est un package qui permet de créer un hotspot sans fil à l'aide d'un Raspberry Pi, et `dnsmasq` quand à lui permet de créer et lancer un serveur DNS et DHCP facilement.

Lors de la configuration du dns avec `dnsmasq` pour le point d'accès WiFi nous avons rajouté l'option permettant de faire la translation dns du nom symbolique associé au raspberry Pi : `mqtt.com` à l'adresse IP statique du point d'accès.

Toutes les configurations à effectuer pour la mise en place du point d'accès sont expliquées sur le **README** du github accompagnant ce rapport.

2.1.2 La connexion du client MQTT de l'ESP8266 vers le serveur MQTT du Raspberry Pi

Pour authentifier les clients MQTT nous avons créé et utilisé des certificats SSL. Nous avons tout d'abord commencé par l'installation du serveur MQTT avec `mosquitto` à l'aide des commandes ci-dessous :

- `sudo apt-get install mosquitto`
- `sudo apt-get install mosquitto-clients`

Ensuite nous avons générer des clés et certificats pour le serveur MQTT et pour le client ESP8266 signés par le même CA.

Les différents certificats et clés que nous avons générer sont :

- Clé et certificat CA :
 - `ecc.ca.key.pem`
 - `ecc.ca.cert.pem`
- Clé et certificat client ESP8266 :
 - `ecc.esp8266.key.pem`
 - `ecc.esp8266.cert.pem`
- Clé et certificat serveur Raspberry Pi :
 - `ecc.raspi.key.pem`
 - `ecc.raspi.cert.pem`

Les configurations à effectuer pour orchestrer l'échange TLS entre le composant ESP8266 miniu du circuit ATEC508 qui se comporte comme client MQTT et du serveur MQTT le Raspberry sont toutes expliquées sur le README du github associé.

2.2 Lora : entre les deux Raspberry

Pour le trafic loRa nous avons écrit un script python qui se connecte à l'aide de la librairie `paho-mqtt` au serveur MQTT en **subscribe** et capture continuellement les données publiées sur le topic `/esp8266` par le capteur puis les chiffre en AES-128 et l'encode avec JWT (Json Web Token). pour transmettre par la suite en LoRa ce token au second RaspBerry Pi. Les configurations et les codes sources des scripts pour chiffrer et déchiffrer les données sont expliqués et disponibles sur le github associé.

3 Chiffrement

3.1 Par courbe elliptique tirant parti de l'ATECC508

L'échange TLS entre le serveur MQTT et le capteur jouant le rôle de client tire parti des courbes elliptiques grâce notamment aux certificats que nous avons générer avec `openssl` pour les courbes elliptiques.

3.1.1 Authentification du serveur MQTT depuis l'ESP8266

la capture ci-dessous montre le trafic chiffré capturé entre le serveur MQTT et le client et nous pouvons voir clairement l'échange et l'authentification des certificats : **client hello** et **server hello**

Source	Destination	Protocol	Length	Info
192.168.4.2	192.168.4.1	TLSv1.2	113	Application Data
192.168.4.1	192.168.4.2	TLSv1.2	87	Application Data
192.168.4.2	192.168.4.1	TLSv1.2	113	Application Data
192.168.4.1	192.168.4.2	TLSv1.2	87	Application Data
192.168.4.2	192.168.4.1	TLSv1.2	189	Client Hello
192.168.4.1	192.168.4.2	TLSv1.2	1249	Server Hello, Certificate, Server Key Exchange, Certificate Request
192.168.4.2	192.168.4.1	TLSv1.2	710	Certificate, Client Key Exchange, Certificate Verify, Change Cipher
192.168.4.1	192.168.4.2	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
192.168.4.2	192.168.4.1	TLSv1.2	136	Application Data
192.168.4.1	192.168.4.2	TLSv1.2	85	Encrypted Alert
192.168.4.1	192.168.4.2	TLSv1.2	87	Application Data
192.168.4.2	192.168.4.1	TLSv1.2	113	Application Data
192.168.4.1	192.168.4.2	TLSv1.2	87	Application Data
192.168.4.2	192.168.4.1	TLSv1.2	113	Application Data
192.168.4.1	192.168.4.2	TLSv1.2	87	Application Data
192.168.4.2	192.168.4.1	TLSv1.2	113	Application Data
192.168.4.1	192.168.4.2	TLSv1.2	87	Application Data
192.168.4.2	192.168.4.1	TLSv1.2	113	Application Data

▶ Frame 1: 113 bytes on wire (904 bits), 113 bytes captured (904 bits)
▶ Ethernet II, Src: Espressi_2e:05:db (a0:20:a6:2e:05:db), Dst: Raspberr_3e:09:a8 (b8:27:eb:3e:09:a8)
▶ Internet Protocol Version 4, Src: 192.168.4.2, Dst: 192.168.4.1
▶ Transmission Control Protocol, Src Port: 57215, Dst Port: 8883, Seq: 1, Ack: 1, Len: 59
▶ Secure Sockets Layer

3.1.2 Authentification du client ESP8266 auprès du serveur MQTT

```
[Jan 8 21:59:53.397] connected with raspberryWifi01, channel 7
[Jan 8 21:59:53.397] dhcp client start...
[Jan 8 21:59:53.397] mgos_wifi.c:136      WiFi STA: Connected, BSSID b8:27:eb:3e:09:a8 ch 7 RSSI -53
[Jan 8 21:59:53.402] mgos_event.c:135     ev WiFi2 triggered 0 handlers
[Jan 8 21:59:53.407] mgos_net.c:90       WiFi STA: connected
[Jan 8 21:59:53.412] mgos_event.c:135     ev NET2 triggered 2 handlers
[Jan 8 21:59:54.334] ip:192.168.4.2,mask:255.255.255.0,gw:192.168.4.1
[Jan 8 21:59:54.334] mgos_event.c:135     ev WiFi3 triggered 0 handlers
[Jan 8 21:59:54.342] mgos_net.c:102      WiFi STA: ready, IP 192.168.4.2, GW 192.168.4.1, DNS 192.168.4.1
[Jan 8 21:59:54.347] mgos_mqtt.c:435     MQTT connecting to mqtt.com:8883
[Jan 8 21:59:54.356] mg_net.c:928      0x3ffef544 mqtt.com:8883 ecc.esp8266.cert.pem,ATCA:0,ecc.ca.cert.pem
[Jan 8 21:59:54.365] mgos_vfs.c:256     ecc.esp8266.cert.pem -> /ecc.esp8266.cert.pem pl 1 -> 1 0x3ffeff8c (refs 1)
[Jan 8 21:59:54.379] mgos_vfs.c:350     open ecc.esp8266.cert.pem 0x0 0x1b6 => 0x3ffeff8c ecc.esp8266.cert.pem 1 => 257 (refs 1)
[Jan 8 21:59:54.385] mgos_vfs.c:509     fstat 257 => 0x3ffeff8c:1 => 0 (size 660)
[Jan 8 21:59:54.391] mgos_vfs.c:509     fstat 257 => 0x3ffeff8c:1 => 0 (size 660)
[Jan 8 21:59:54.396] mgos_vfs.c:537     lseek 257 0 1 => 0x3ffeff8c:1 => 0
[Jan 8 21:59:54.402] mgos_vfs.c:537     lseek 257 0 0 => 0x3ffeff8c:1 => 0
[Jan 8 21:59:54.408] mgos_vfs.c:383     close 257 => 0x3ffeff8c:1 => 0 (refs 0)
[Jan 8 21:59:54.421] mgos_vfs.c:256     ecc.ca.cert.pem -> /ecc.ca.cert.pem pl 1 -> 1 0x3ffeff8c (refs 1)
[Jan 8 21:59:54.430] mgos_vfs.c:350     open ecc.ca.cert.pem 0x0 0x1b6 => 0x3ffeff8c ecc.ca.cert.pem 1 => 257 (refs 1)
[Jan 8 21:59:54.436] mgos_vfs.c:383     close 257 => 0x3ffeff8c:1 => 0 (refs 0)
[Jan 8 21:59:54.448] mg_net.c:928      0x3fff1ff4 udp://192.168.4.1:53 -,,-
[Jan 8 21:59:54.448] mg_net.c:796      0x3fff1ff4 udp://192.168.4.1:53
[Jan 8 21:59:54.453] mgos_event.c:135     ev NET3 triggered 2 handlers
[Jan 8 21:59:54.460] mg_net.c:811      0x3fff1ff4 udp://192.168.4.1:53 -> 0
[Jan 8 21:59:54.466] mgos_mongoose.c:66   New heap free LWM: 38952
[Jan 8 21:59:54.482] mg_net.c:796      0x3ffef544 tcp://192.168.4.1:8883
[Jan 8 21:59:54.488] mgos_mongoose.c:66   New heap free LWM: 38704
[Jan 8 21:59:54.496] mg_net.c:811      0x3ffef544 tcp://192.168.4.1:8883 -> 0
[Jan 8 21:59:54.520] mg_ssl_if_mbedtls.c:35 0x3ffef544 ciphersuite: TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256
[Jan 8 21:59:54.533] mgos_vfs.c:256     ecc.ca.cert.pem -> /ecc.ca.cert.pem pl 1 -> 1 0x3ffeff8c (refs 1)
[Jan 8 21:59:54.543] mgos_vfs.c:350     open ecc.ca.cert.pem 0x0 0x1b6 => 0x3ffeff8c ecc.ca.cert.pem 1 => 257 (refs 1)
[Jan 8 21:59:54.549] mgos_vfs.c:509     fstat 257 => 0x3ffeff8c:1 => 0 (size 635)
[Jan 8 21:59:54.682] ATCA ECDSA verify ok, verified
[Jan 8 21:59:54.688] mgos_vfs.c:383     close 257 => 0x3ffeff8c:1 => 0 (refs 0)
[Jan 8 21:59:54.818] ATCA ECDSA verify ok, verified
```

```
[Jan 8 21:59:55.198] mgos_mongoose.c:66   New heap free LWM: 31220
[Jan 8 21:59:55.198] mgos_mqtt.c:141     MQTT TCP connect ok (0)
[Jan 8 21:59:55.220] mgos_mqtt.c:135     MQTT event: 202
[Jan 8 21:59:55.220] mgos_mqtt.c:185     MQTT CONNACK 0
[Jan 8 21:59:55.224] mgos_event.c:135     ev MOS4 triggered 0 handlers
[Jan 8 21:59:56.480] mgos_mqtt.c:529     Publishing to /esp8266 @ 1 (16): [Hello im esp8266]
[Jan 8 21:59:56.493] mgos_mqtt.c:135     MQTT event: 204
[Jan 8 21:59:58.480] mgos_mqtt.c:529     Publishing to /esp8266 @ 1 (16): [Hello im esp8266]
[Jan 8 21:59:58.494] mgos_mqtt.c:135     MQTT event: 204
[Jan 8 22:00:00.480] mgos_mqtt.c:529     Publishing to /esp8266 @ 1 (16): [Hello im esp8266]
[Jan 8 22:00:00.493] mgos_mqtt.c:135     MQTT event: 204
[Jan 8 22:00:02.480] mgos_mqtt.c:529     Publishing to /esp8266 @ 1 (16): [Hello im esp8266]
[Jan 8 22:00:02.493] mgos_mqtt.c:135     MQTT event: 204
[Jan 8 22:00:03.370] pm open,type:0 0
[Jan 8 22:00:04.480] mgos_mqtt.c:529     Publishing to /esp8266 @ 1 (16): [Hello im esp8266]
[Jan 8 22:00:04.494] mgos_mqtt.c:135     MQTT event: 204
[Jan 8 22:00:06.480] mgos_mqtt.c:529     Publishing to /esp8266 @ 1 (16): [Hello im esp8266]
[Jan 8 22:00:06.494] mgos_mqtt.c:135     MQTT event: 204
[Jan 8 22:00:08.480] mgos_mqtt.c:529     Publishing to /esp8266 @ 1 (16): [Hello im esp8266]
[Jan 8 22:00:08.493] mgos_mqtt.c:135     MQTT event: 204
[Jan 8 22:00:10.480] mgos_mqtt.c:529     Publishing to /esp8266 @ 1 (16): [Hello im esp8266]
[Jan 8 22:00:10.493] mgos_mqtt.c:135     MQTT event: 204
[Jan 8 22:00:12.480] mgos_mqtt.c:529     Publishing to /esp8266 @ 1 (16): [Hello im esp8266]
[Jan 8 22:00:12.493] mgos_mqtt.c:135     MQTT event: 204
[Jan 8 22:00:14.480] mgos_mqtt.c:529     Publishing to /esp8266 @ 1 (16): [Hello im esp8266]
[Jan 8 22:00:14.494] mgos_mqtt.c:135     MQTT event: 204
[Jan 8 22:00:16.480] mgos_mqtt.c:529     Publishing to /esp8266 @ 1 (16): [Hello im esp8266]
[Jan 8 22:00:16.493] mgos_mqtt.c:135     MQTT event: 204
[Jan 8 22:00:18.480] mgos_mqtt.c:529     Publishing to /esp8266 @ 1 (16): [Hello im esp8266]
[Jan 8 22:00:18.493] mgos_mqtt.c:135     MQTT event: 204
[Jan 8 22:00:20.481] mgos_mqtt.c:529     Publishing to /esp8266 @ 1 (16): [Hello im esp8266]
[Jan 8 22:00:20.493] mgos_mqtt.c:135     MQTT event: 204
```

3.2 Par AES

3.2.1 Échange des valeurs entre les deux Raspberry Pi par LoRa & Utilisation du format JWT

```
pi@mqtt:~/RadioHead/examples/raspi/rf95 $ sudo python3 mqtt_client.py
Receive from topic /esp8266 ==> Hello im esp8266
Sending 03 bytes LORA to node #1 => eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJkYXRhIjoiaWdhVWk1JZUzUXB2Zms1QXxc3VaZz09In0.SuRvemI5_fd9Ep4kS_hUX660V5NGyWYwWoepWBdWoTI
Receive from topic /esp8266 ==> Hello im esp8266
Sending 03 bytes LORA to node #1 => eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJkYXRhIjoiaWdhVWk1JZUzUXB2Zms1QXxc3VaZz09In0.SuRvemI5_fd9Ep4kS_hUX660V5NGyWYwWoepWBdWoTI
Receive from topic /esp8266 ==> Hello im esp8266
Sending 03 bytes LORA to node #1 => eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJkYXRhIjoiaWdhVWk1JZUzUXB2Zms1QXxc3VaZz09In0.SuRvemI5_fd9Ep4kS_hUX660V5NGyWYwWoepWBdWoTI
Receive from topic /esp8266 ==> Hello im esp8266
```

```
jalix@sarra: ~/Bureau/TMC_projet x jalix@sarra: ~/Bureau/TMC_projet x pi@r
pi@raspberrypi:~/LoRa/RadioHead/examples/raspi/rf95 $ sudo ./rf95_server
rf95_server
RF95 CS=GPI025, IRQ=GPI04, RST=GPI017, LED=GPI0255 OK NodeID=1 @ 868.00MHz
Listening packet...
Packet[129] #10 => #1 -22dB:
Receive JWT token : eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJkYXRhIjoiaWdhVWk1JZUzUXB2Zms1QXxc3VaZz09In0.SuRvemI5_fd9Ep4kS_hUX660V5NGyWYwWoepWBdWoTI
Decoded JWT data (encoded base64 + AES ) : X8UZIIEIsQpvfk5ALqsuZg==
Decoded AES data : Hello im esp8266
Packet[129] #10 => #1 -22dB:
Receive JWT token : eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJkYXRhIjoiaWdhVWk1JZUzUXB2Zms1QXxc3VaZz09In0.SuRvemI5_fd9Ep4kS_hUX660V5NGyWYwWoepWBdWoTI
Decoded JWT data (encoded base64 + AES ) : X8UZIIEIsQpvfk5ALqsuZg==
Decoded AES data : Hello im esp8266
Packet[129] #10 => #1 -22dB:
```

Enfin, toutes les configurations et codes sources qui ont permis d'orchestrer et de mettre en place les communications chiffrés entre les capteurs et le concentrateur et du trafic LoRa entre les deux concentrateurs ainsi qu'une vidéo **démo** sont disponibles et expliqués sur le **github** du projet. Par ailleurs la plupart des configurations que nous avons effectués sont totalement ou partiellement une reprise des explications des TP de L'UE TMC de Mr Bonnefoi que nous remercions pour ce TP plein d'enseignement sur les objets connectés.