Projet TMC

Réalisé par :
Amadou Ourry Diallo
Wajdi KILANI
Moetaz RABAI

Table des matières

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 capturé sur un serveur MQTT securisé par l'utilisation de certificats et du protocole TLS, cette donnée sera ensuite chiffré et transmis 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 a 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 rajouter 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 crée 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 minu 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 écris 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 chiffres 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.

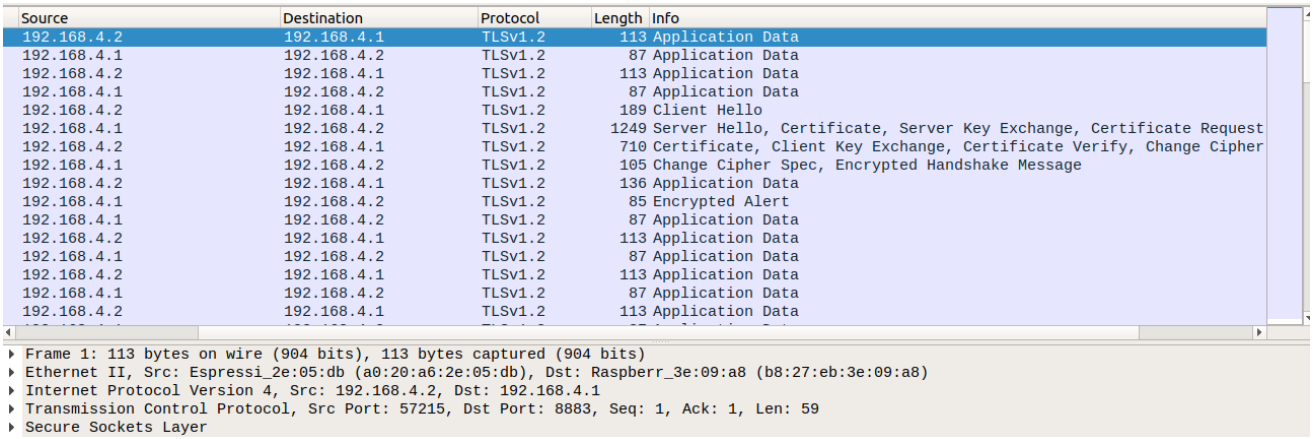
3 Chiffrement

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.

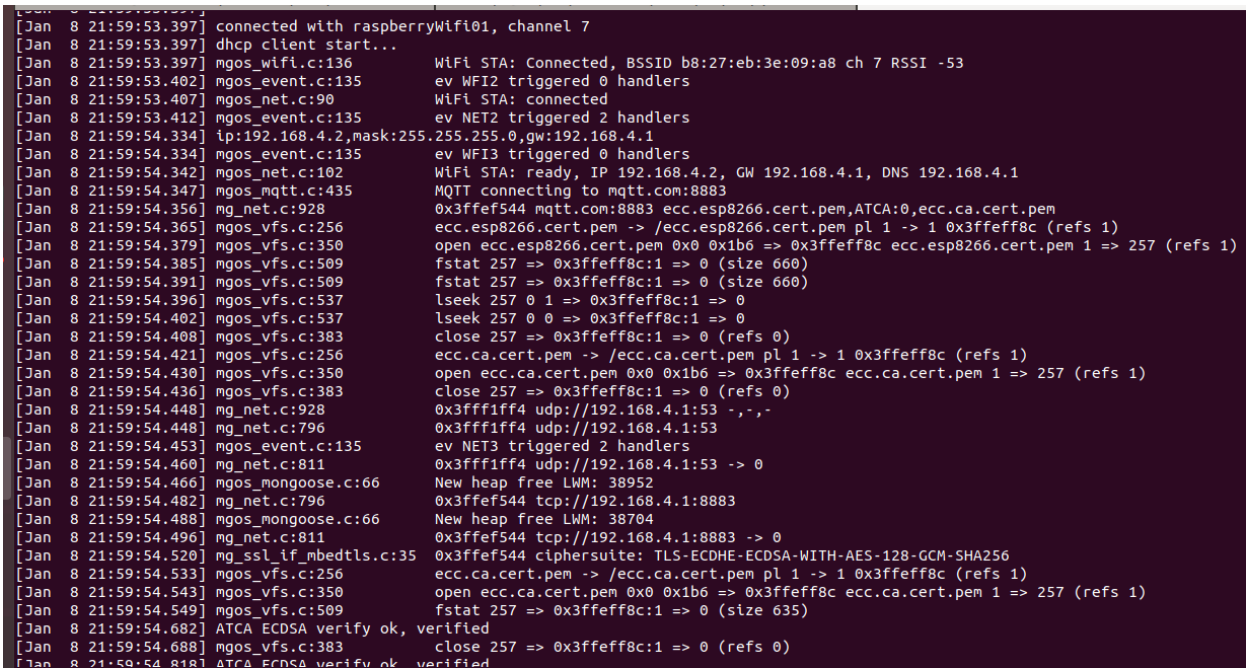
3.1.1 Authentification du serveur MQTT depuis l'ESP8266

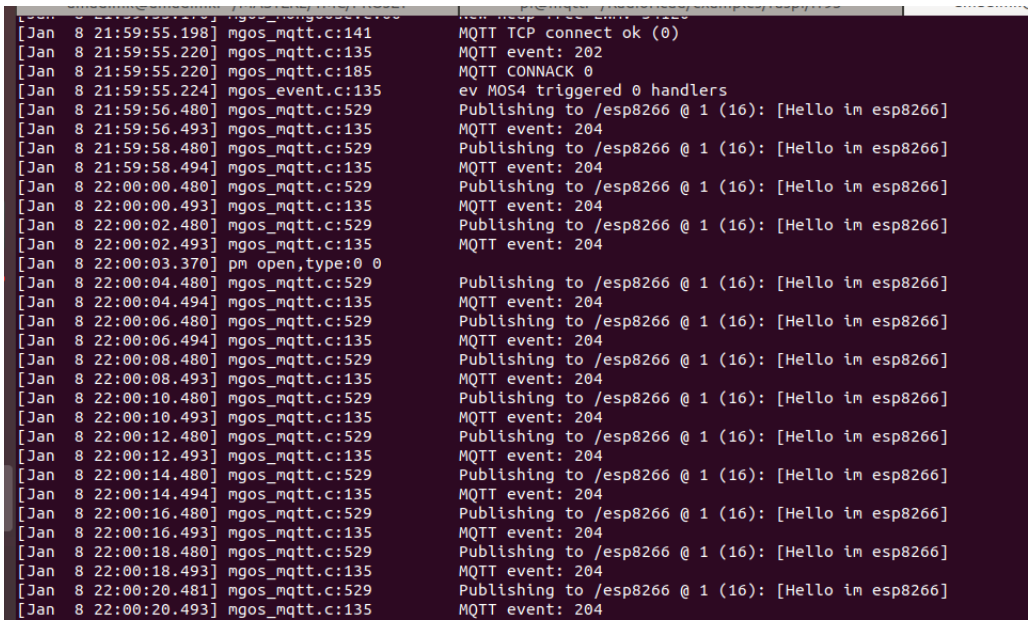
la capture ci-dessous montre le trafic chiffré capturé entre le serveur MQTT et le client et nous pouvons voir clairement l'échange et l'authentification des certificats : **client hello** et **server hello**



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

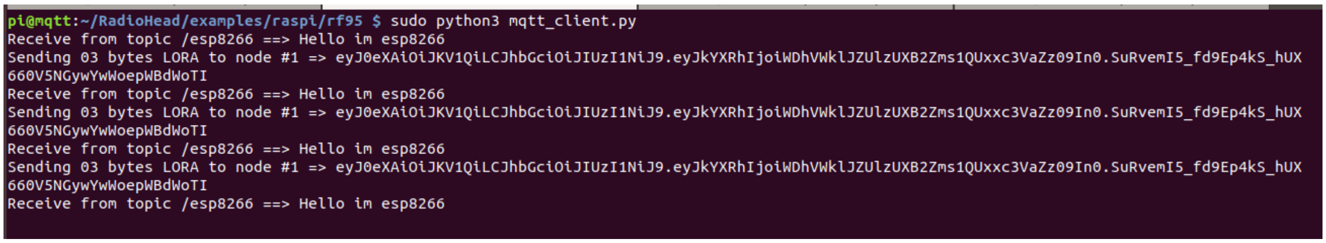
3.1.2 Authentification du client ESP8266 auprès du serveur MQTT





3.2 Par AES

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



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 pluplart des configurations que nous avons effectués sont totalement ou partiellement une reprise des explications des TPs de L'UE TMC de Mr Bonnefoi que nous remercions pour ce TP plein d'enseignement sur les objets connectés.