

IP broker : 192.168.56.101

IP client : 192.168.56.102

Pour vérifier la publication et la souscription, utilisez les commandes suivantes :  
mosquitto\_pub -h ip\_du\_broker -m hello -t "world"

```
1654094034: New connection from 192.168.56.102 on port 1883.
1654094034: New client connected from 192.168.56.102 as mosqpub|1705-debian (c1, k60).
1654094034: No will message specified.
1654094034: Sending CONNACK to mosqpub|1705-debian (0, 0)
1654094034: Received PUBLISH from mosqpub|1705-debian (d0, q0, r0, m0, 'world', ... (5 bytes))
1654094034: Received DISCONNECT from mosqpub|1705-debian
1654094034: Client mosqpub|1705-debian disconnected.
```

mosquitto\_sub -h ip\_du\_broker -t "#" -v

```
1654095227: New connection from 192.168.56.102 on port 1883.
1654095227: New client connected from 192.168.56.102 as mosqsub|1764-debian (c1, k60).
1654095227: No will message specified.
1654095227: Sending CONNACK to mosqsub|1764-debian (0, 0)
1654095227: Received SUBSCRIBE from mosqsub|1764-debian
1654095227:      # (QoS 0)
1654095227: mosqsub|1764-debian 0 #
1654095227: Sending SUBACK to mosqsub|1764-debian
```

La publication et la souscription fonctionnnent correctement

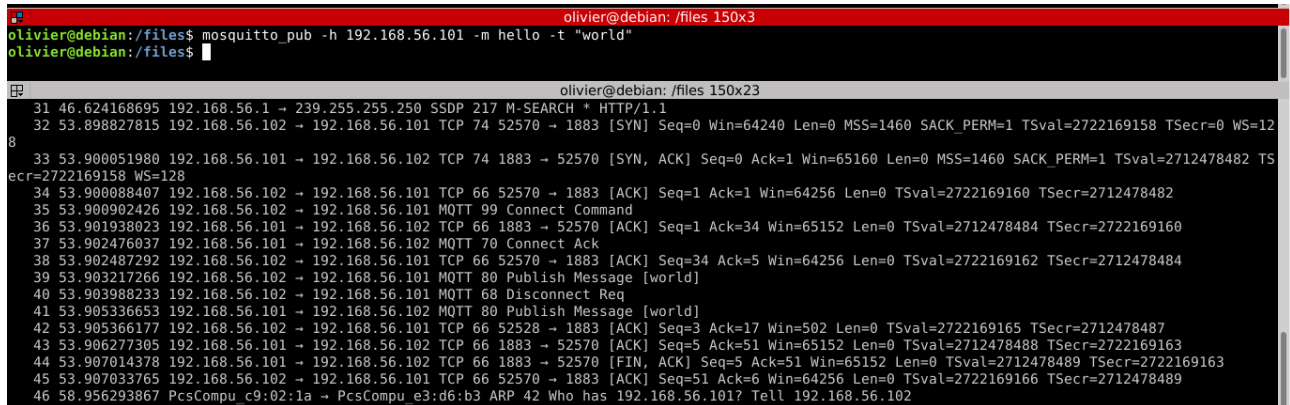
Connexion de mosquitto\_client A à son broker :

```
1654095368: New connection from ::1 on port 1883.
1654095368: New client connected from ::1 as mosqsub|1327-debian2 (c1, k60).
1654095368: No will message specified.
1654095368: Sending CONNACK to mosqsub|1327-debian2 (0, 0)
1654095368: Received SUBSCRIBE from mosqsub|1327-debian2
1654095368:      # (QoS 0)
1654095368: mosqsub|1327-debian2 0 #
1654095368: Sending SUBACK to mosqsub|1327-debian2
```

```
olivier@debian2: ~ 109x7
olivier@debian2:~$ echo Je suis A
Je suis A
olivier@debian2:~$ mosquitto_sub -h localhost -t "#" -v
```

Vous allez à présent observer ce qui se passe sur le réseau. Lancez tshark et publiez un message sur le broker.

Produisez une capture d'écran de ce que tshark a collecté.



```
olivier@debian: /files 150x3
olivier@debian:/files$ mosquitto_pub -h 192.168.56.101 -m hello -t "world"
olivier@debian:/files$

olivier@debian: /files 150x23
31 46.624168695 192.168.56.1 → 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1
32 53.898827815 192.168.56.102 → 192.168.56.101 TCP 74 52570 → 1883 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2722169158 TSecr=0 WS=12
33 53.900051980 192.168.56.101 → 192.168.56.102 TCP 74 1883 → 52570 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2712478482 TSecr=2722169158 WS=128
34 53.900088407 192.168.56.102 → 192.168.56.101 TCP 66 52570 → 1883 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2722169160 TSecr=2712478482
35 53.900902426 192.168.56.102 → 192.168.56.101 MQTT 99 Connect Command
36 53.901938023 192.168.56.101 → 192.168.56.102 TCP 66 1883 → 52570 [ACK] Seq=1 Ack=34 Win=65152 Len=0 TSval=2712478484 TSecr=2722169160
37 53.902476037 192.168.56.101 → 192.168.56.102 MQTT 70 Connect Ack
38 53.902487292 192.168.56.102 → 192.168.56.101 TCP 66 52570 → 1883 [ACK] Seq=34 Ack=5 Win=64256 Len=0 TSval=2722169162 TSecr=2712478484
39 53.903217266 192.168.56.102 → 192.168.56.101 MQTT 80 Publish Message [world]
40 53.903988233 192.168.56.102 → 192.168.56.101 MQTT 68 Disconnect Req
41 53.905336653 192.168.56.101 → 192.168.56.102 MQTT 80 Publish Message [world]
42 53.905366177 192.168.56.102 → 192.168.56.101 TCP 66 52528 → 1883 [ACK] Seq=3 Ack=17 Win=502 Len=0 TSval=2722169165 TSecr=2712478487
43 53.906277305 192.168.56.101 → 192.168.56.102 TCP 66 1883 → 52570 [ACK] Seq=5 Ack=51 Win=65152 Len=0 TSval=2712478488 TSecr=2722169163
44 53.907014378 192.168.56.101 → 192.168.56.102 TCP 66 1883 → 52570 [FIN, ACK] Seq=5 Ack=51 Win=65152 Len=0 TSval=2712478489 TSecr=2722169163
45 53.907033765 192.168.56.102 → 192.168.56.101 TCP 66 52570 → 1883 [ACK] Seq=51 Ack=6 Win=64256 Len=0 TSval=2722169166 TSecr=2712478489
46 58.956293867 PcsCompu_c9:02:1a → PcsCompu_e3:d6:b3 ARP 42 Who has 192.168.56.101? Tell 192.168.56.102
```

Figure : Capture écran tshark

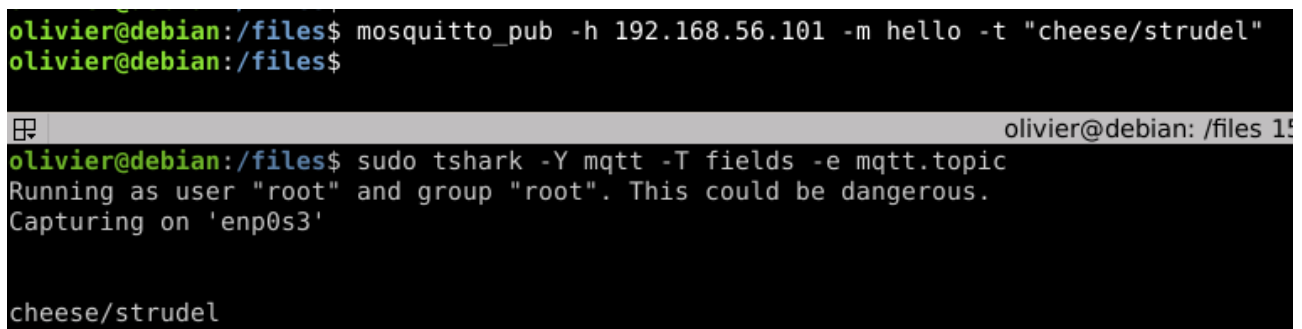
Le TCP Handshake est-il présent dans votre capture ?

Oui le TCP HandShake est visible, on observe les échanges SYN, SYN-ACK et ACK avant le début de l'échange MQTT et les échanges de fin d'échanges (FIN).

Le lot de données produit par tshark n'est pas forcément le plus lisible. Filtrons :  
tshark -Y mqtt -T fields -e mqtt.topic

Publiez un nouveau message sur le topic « cheese/strudel ». Que vous retourne tshark ?

La commande : `mosquitto_pub -h 192.168.56.101 -m hello -t "cheese/strudel"`  
Tshark retourne le nom du topic qu'il a extrait du message MQTT



```
olivier@debian:/files$ mosquitto_pub -h 192.168.56.101 -m hello -t "cheese/strudel"
olivier@debian:/files$

olivier@debian: /files 150x3
olivier@debian:/files$ sudo tshark -Y mqtt -T fields -e mqtt.topic
Running as user "root" and group "root". This could be dangerous.
Capturing on 'enp0s3'

cheese/strudel
```

Figure : envoi topic « cheese/strudel »

Conclusion quant à la confidentialité de l'échange ?

Cette échange n'est pas confidentiel, les messages transitent en clair entre les hôtes.

# MQTTS

Il est possible d'exiger l'authentification des clients sur MQTT par un couple login/mdp ou via des certificats. Vous allez mettre en oeuvre l'authentification basée sur les certificats. Pour ceci, vous allez générer les certificats, les distribuer et les utiliser.

Rappelez le lien entre les certificats et le chiffrement asymétrique.

Un certificat contient une clé publique, et il est signé par la clé privée de la CA (Certificate Authority). Le client a confiance en la CA et peut grâce à la clé publique de la CA contrôler le certificat du serveur pour s'assurer que celui-ci est bien celui qu'il prétend.

Le client communique alors un secret commun au serveur chiffré grâce à la clé publique du serveur pour constituer la clé symétrique qui servira à l'échange de données.

Les couples clés privé/publics utilisés pour sécuriser les certificats, sont générés au moyen d'un algorithme de chiffrement asymétrique comme RSA. Des données chiffrées avec une clé ne peuvent être déchiffrées qu'avec la clé associée.

Rappelez le processus de génération d'un certificat.

Un certificat est généré en plusieurs étapes :

- création d'un couple clé privée et clé publique au moyen d'un algorithme de chiffrement asymétrique
- saisie des informations du certificat (adresse site web, administrateur, pays, etc.)
- calcul du hash des données du certificat
- signature du certificat : chiffrement du hash par la clé privée de la CA

Pour générer vos certificats, vous devez créer votre propre autorité. Quelle est la version installée ?

La version d'OpenSSL installée est : OpenSSL 1.1.1n 15 Mar 2022

Quel algorithme de chiffrement est utilisé ?

L'algorithme de chiffrement utilisé est RSA.

```
openssl x509 -req -in broker.csr -CA ../ca/ca.crt -CAkey ../ca/ca.key  
-CAcreateserial -out broker.crt -days 100
```

Quelle est la durée de validité du certificat ?

La durée de validité du certificat est de 100 jours à compter de la date de création.

Pour activer la connexion via TLS, il faut que le broker ait connaissance des fichiers suivants : le certificat du CA, la clé privée du broker et le certificat du broker.

Comment choisissez-vous de transférer ces fichiers au broker ?

On choisit de les transférer en scp, il garanti la confidentialité et l'intégrité des données.

```
olivier@debian:~/TP/certs/broker$ scp broker.crt olivier@192.168.56.101:/etc/mosquitto/mytls/broker.crt
The authenticity of host '192.168.56.101 (192.168.56.101)' can't be established.
ECDSA key fingerprint is SHA256:HaQwTmj07bDV034NVX0DzwcFLDQl8l0fkeiQCBUU1GI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.56.101' (ECDSA) to the list of known hosts.
olivier@192.168.56.101's password:
broker.crt                                                                                               100% 1058    653.9KB/s   00:00
olivier@debian:~/TP/certs/broker$ scp broker.key olivier@192.168.56.101:/etc/mosquitto/mytls/broker.key
olivier@192.168.56.101's password:
broker.key                                                                                               100% 1679    717.0KB/s   00:00
olivier@debian:~/TP/certs/broker$ sudo scp ../ca/ca.crt olivier@192.168.56.101:/etc/mosquitto/mytls/
olivier@192.168.56.101's password:
ca.crt                                                                                                   100% 1180    686.9KB/s   00:00
```

Figures : envoi des certificat et de la clé privée par SCP

Modifier le fichier de configuration de mosquitto

```
olivier@debian2: /etc/mosquitto/mytls 109x12
listener 8883
protocol mqtt
allow_anonymous true
cafile /etc/mosquitto/ca_certificates/ca.crt
certfile /etc/mosquitto/certs/broker.crt
keyfile /etc/mosquitto/mytls/broker.key
require_certificate true
~
~
~
~
"../conf.d/confTP.conf" 7L, 202C written
```

Figure :Activez la connexion via TLS dans le serveur : dans le fichier confTP.conf

Sur quels ports le serveur écoute-t-il ?

Le serveur écoute sur le port 8883

## Commande mosquitto\_pub avec MQTTS

```
mosquitto_pub -p 8883 --cafile /home/olivier/TP/certs/ca/ca.crt --cert  
home/olivier/TP/certs/client/client.crt --key client.key -h 192.168.56.101 -m hello -t "world"
```

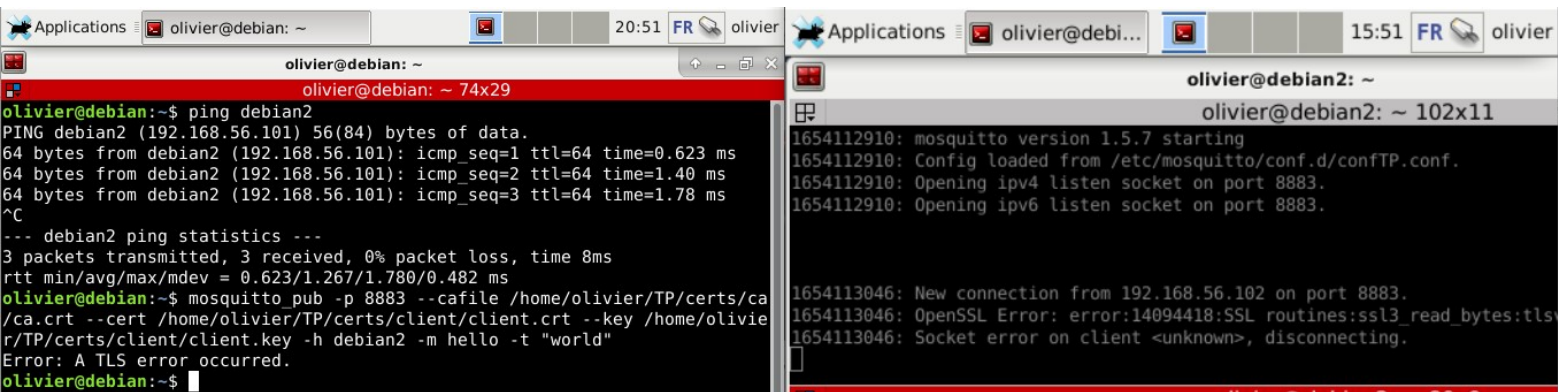


Figure : Erreur rencontrée en MQTTS

Je rencontre l'erreur « A TLS error occurred », erreur affichée par le client. Le serveur affiche l'erreur : « OpenSSL Error: error:14094418:SSL routines:ssl3\_read\_bytes:tlsv1 alert unknown ca »

## Analyse des certificats suite à l'erreur

Certificat du broker :

```
olivier@debian:~/TP/certs/broker$ openssl x509 -in broker.crt -text -noout
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            72:e9:8e:4e:7b:c9:fc:0c:73:ed:19:5d:40:9e:05:07:18:68:05:2a
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = FR, ST = France, O = ISIS
        Validity
            Not Before: Jun  1 15:17:29 2022 GMT
            Not After : Sep  9 15:17:29 2022 GMT
        Subject: C = FR, ST = France, O = ISIS
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public-Key: (2048 bit)
            Modulus:
                00:c6:44:57:1b:38:d9:04:a3:2e:ba:9e:d9:f6:d6:
                6d:8f:e2:1c:58:a9:80:7a:62:07:3d:36:73:04:02:
                54:14:5a:a0:63:2d:08:b7:de:51:d2:a0:0f:7c:44:
                45:8d:b5:24:cc:a1:f7:43:5a:8a:ae:e7:f3:00:6d:
                f2:af:96:7b:76:b1:da:33:5b:37:62:59:c6:f2:e1:
                ea:a4:88:0f:54:0f:43:32:c1:49:9f:31:35:8b:ca:
                31:d2:c3:dd:a5:07:fd:e3:12:08:71:74:b9:38:e3:
                0b:f3:42:c9:06:92:76:e7:c1:8b:f7:9b:c6:a7:23:
                f1:fe:9f:6d:4b:cd:75:97:d6:6c:a6:66:51:f6:77:
                d3:4b:46:4d:55:fe:9b:8d:ca:b1:bf:76:83:5a:52:
                df:f0:66:a3:3b:c2:cc:9d:ae:e4:8c:70:44:01:52:
                b8:ad:c2:1d:45:f5:bc:af:b4:88:6f:19:74:d5:c2:
```

Analyse du certificat du client :

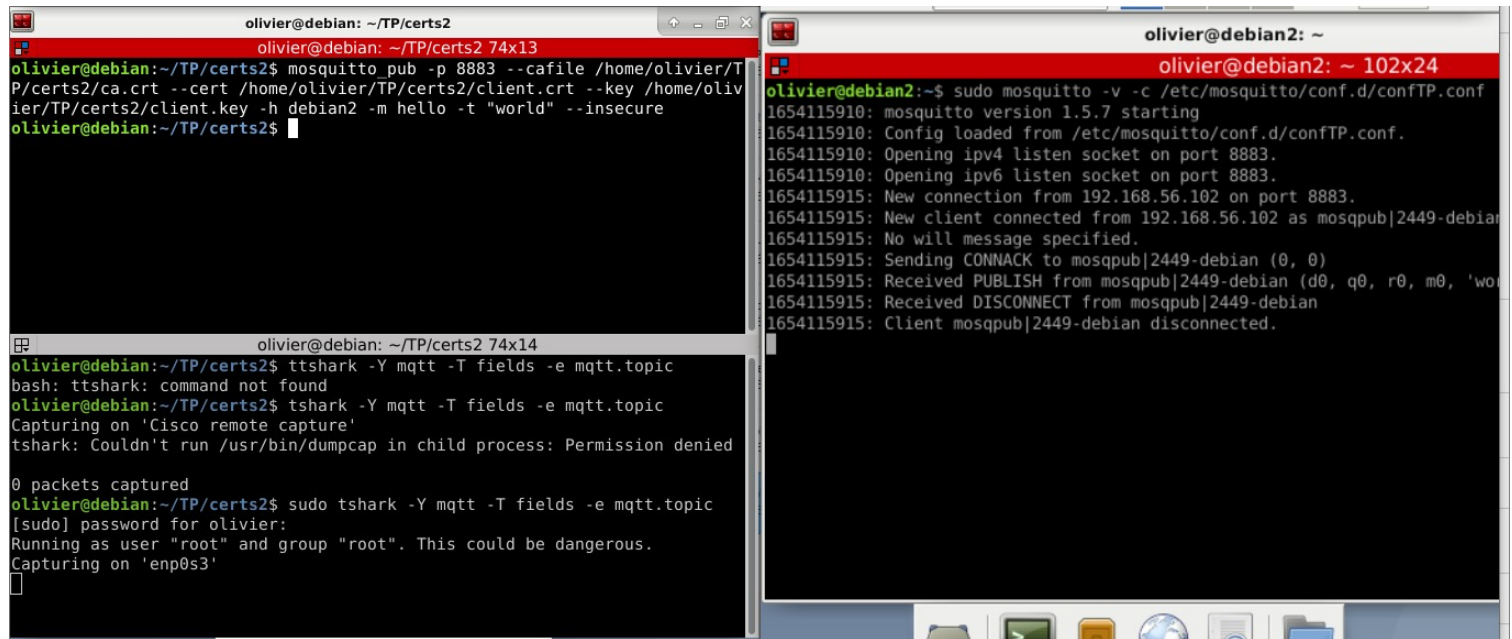
```
olivier@debian:~/TP/certs/client$ openssl x509 -in client.crt -text -noout
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            72:e9:8e:4e:7b:c9:fc:0c:73:ed:19:5d:40:9e:05:07:18:68:05:2b
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = FR, ST = France, O = ISIS
        Validity
            Not Before: Jun  1 15:20:29 2022 GMT
            Not After : Sep  9 15:20:29 2022 GMT
        Subject: C = FR, ST = France, O = Internet Widgits Pty Ltd
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public-Key: (2048 bit)
            Modulus:
                00:dc:ca:f0:b2:c5:38:e6:c1:69:25:a1:23:26:2f:
                3f:03:e5:28:88:8d:e1:08:e9:34:22:d9:5e:b9:0c:
                7c:24:99:6f:90:7d:bf:af:83:7b:d3:65:0e:78:c9:
                47:65:f4:ca:eb:66:9a:da:a8:30:2e:09:b0:7e:25:
                92:a7:a4:2f:98:5c:c7:70:b5:1b:6f:31:63:03:92:
                fc:75:c3:e3:6b:41:2d:0c:12:e7:01:e6:62:ff:d3:
                ef:d1:87:57:3d:7a:8f:52:7e:0b:7f:e5:e6:65:63:
                37:76:e0:bc:09:eb:70:00:26:a6:21:b4:24:93:67:
                ba:09:8e:51:65:0b:0f:0b:a0:59:e3:e6:3f:fb:1f:
                31:0d:0d:be:0f:10:d2:91:82:b4:32:b3:f1:42:f2:
                da:bb:2d:0b:2e:8b:7b:32:77:b5:3a:a0:6c:fd:9a:
                8b:4a:58:a7:cc:74:a7:4f:ed:80:3f:42:a9:9e:cf:
                ee:7d:94:eb:19:56:0e:6a:59:ed:7a:5f:2b:97:e7:
```

Solutions testées :

- régénérer le certificat client avec le même nom que la machine client (« debian »)
- ajouter le certificat CA au gestionnaire de certificat de Debian
- solutions de ce flux : <https://github.com/eclipse/mosquitto/issues/689> (régénérer complètement les certificats CA, broker et client)

Ces solutions n'ont pas fonctionné. Pour terminer le TP j'ai ajouté l'option `--insecure` au client qui évite la vérification des certificats par le CA.

Résultat :



The image shows three terminal windows from a Linux desktop environment. The top-left window, titled 'olivier@debian: ~/TP/certs2', shows the command to start Mosquitto as a publisher: `mosquitto pub -p 8883 --cafile /home/olivier/TP/certs2/ca.crt --cert /home/olivier/TP/certs2/client.crt --key /home/olivier/TP/certs2/client.key -h debian2 -m hello -t "world" --insecure`. The top-right window, titled 'olivier@debian2: ~', shows Mosquitto running as a subscriber: `sudo mosquitto -v -c /etc/mosquitto/conf.d/confTP.conf`. It displays logs for Mosquitto version 1.5.7 starting, loading the config, opening listen sockets on ports 8883, and receiving a PUBLISH message from 'mosqpub|2449-debian' with topic 'world'. The bottom-left window, titled 'olivier@debian: ~/TP/certs2 74x14', shows attempts to use 'ttshark' (not found) and 'tshark' (permission denied) to sniff traffic on 'enp0s3'. It then shows 'sudo tshark' being run, resulting in 0 packets captured.

```
olivier@debian: ~/TP/certs2
olivier@debian: ~/TP/certs2 74x13
olivier@debian:~/TP/certs2$ mosquitto pub -p 8883 --cafile /home/olivier/TP/certs2/ca.crt --cert /home/olivier/TP/certs2/client.crt --key /home/olivier/TP/certs2/client.key -h debian2 -m hello -t "world" --insecure
olivier@debian:~/TP/certs2$

olivier@debian: ~/TP/certs2 74x14
olivier@debian:~/TP/certs2$ ttshark -Y mqtt -T fields -e mqtt.topic
bash: ttshark: command not found
olivier@debian:~/TP/certs2$ tshark -Y mqtt -T fields -e mqtt.topic
Capturing on 'Cisco remote capture'
tshark: Couldn't run /usr/bin/dumpcap in child process: Permission denied

0 packets captured
olivier@debian:~/TP/certs2$ sudo tshark -Y mqtt -T fields -e mqtt.topic
[sudo] password for olivier:
Running as user "root" and group "root". This could be dangerous.
Capturing on 'enp0s3'

olivier@debian2: ~
olivier@debian2: ~ 102x24
olivier@debian2:~$ sudo mosquitto -v -c /etc/mosquitto/conf.d/confTP.conf
1654115910: mosquitto version 1.5.7 starting
1654115910: Config loaded from /etc/mosquitto/conf.d/confTP.conf.
1654115910: Opening ipv4 listen socket on port 8883.
1654115910: Opening ipv6 listen socket on port 8883.
1654115915: New connection from 192.168.56.102 on port 8883.
1654115915: New client connected from 192.168.56.102 as mosqpub|2449-debian
1654115915: No will message specified.
1654115915: Sending CONNACK to mosqpub|2449-debian (0, 0)
1654115915: Received PUBLISH from mosqpub|2449-debian (d0, q0, r0, m0, 'world')
1654115915: Received DISCONNECT from mosqpub|2449-debian
1654115915: Client mosqpub|2449-debian disconnected.
```

Figure : Envoi des données en MQTTS et sniffing avec tshark

Sur la fenetre tshark en bas à gauche on voit que le filtre ne permet plus d'obtenir les topics envoyés alors que le client envoie des topics et le serveur est capable de les recevoir. Le chiffrement rend impossible la lecture des messages MQTT par les intermédiaires, il garantit la confidentialité (et l'authentification si le CA marchait bien...).