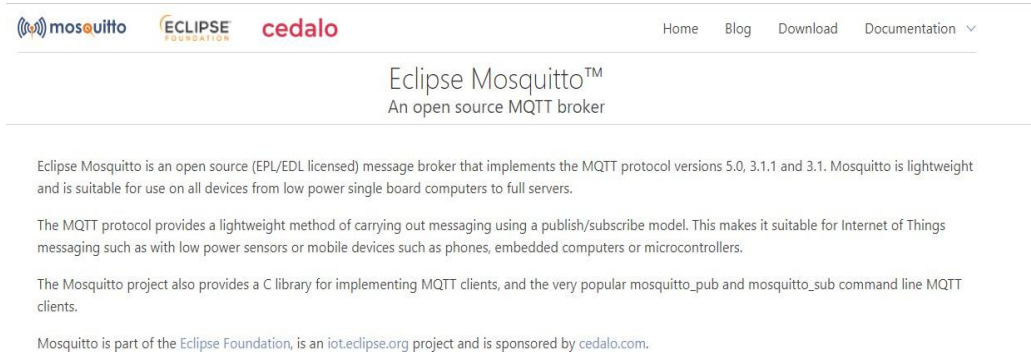


..: Mosquitto MQTT Broker y ESP32/ESP8266:..

Autor

Freddy Alcarazo | @surflaweb | @alcarazolabs

19 de febrero del 2025 – Chiclayo Perú.



Objetivo:

Instalar mosquitto y ejecutar ejemplo en ESP32 o ESP8266.

- <https://mosquitto.org/>

Video Tutorial:

- <https://youtu.be/Mn5HzdYk0>
- <https://youtu.be/yLYZafxgP-A>

Instalar Mosquito:

```
$ sudo apt-get update
```

```
$ sudo apt install mosquitto
```

ver donde está ubicado mosquitto:

```
$ ps -ef | grep mosquitto
```

ver estado de mosquitto:

```
$ sudo systemctl status mosquitto
```

reiniciar mosquitto:

```
$ sudo systemctl restart mosquitto
```

ver puertos que esta usando mosquitto:

```
$ sudo netstat -tunlp
```

Permitir conexiones anónimas/remotas a mosquitto broker:

Abrir el archivo default.conf de mosquitto y agregar lo siguiente:

```
sudo nano /etc/mosquitto/mosquitto.conf
```

```
allow_anonymous true  
listener 1883 0.0.0.0
```

Más configuraciones ver el siguiente link:

<https://hostadvice.com/how-to/how-to-install-and-configure-mosquitto-mqtt-on-your-ubuntu-18-04-server/>

Luego de haber agregado esto es necesario reiniciar mosquitto:

```
$ sudo systemctl restart mosquitto
```

Si aún así el cliente no puede conectarse debemos de abrir el puerto 1883 en el firewall:

Pasos:

1. Aperturar firewall:

Ver estado del firewall:

```
$ sudo ufw status verbose
```

Activar firewall:

```
$ sudo ufw enable
```

Abrir puerto:

```
$ sudo ufw allow 1883/tcp
```

Ver estado otra vez:

```
$ sudo ufw status verbose
```

Más configuraciones:

<https://linuxways.net/ubuntu/how-to-open-a-port-on-ubuntu-20-04/>

Instalar cliente en ubuntu para enviar mensajes al broker:

```
$ sudo apt install mosquitto-clients
```

Código de ejemplo para el ESP8266:

- <https://github.com/alcarazolabs/surflaweb-scripts/blob/main/Mosquitto-ClienteESP8266-Ejemplo.ino>

Enviar mensaje de un topico desde la consola:

```
mosquitto_pub -h localhost -t /casa/foco -m 'ON'
```

Suscribirse a todos los topicos en la consola

```
mosquitto_sub -h localhost -t \# -d
```

Suscribirse a un solo topico en la consola

```
mosquitto_sub -h localhost -t /project/audio -d
```

Más detalles sobre suscripciones:

http://www.steves-internet-guide.com/mosquitto_pub-sub-clients/

*** Nota: Si están en un servidor de producción abrir puerto en el firewall:**

```
$ sudo ufw allow 1883
```

* Si usa un VPS como DigitalOcean abrir puerto desde el firewall en el panel de control del droplet.

Fuentes:

- <https://hostadvice.com/how-to/how-to-install-and-configure-mosquitto-mqtt-on-your-ubuntu-18-04-server/>
- http://manpages.ubuntu.com/manpages/trusty/man1/mosquitto_pub.1.html

Configuración de Usuario y Contraseña:

Para configurar Mosquitto para que requiera autenticación con nombre de usuario y contraseña en cada conexión, sigue estos pasos:

Abre una terminal y ejecuta el siguiente comando para crear un nuevo archivo de contraseñas y agregar un usuario:

Agregar usuario:

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd nombre_de_usuario
```

- Se te pedirá que ingreses una contraseña para el usuario.
- Si deseas agregar más usuarios, puedes usar el siguiente comando (sin la opción -c):

```
sudo mosquitto_passwd /etc/mosquitto/passwd otro_usuario
```

2. Configurar Mosquitto para usar el archivo de contraseñas

- Abre el archivo de configuración de Mosquitto en un editor de texto:

```
sudo nano /etc/mosquitto/mosquitto.conf
```

- Agrega las siguientes líneas al archivo de configuración:

```
allow_anonymous false  
password_file /etc/mosquitto/passwd
```

3. Reiniciar el servicio Mosquitto

Para que los cambios surtan efecto, debes reiniciar el servicio de Mosquitto.

- Ejecuta el siguiente comando:

```
sudo systemctl restart mosquitto
```

4. Si mosquito se cae luego de crear usuario y clave, verificar permisos del archivo passwd:

Solución: Cambiar los permisos del archivo **passwd**

1. Cambia el propietario del archivo:

- El archivo debe ser propiedad del usuario mosquito y del grupo mosquito.
- Ejecuta el siguiente comando:

```
sudo chown mosquito:mosquito /etc/mosquito/passwd
```

2. Cambia los permisos del archivo:

- El archivo debe tener permisos de lectura para el propietario (mosquito).
- Ejecuta el siguiente comando:

```
sudo chmod 644 /etc/mosquito/passwd
```

- Esto establecerá los permisos como -rw-r--r--, lo que permitirá que el usuario mosquito lea el archivo.

3. Verifica los permisos:

- Ejecuta el siguiente comando para asegurarte de que los permisos se hayan aplicado correctamente:

```
ls -l /etc/mosquito/passwd
```

- Deberías ver algo como esto:

```
-rw-r--r-- 1 mosquito mosquito 20 May 1 12:34 /etc/mosquito/passwd
```

4. Reinicia Mosquitto:

- Reinicia el servicio de Mosquitto para aplicar los cambios:

```
sudo systemctl restart mosquitto
```

5. Verifica el estado del servicio:

- Comprueba que Mosquitto se esté ejecutando correctamente:

```
sudo systemctl status mosquitto
```

Publicar un mensaje con mosquitto_pub usando usuario y contraseña.

Suscribirse:

```
mosquitto_sub -h 192.168.18.148 -t /casa/foco -d -u myusuario -P mypasswd
```

Publicar:

```
mosquitto_pub -h 192.168.18.148 -t /casa/foco -m 'hola mundo' -u myusuario -P mypasswd -d
```

Prueba;

```
fredd@freddy-alcarazo: ~  
eddy-alcarazo:~$ mosquitto_pub -h 192.168.18.148 -t /casa/foco -m 'hola mundo'  
eddy-alcarazo:~$ mosquitto_pub -h 192.168.18.148 -t /casa/foco -m 'hola mundo'  
on error: Connection Refused: not authorised.  
he connection was refused.  
eddy-alcarazo:~$ mosquitto_pub -h 192.168.18.148 -t /casa/foco -m 'hola mundo' -u admin -P 123456  
eddy-alcarazo:~$ mosquitto_pub -h 192.168.18.148 -t /casa/foco -m 'hola mundo' -u admin -P 123456  
eddy-alcarazo:~$ mosquitto_pub -h 192.168.18.148 -t /casa/foco -m 'hola mundo' -u admin -P 123456  
eddy-alcarazo:~$ mosquitto_pub -h 192.168.18.148 -t /casa/foco -m 'hola mundo' -u admin -P 123456  
eddy-alcarazo:~$ mosquitto_pub -h 192.168.18.148 -t /casa/foco -m 'hola mundo' -u admin -P 123456  
eddy-alcarazo:~$ mosquitto_pub -h 192.168.18.148 -t /casa/foco -m 'hola mundo' -u admin -P 123456  
eddy-alcarazo:~$  
  
fredd@freddy-alcarazo:~  
fredd@freddy-alcarazo:~$ mosquitto_sub -h 192.168.18.148 -t /casa/foco -d -u admin -P 123456  
Client null sending CONNECT  
Client null received CONNACK (0)  
Client null sending SUBSCRIBE (Mid: 1, Topic: /casa/foco, QoS: 0, Options: 0x00)  
Client null received SUBACK  
Subscribed (mid: 1): 0  
Client null received PUBLISH (d0, q0, r0, m0, '/casa/foco', ... (10 bytes))  
hola mundo  
Client null received PUBLISH (d0, q0, r0, m0, '/casa/foco', ... (10 bytes))  
hola mundo
```

Habilitar WebSockets

Pasos:

1. Abrir el archivo **mosquitto.conf** y agregar lo siguiente:

```
listener 9001
protocol websockets
allow_anonymous false
```

2. Reiniciar mosquitto.

3. Abrir el puerto 9001 en el firewall:

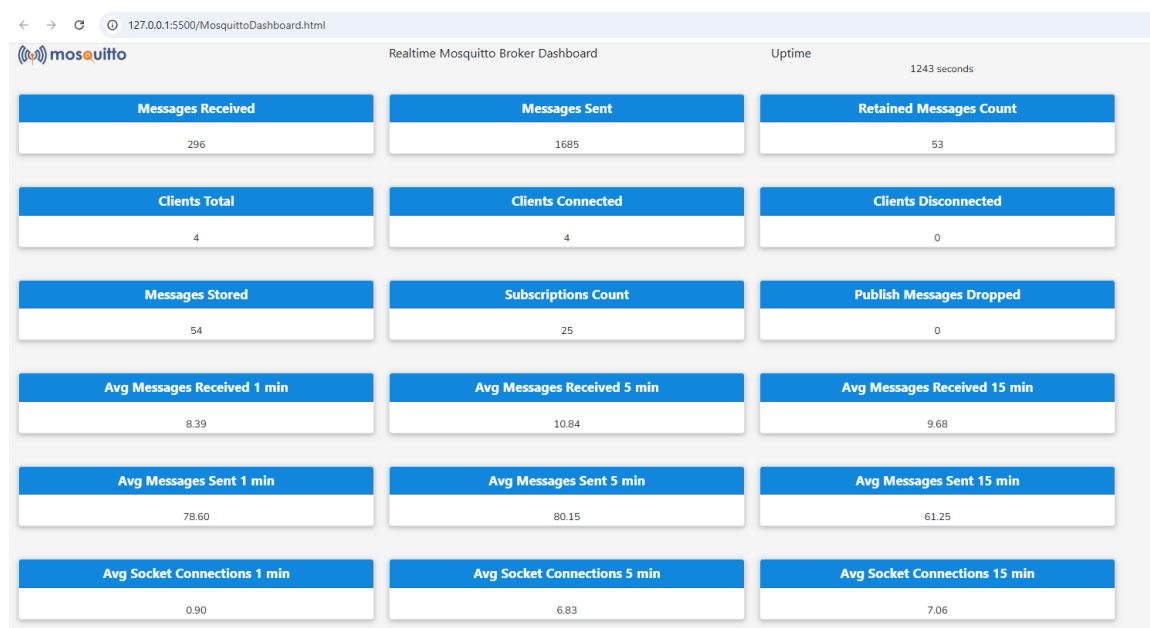
```
$ sudo ufw allow 9001
```

```
$ sudo ufw reload
```

Con esto será suficiente para que otras aplicaciones se puedan conectar mediante websockets. Ejemplo de esto es la aplicación "MosquittoDashboard":

<https://github.com/sanjeshpathak/Mosquitto-Dashboard>

En esta aplicacion modificar el archivo "dashboard.js" y agregar la ip o dominio del servidor y el puerto, usar SSL false. Y listo con esto abrimos el archivo "MosquittoDashboard.html" y ingresan las credenciales y luego se mostrará el dashboard.



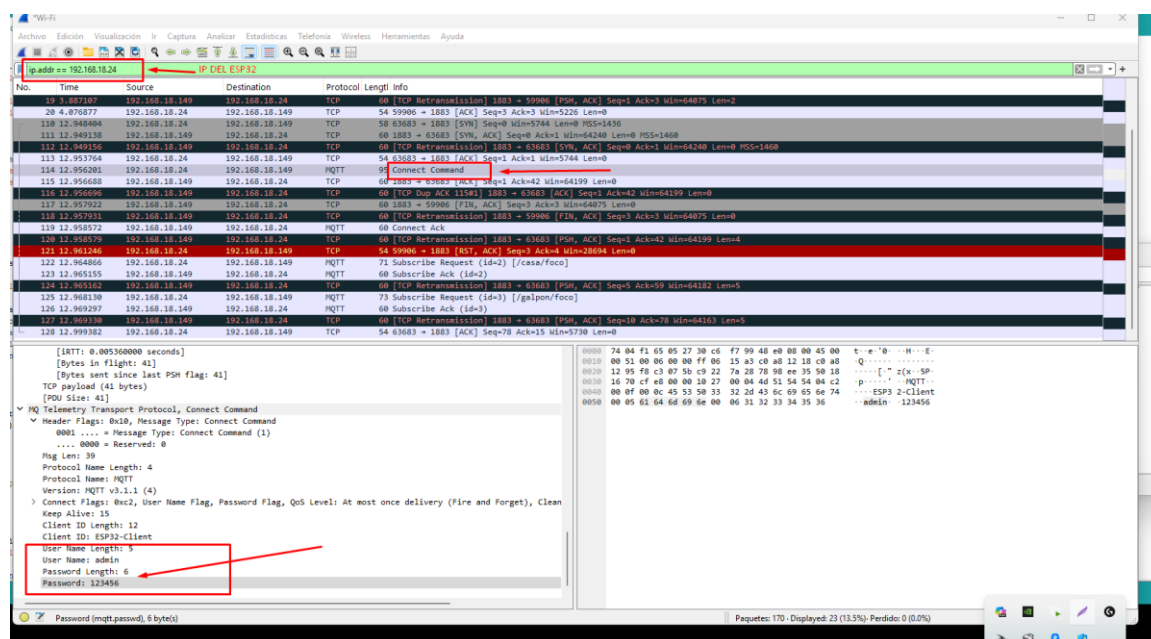
Fuentes:

http://www.steves-internet-guide.com/mosquitto_pub-sub-clients/

Generar certificados autofirmados (self-signed certificates) y configurar TLS en Mosquitto

Veamos como instalar los certificados a nuestro broker MQTT y firmar todos los certificados por nuestra cuenta, exploremos eso.

Actualmente solo tenemos un método de seguridad muy debil por no decir vulnerable, solo veamos la imagen de abajo, usando el sniffer de red wireshark podemos ver que cuando el ESP32 inicia la conexión con el broker las credenciales de usuario y contraseña son visible en la trama. Además los topicos o comandos tambien son visibles, esto no va a pasar si habilitamos TLS.



He encontrado un script bash para simplificar la generación de claves y moverlas al directorio correcto de certs de mosquitto. A continuación debemos de crear un script en lenguaje bash para poder generar los certificados, solo tenemos que cambiar la IP por la de nuestro servidor o si estamos en producción agregar el dominio ejemplo "mqtt.miempresa.com"

Este script tambien lo pueden encontrar en mi repositorio:

<https://github.com/alcarazolabs/mosquitto-esp32-tls-security/blob/main/tls-certs-script.bash>

O bien pueden copiar el contenido de este script que se muestra a continuación y crearlo por su propia cuenta:


```
#!/bin/bash
```

```
IP="192.168.1.22"
SUBJECT_CA="/C=SE/ST=Stockholm/L=Stockholm/O=himinds/OU=CA/CN=$IP"
SUBJECT_SERVER="/C=SE/ST=Stockholm/L=Stockholm/O=himinds/OU=Server/CN=$IP"
SUBJECT_CLIENT="/C=SE/ST=Stockholm/L=Stockholm/O=himinds/OU=Client/CN=$IP"

function generate_CA () {
    echo "$SUBJECT_CA"
    openssl req -x509 -nodes -sha256 -newkey rsa:2048 -subj "$SUBJECT_CA" -days 365 -keyout
ca.key -out ca.crt
}

function generate_server () {
    echo "$SUBJECT_SERVER"
    openssl req -nodes -sha256 -new -subj "$SUBJECT_SERVER" -keyout server.key -out server.csr
    openssl x509 -req -sha256 -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out
server.crt -days 365
}

function generate_client () {
    echo "$SUBJECT_CLIENT"
    openssl req -new -nodes -sha256 -subj "$SUBJECT_CLIENT" -out client.csr -keyout client.key
    openssl x509 -req -sha256 -in client.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out
client.crt -days 365
}

function copy_keys_to_broker () {
    sudo cp ca.crt /etc/mosquitto/certs/
    sudo cp server.crt /etc/mosquitto/certs/
    sudo cp server.key /etc/mosquitto/certs/
}

generate_CA
generate_server
generate_client
copy_keys_to_broker
```

Ejecución:

1. Cambiar la IP por su IP o agregar dominio.

2. Agregar el numero de dias de vencimiento ver -days por ejemplo 365.

3. Dar permisos de ejecucion al script:

sudo chmod +x tls-certs-script.bash

4. Ejecutar:

sudo ./ls-certs-script.bash

Para ejecutar este script, les recomiendo que lo ejecuten dentro de una carpeta creada en su directorio `/home/myuser/Documents/mosquitto-certs` porque se van a generar varios archivos, pero al final solo se van a copiar 3 archivos a la carpeta "certs" de mosquitto los cuales son:

```
ca.crt  
server.crt  
server.key
```

Observar el código del script bash, al final estos certificados se copian mediante el comando "**cp**" al directorio "**certs**" de mosquitto como se ve abajo:

```
sudo cp ca.crt /etc/mosquitto/certs/  
sudo cp server.crt /etc/mosquitto/certs/  
sudo cp server.key /etc/mosquitto/certs/
```

Ahora es momento de indicarle al archivo "mosquitto.conf" que use dichos certificados para lograr una conexión segura. Editamos el archivo ubicado en `/etc/mosquitto/mosquitto.conf` y agregamos lo siguiente:

```
listener 8883  
cafile /etc/mosquitto/certs/ca.crt  
certfile /etc/mosquitto/certs/server.crt  
keyfile /etc/mosquitto/certs/server.key  
require_certificate true  
use_identity_as_username true
```

El comando "`use_identity_as_username true`" con valor "true" se le indica a mosquitto a que no use usuario y contraseña en la conexión en su lugar que use los certificados, si lo ponemos en "false" tenemos que pasarle en el código del "Esp32" o en la consola si estamos usando mosquitto-clients el usuario y contraseña creados anteriormente.

A continuación les dejo una captura de pantalla del archivo mosquitto.conf y vean como ha quedado:

```
freddy-alc@freddy-pc: ~  
persistence true  
persistence_location /var/lib/mosquitto/  
  
log_dest file /var/log/mosquitto/mosquitto.log  
  
include_dir /etc/mosquitto/conf.d  
  
allow_anonymous false  
listener 1883 0.0.0.0  
password_file /etc/mosquitto/passwd  
cafile /etc/mosquitto/certs/ca.crt  
certfile /etc/mosquitto/certs/server.crt  
keyfile /etc/mosquitto/certs/server.key  
require_certificate true  
use_identity_as_username true  
  
listener 9001  
protocol websockets  
allow_anonymous false  
freddy-alc@freddy-pc:~$
```

Una vez actualizado el archivo "mosquitto.conf" y reiniciamos el servicio mosquitto, sin embargo verán que no puede iniciar esto es debido a que los certificados copiados a la carpeta "certs" de mosquitto no tienen los permisos adecuados, para solucionar esto ejecutemos el siguiente script bash para poder cambiarles los permisos:

```
#!/bin/bash  
echo "changing to mosquitto:mosquitto group.."  
chown mosquitto:mosquitto /etc/mosquitto/certs/ca.crt  
chown mosquitto:mosquitto /etc/mosquitto/certs/server.crt  
chown mosquitto:mosquitto /etc/mosquitto/certs/server.key  
echo "1. Group changed (ok).."  
echo "adding lecture permissions"  
chmod 664 /etc/mosquitto/certs/ca.crt  
chmod 664 /etc/mosquitto/certs/server.crt  
chmod 664 /etc/mosquitto/certs/server.key  
echo "2. lecture permissions done (ok)."  
echo "run ls -l /etc/mosquitto/certs/ to check!.."  
echo "end!"
```

Ejecución:

1. Dar permisos de ejecución a este script:

```
# sudo chmod +x change-permissions-certs.bash
```

2. Ejecutar con sudo:

```
# sudo ./change-permissions-certs.bash
```

Este script igualmente que el anterior se encuentra en mi repositorio:

<https://github.com/alcarazolabs/mosquitto-esp32-tls-security/blob/main/change-permissions-certs.bash>

Luego de la ejecución verán que los permisos han cambiado:

```
freddy-alc@freddy-pc:~$ ls -l /etc/mosquitto/certs/
total 16
-rw-rw-r-- 1 mosquitto mosquitto 1354 feb 18 18:23 ca.crt
-rw-r--r-- 1 root      root      130  abr 15  2024 README
-rw-rw-r-- 1 mosquitto mosquitto 1237 feb 18 18:23 server.crt
-rw-rw-r-- 1 mosquitto mosquitto 1704 feb 18 18:23 server.key
freddy-alc@freddy-pc:~$
```

Ahora si reinicia el servicio y verán que se ejecuta con normalidad:

```
# sudo systemctl restart mosquitto
```

Resumen

Para el MOSQUITTO broker usamos:

ca.crt

server.key

server.crt

Y para el el Publisher or subscriber "ESP32" usaremos:

ca.crt

client.key

client.crt

Ahora les comparto el código del ESP32 para conectarnos al broker mediante TLS:

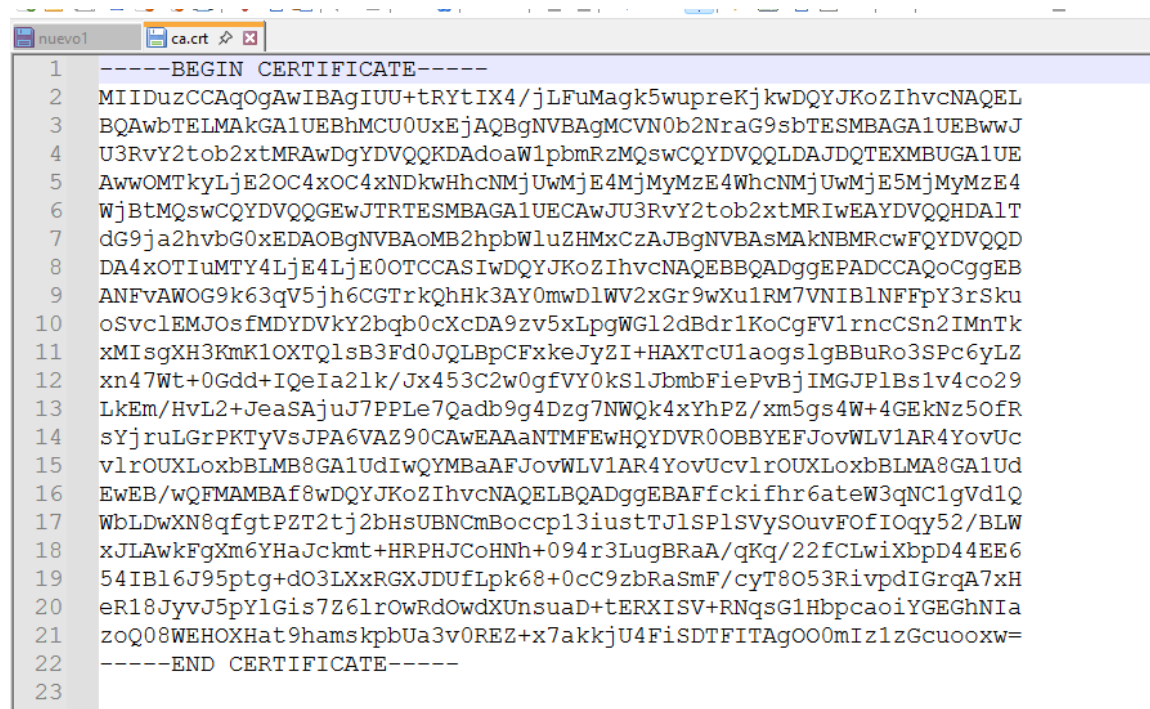
<https://github.com/alcazolzabs/mosquitto-esp32-tls-security/blob/main/mqtt-esp32-autenticado-tls-encryption/mqtt-esp32-autenticado-tls-encryption.ino>

Reemplazar los valores de las tres variables:

1. ca_cert[]
2. client_cert[]
3. client_key[]

para plasmar los datos de esos archivos mencionados arriba, abrir con un editor de código los archivos, ca.crt, client.key y client.crt

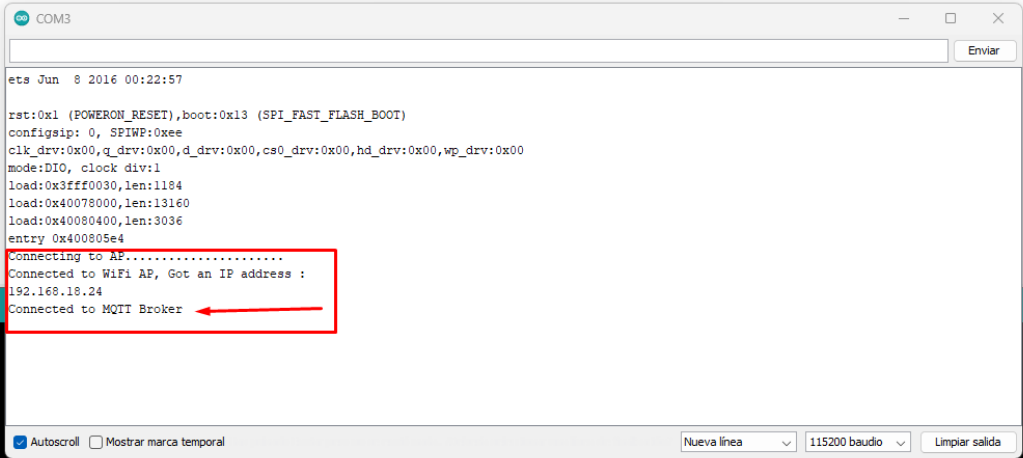
Ejemplo de ca.crt.



Nota: Estos tres archivos los copie a windows solo por cuestiones de comodidad. El broker lo tengo en una maquina virtual y mi arduino ide esta en windows.. etc etc. Copien los valores de los certificados y peguenlos en las variables respectiva mencionadas anteriormente.

Compilen y suban su código y verán que podrán conectarse:

```
MzE4
DA1T
VQ0D
ggEB
rSkU
MnTr
6yLZ
co29
50fR
ovUc
AlUd
Vd1Q
/BLW
4EE6
A7xH
hN1a
oxW=
```



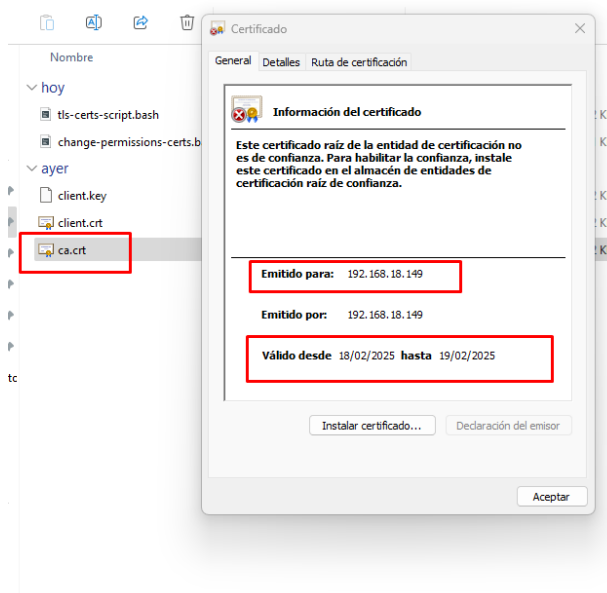
```
ets Jun 8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1184
load:0x40078000,len:13160
load:0x40080400,len:3036
entry 0x400805e4
Connecting to AP.....
Connected to WiFi AP, Got an IP address :
192.168.18.24
Connected to MQTT Broker
```

Autoscroll ☐ Mostrar marca temporal Nueva línea 115200 baudio Limpiar salida

seconds (effective 845.8 kbit/s)...

Otras cosas:

Pueden abrir con doble click en windows el archivo ca.cert y verán detalles como la fecha de emisión del certificado y fecha de vencimiento:



*Notas

Actualización y Duración de los Certificados

Es una excelente idea aumentar la duración de los certificados para evitar problemas de renovación frecuentes. Sin embargo, debes equilibrar la duración con la seguridad. Un certificado con una duración demasiado larga (por ejemplo, 10 años) puede ser un riesgo si se ve comprometido.

Recomendación:

- Para certificados de servidor (server.crt), una duración de **2 a 5 años** es razonable.
- Para certificados de clientes (client.crt), una duración de **1 a 2 años** es suficiente.
- Para el certificado de la CA (ca.crt), puedes usar una duración más larga, como **10 años**, ya que es menos probable que cambie.

Cómo cambiar la duración:

En el script bash, modifica el parámetro `-days` en los comandos `openssl` para ajustar la duración. Por ejemplo:

```
openssl req -x509 -nodes -sha256 -newkey rsa:2048 -subj "$SUBJECT_CA" -days 3650 -keyout ca.key -out ca.crt  
  
openssl x509 -req -sha256 -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days 1825  
  
openssl x509 -req -sha256 -in client.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out client.crt -days 365
```

- `-days 3650`: 10 años para la CA.
- `-days 1825`: 5 años para el servidor.
- `-days 365`: 1 año para los clientes.

Reutilización de Certificados

Si ya has generado los certificados y no quieres volver a generarlos para no afectar a los clientes, puedes **reutilizarlos** en el nuevo servidor. Simplemente copia los archivos **ca.crt**, **server.crt** y **server.key** al nuevo servidor y configúralos en Mosquitto.

Ventajas de reutilizar certificados:

- No necesitas redistribuir nuevos certificados a los clientes.
- Evitas interrupciones en la conectividad.

Consideraciones:

- Asegúrate de que los certificados no estén comprometidos.
- Verifica que la fecha de expiración de los certificados sea suficiente para tu uso.

Opinión sobre `ca.crt`, `server.crt` y `server.key`

- **ca.crt:** Este es el certificado de la Autoridad Certificadora (CA). Es el más importante, ya que se utiliza para verificar la autenticidad de los certificados del servidor y los clientes. Debes protegerlo y asegurarte de que tenga una duración larga (por ejemplo, 10 años).
- **server.crt:** Este es el certificado del servidor Mosquitto. Debe ser firmado por la CA y tener una duración moderada (por ejemplo, 2-5 años).
- **server.key:** Esta es la clave privada del servidor. Es el archivo más sensible, ya que cualquiera que tenga acceso a él puede suplantar al servidor. Debe estar protegido con permisos restrictivos (600) y almacenado en un lugar seguro.

Recomendaciones Finales

- **Copia de seguridad:** Guarda una copia de seguridad de `ca.crt`, `server.crt` y `server.key` en un lugar seguro.
- **Renovación planificada:** Aunque aumentes la duración de los certificados, planifica su renovación con anticipación para evitar problemas.
- **Documentación:** Documenta el proceso de generación y copia de los certificados para facilitar su gestión en el futuro.

Con estas prácticas, podrás mover y reutilizar tus certificados sin problemas, y evitarás afectar a los clientes existentes. ¡Buena suerte!

¿Qué sucede cuando actualizas los certificados?

Cuando ejecutas los comandos que mencione, estás generando nuevos certificados con una duración extendida:

- **ca.crt:** El certificado de la Autoridad Certificadora (CA) tiene una duración de 10 años (-days 3650).
- **server.crt:** El certificado del servidor tiene una duración de 5 años (-days 1825).
- **client.crt:** El certificado del cliente tiene una duración de 1 año (-days 365).

¿Debes actualizar el `ca.crt` en el ESP32?

Depende de si el `ca.crt` ha cambiado o no.

Caso 1: El `ca.crt` no ha cambiado

- Si solo has actualizado los certificados del servidor (`server.crt`) y del cliente (`client.crt`), pero **no has regenerado el `ca.crt`**, entonces **no necesitas actualizar el `ca.crt` en el ESP32.**

- El `ca.crt` es el certificado raíz que se utiliza para verificar la autenticidad de los certificados del servidor y del cliente. Si no ha cambiado, el ESP32 puede seguir usando el mismo `ca.crt` que ya tiene.

Caso 2: El `ca.crt` ha cambiado

- Si has regenerado el `ca.crt` (por ejemplo, porque has cambiado la clave privada de la CA o porque has decidido crear una nueva CA), entonces **sí debes actualizar el `ca.crt` en el ESP32**.
- Si no actualizas el `ca.crt` en el ESP32, este no podrá verificar la autenticidad de los nuevos certificados del servidor (`server.crt`) y del cliente (`client.crt`), lo que resultará en un fallo de conexión TLS.

¿Qué pasa con el servidor?

En el servidor (Mosquitto), debes asegurarte de que:

1. **ca.crt:** Si has regenerado el `ca.crt`, debes copiar el nuevo archivo a la carpeta de certificados del servidor (por ejemplo, `/etc/mosquitto/certs/`).
2. **server.crt y server.key:** Debes copiar los nuevos archivos generados (`server.crt` y `server.key`) a la carpeta de certificados del servidor.
3. **Reiniciar Mosquitto:** Después de actualizar los certificados, reinicia el servicio Mosquitto para que cargue los nuevos archivos.

¿Qué pasa con el ESP32?

En el ESP32, debes asegurarte de que:

1. **ca.crt:** Si has regenerado el `ca.crt`, debes actualizar el contenido de la variable `ca_cert` en el código del ESP32 con el nuevo certificado.
2. **client.crt y client.key:** Si has regenerado el `client.crt` y el `client.key`, debes actualizar las variables `client_cert` y `client_key` en el código del ESP32 con los nuevos valores.

3. **Recompilar y subir el código:** Después de actualizar los certificados en el código, recompila y sube el nuevo firmware al ESP32.

Resumen

- **Si solo has actualizado** `server.crt` **y** `client.crt`, no necesitas actualizar el `ca.crt` en el ESP32.
- **Si has regenerado el** `ca.crt`, debes actualizarlo tanto en el servidor como en el ESP32.
- **Siempre verifica** que los certificados en el servidor y en el ESP32 estén sincronizados para evitar errores de conexión TLS.

Recomendación adicional

Si decides regenerar el `ca.crt`, asegúrate de que todos los dispositivos (tanto servidores como clientes) tengan el nuevo `ca.crt`. Esto incluye:

- Servidores Mosquitto.
- Dispositivos IoT (como el ESP32).
- Cualquier otro cliente que se conecte al servidor.

Si no regeneras el `ca.crt`, pero actualizas `server.crt` y `client.crt`, solo necesitas actualizar esos archivos en el servidor y en los clientes afectados.

Fuentes:

- <https://suru-dissanaike.medium.com/mqtt-broker-with-secure-tls-communication-on-ubuntu-18-04-lts-and-an-esp32-mqtt-client-5c25fd7afe67>
- http://www.steves-internet-guide.com/mosquitto_pub-sub-clients/
- <https://medium.com/@alwaysHopeGood/iot-with-secure-mqtt-9239834107b1>
- https://www.reddit.com/r/esp32/comments/lia89w/how_do_you_secure_your_mqtt_based_esp32/