Certificate Management for MQTT with Eclipse Mosquitto

2021-10-04, Version 1.0

Jens Eliasson, jens.eliasson@thingwave.eu

1. Security in an Eclipse Arrowhead local cloud

An Arrowhead local cloud [1] can be deployed using three different security modes: NOT_SECURE, CERTIFICATE, and TOKEN. NOT_SECURE means that no encryption or security is used. CERTIFICATE means that TLS (or similar) is used with X.509 certificates. TOKEN means that in addition to X.509 certificates, tokens must also be used to further increase the security.

When using MQTT, a Broker must be configured with the proper settings for the corresponding mode. This document will outline hoe Eclipse Mosquitto, a very popular and widely used MQTT Broker, should be configured for use in an Arrowhead local cloud.

2. Certificate formats

The Core Java Spring reference implementation makes heavy use of PKCS #12 certificates. However, many programming languages and implementations doesn't support PKCS #12. Eclipse Mosquitto is one example where the certificate support is restricted to CRT/PEM formats. Also, many programming languages such as Python and Go primarily use CRT/PEM certificates. For this reason, it is important to be able to convert an Arrowhead certificate in PKCS #12 to PEM.

2.1 PKCS #12

PKCS #12 is an archive format that can contain certificates, key and certificate chains. PKCS #12 files use .p12 file ending. PKCS #12 files can also be password protected.

2.2 PEM / CRT

PEM is a files format based on BASE64 that can contain private keys, certificates and certificate chains. The file ending is .pem, but .crt is also often used. PEM is widely used in many applications and programming languages. Eclipse Mosquitto use PEM/CRT for supporting TLS in MQTTS.

3. Create PKCS #12 certificate for MQTT broker

This section will present how a standard Arrowhead PKCS #12 certificate should be created. The next Section will present how the broker PKCS #12 certificate can be converted to PEM format for use in Mosquitto, Python, Go, etc.

In [2], it is explained how a client certificate should be created using KeyStore Explorer. The rest of this section will use that document as a reference but provide the real names and settings for creating a MQTT broker certificate (for Eclipse Mosquitto [4]). This document is based on KeyStore Explorer 5.4.4 [3].

3.1 Load a local cloud truststore

In this example, we will use the *testcloud2* cloud certificate from GitHub [5] as an example. In a real deployment, custom certificate for all systems and clouds should of course be used instead. Start KeyStore Explorer and open testcloud2.p12 as depicted in Figure 1. Enter the default password of "123456".

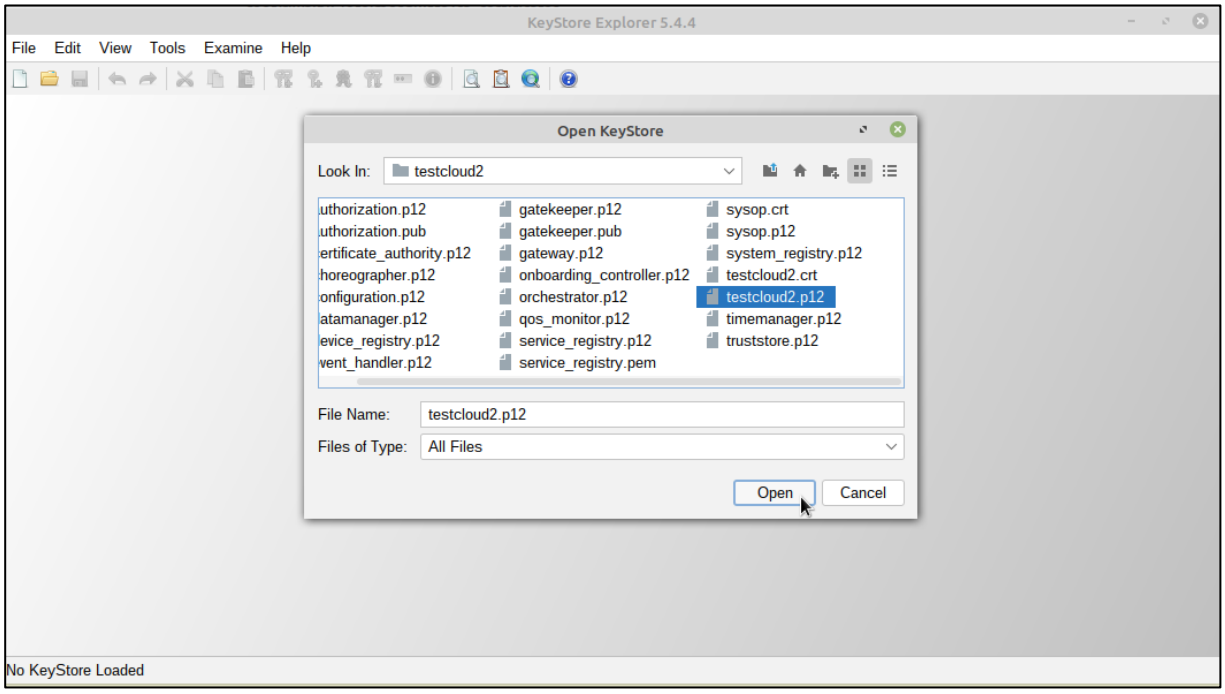

Figure 1 Load a local cloud truststore

3.2 Create new key pair

Right click on the testcloud2 entry and select "Sign New Key Pair" (enter the "123456" password again), as shown in Figure 2.

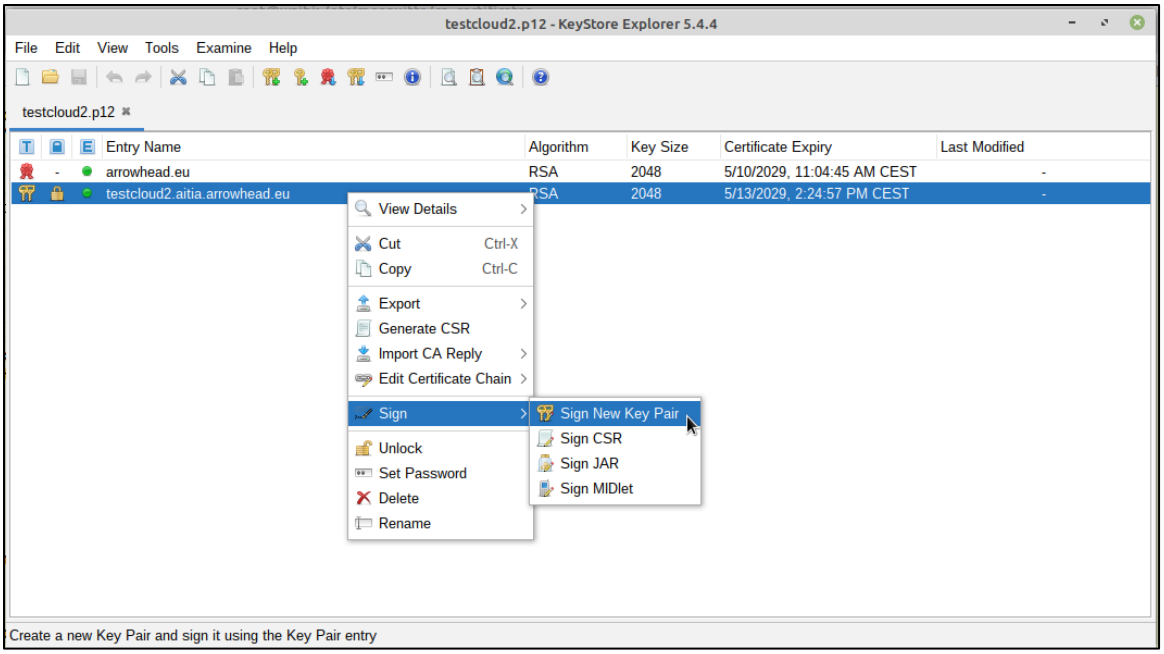

Figure 2 Sign New Key Pair

Select "RSA" with 2048 bits Key size and click "OK". Change the "Validity Period" to a suitable length, 1-10 years, klick "Apply". Then edit the "Name" and enter "**mqttbroker.testcloud2.aitia.arrowhead.eu**" as "Common Name", see Figure 3.

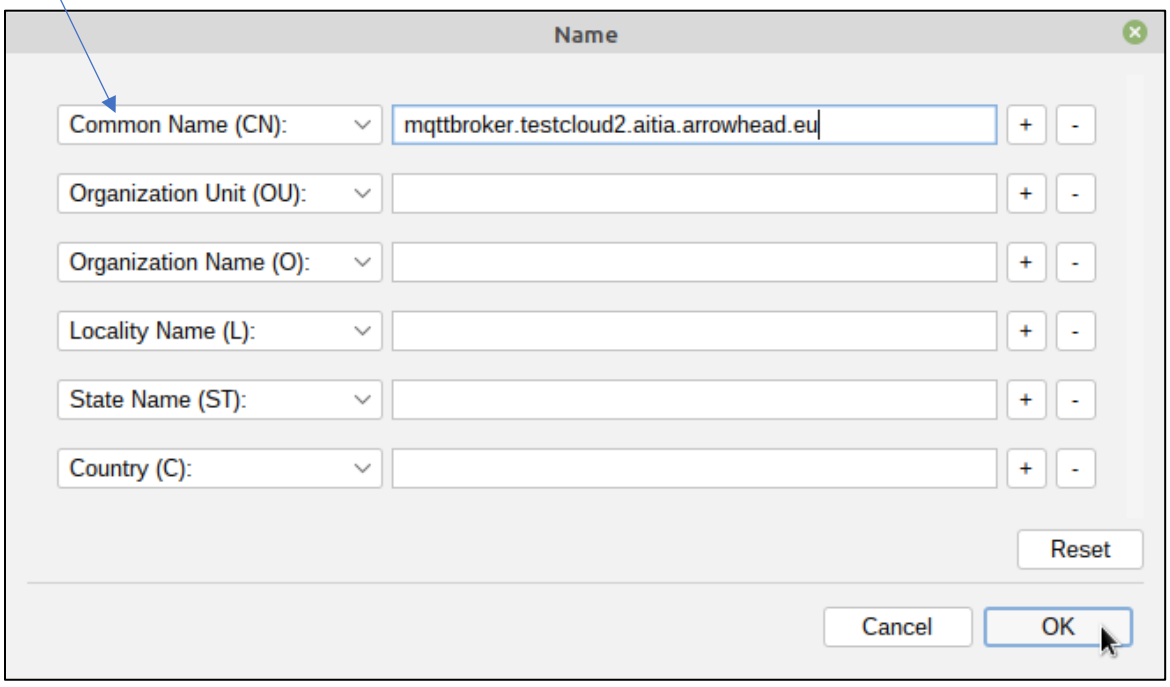

Figure 3 Certificate name details

The window should look like Figure 4 now.

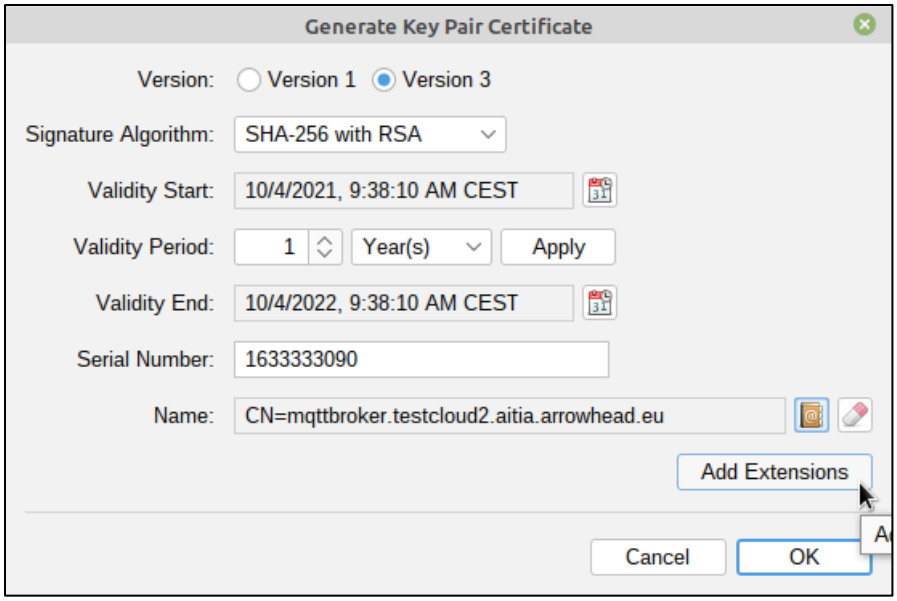

Figure 4 Certificate settings

Then click "Add Extensions" and click on the green plus sign. Then select to add "Authority Key Identifier" and click "OK". In the new window, then click on "Edit", as shown in Figure 5.

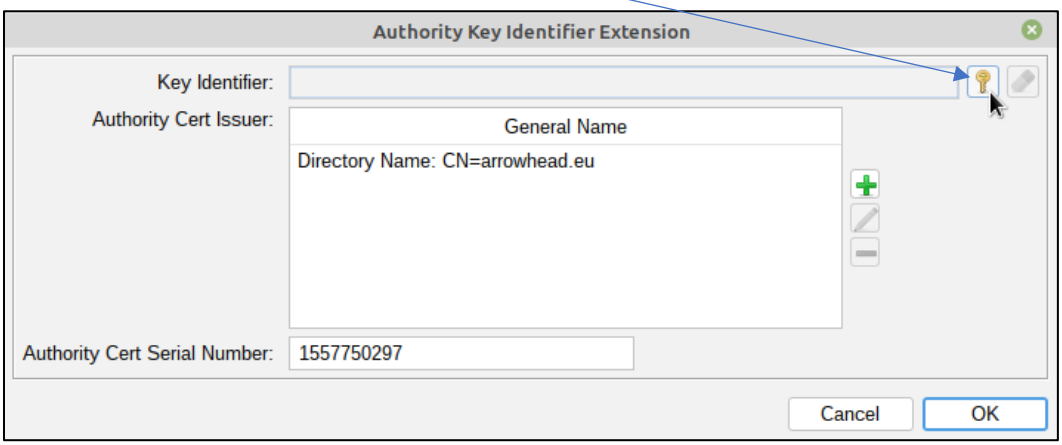

Figure 5 Edit Authority Key Identifier

Select 160-bit has (Figure 6) and click OK to close the window.

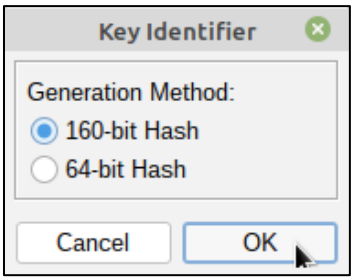

Figure 6 Choose a 160-bit hash

Now, click on the green plus sign again and select “Subject Alternative Name” and click OK. Click on the + button, select “DNS Name” and enter “localhost” as shown in. Repeat using the green plus sign and select IP “Address” and fill in “127.0.0.1” in the “General Name Value” field. If the local cloud will run in a distributed manner, repeat the process again but add the real IP address that Mosquitto will run on, for example 10.0.10.42. The “Subject Alternative Name Extension” window should look something like Figure 7. Then click OK to close it.

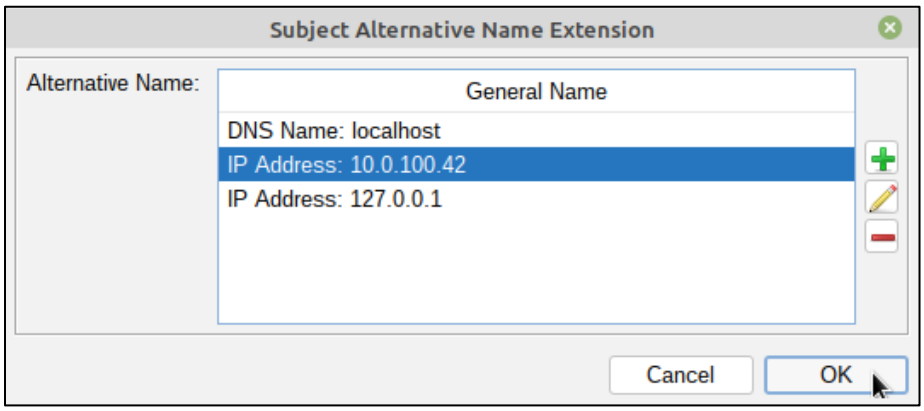

Figure 7 Subject Alternative Name Extension window

Next, click the + button to add another extension. Select “Subject Key Identifier” and choose 160-bit identifier. Click “OK”, and then “OK” again to save the settings.

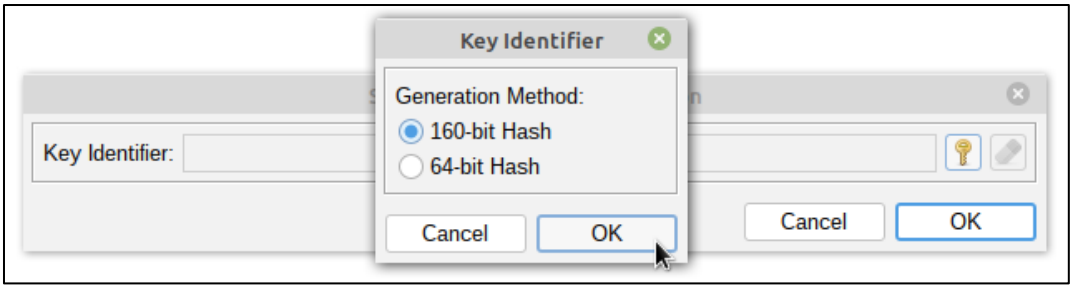

Figure 8 Key Identifier window

Click “OK” in the “Add Certificate Extensions” window, and “OK” in the “Generate Key Pair Certificate” window. Edit the alias and type “mqttbroker” as shown in Figure 9 and then click “OK”. Enter a password if your choice (use “123456” for the example certificates and something much stronger in a real deployment). Click “OK”. A pop-up window should now inform that the key pair generation was successful. Click OK in that window.

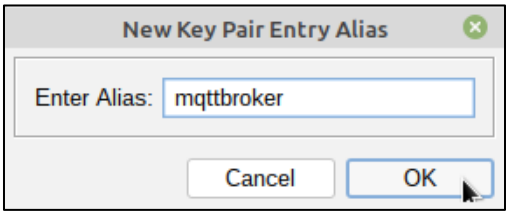

Figure 9 Key Pair Entry Alias window

The PKCS #12 certificate from the MQTT broker is now completed and should look like Figure 10.

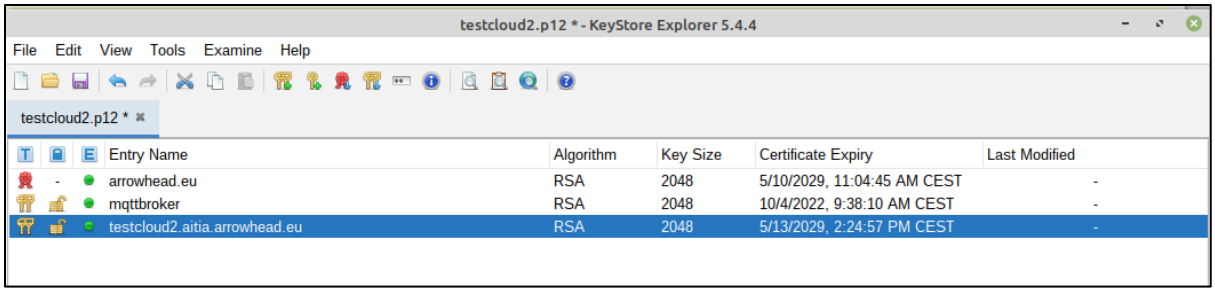

Figure 10 Complete certificate

Now, it is time to create the certificate chain in a standalone file. Drag and drop the *mqttbroker* certificate into a new tab, enter the password you selected earlier. In the new tab, right-click and choose "Import Trusted Certificate" as shown in Figure 11. Select the file "master.crt" (or your own root certificate).

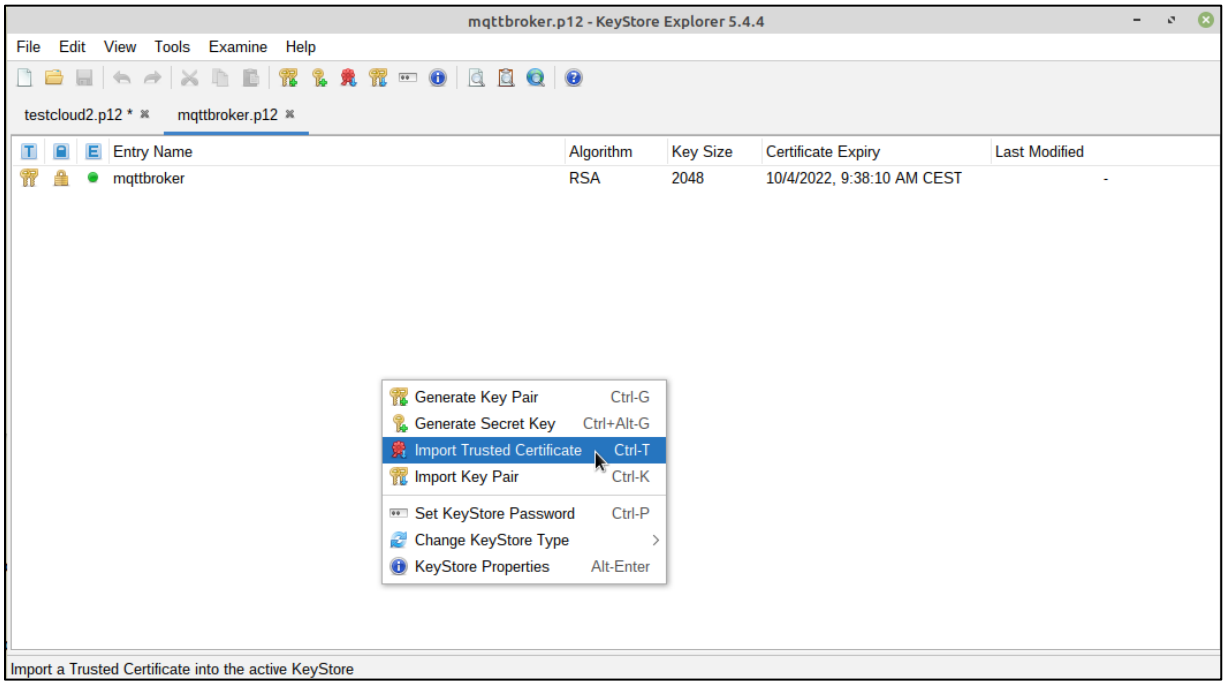

Figure 11 Import root certificate

Repeat and import the local cloud certificate (in this example testcloud2.crt or your own cloud certificate). After the root and intermediate certificates have been imported, click "Tools -> Set KeyStore Password". Select a password of your choice (e.g., "123456" when working with the test clouds provided in GitHub), as shown in Figure 12.

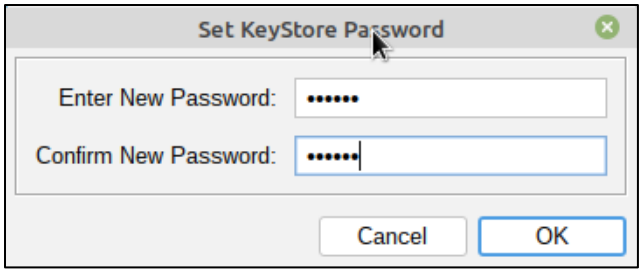

Figure 12 Set KeyStore Password window

Also click "Tools -> Change KeyStore Type" and make sure that the type is PKCS #12. The keystore should look like in Figure 13. Finally, to complete all steps, click "File -> Save As" and save the file as **mqttbroker.p12** in the proper folder (certificates/testcloud2 in the GitHub example code [5]).

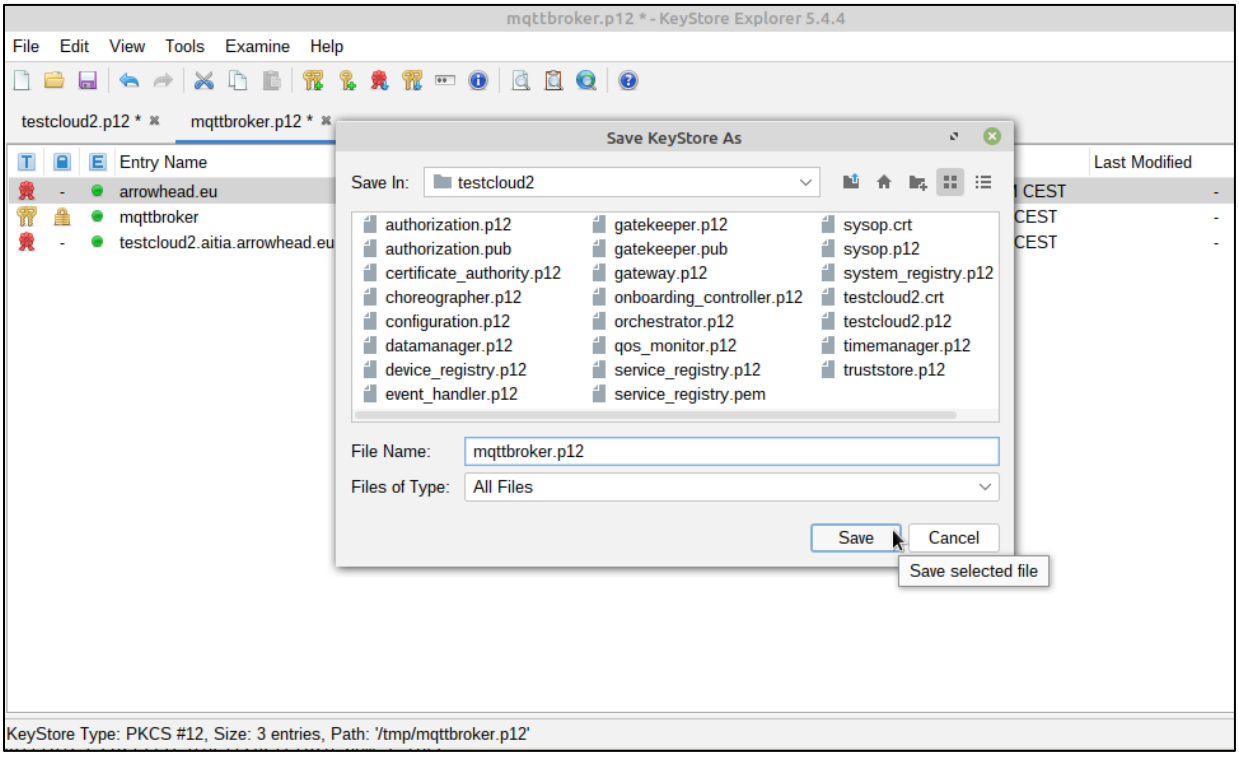

Figure 13 Finalized certificate

Also close the textcloud2.p12 file but DON'T save the changes! Now, the last step is to convert the PKCS #12 certificate we just created into a standard PEM encoded certificate chain that can be used by Eclipse Mosquitto.

4. Generate PEM from P12

4.1 Key file

To export the private key from the .p12 file, first right-click on the *mqttbroker* entry and then in the menu select “View Details -> Private Key Details”. Then click “PEM”.

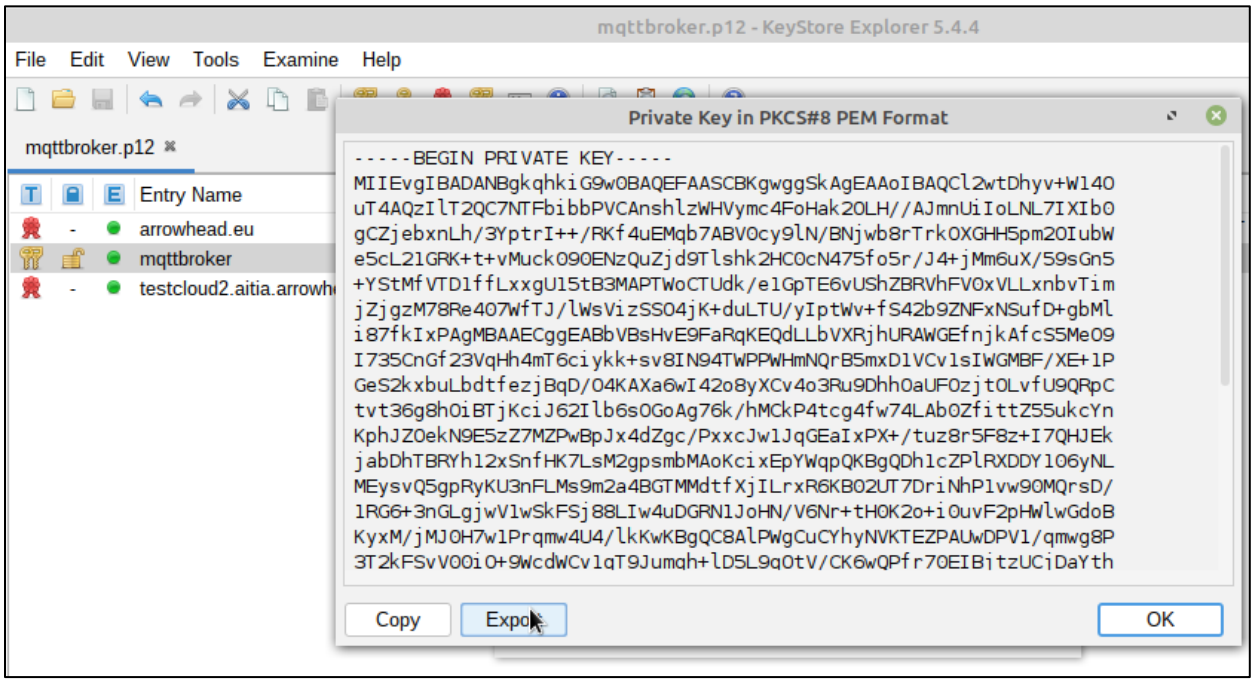

Figure 14 Export private key to PEM file

Click “Export”, as shown in Figure 14, and save the file as **mqttbroker.key** .

4.2 Certificate file

The certificate file **mqttbroker.pem** that will be created now will not only contain the *mqttbroker* certificate but also the root and intermediate certificates. The final PEM file will thus contain three certificates. The order of the certificates is important, going from the client certificate in the beginning of the file to intermediate certificate in the middle and the root certificate in the end of the file. See Appendix 3 for an example PEM certificate chain file.

Client: mqttbroker testcloud2.aita.arrowhead.eu
Intermediate: testcloud2.aita.arrowhead.eu
Root: arrowhead.eu

Schema 1 PEM file certificate chain

In order to export *mqttbroker’s* public key, double-click on the *mqttbroker* entry and in the “Certificate Hierarchy” select mqttbroker.testcloud2.aitia.arrowhead.eu (or what the cloud name is). Then click “PEM” to view the PEM encoded public key, shown in Figure 15. Click “Export” and save the public key as **mqttbroker.pem** or any other filename that is suitable.

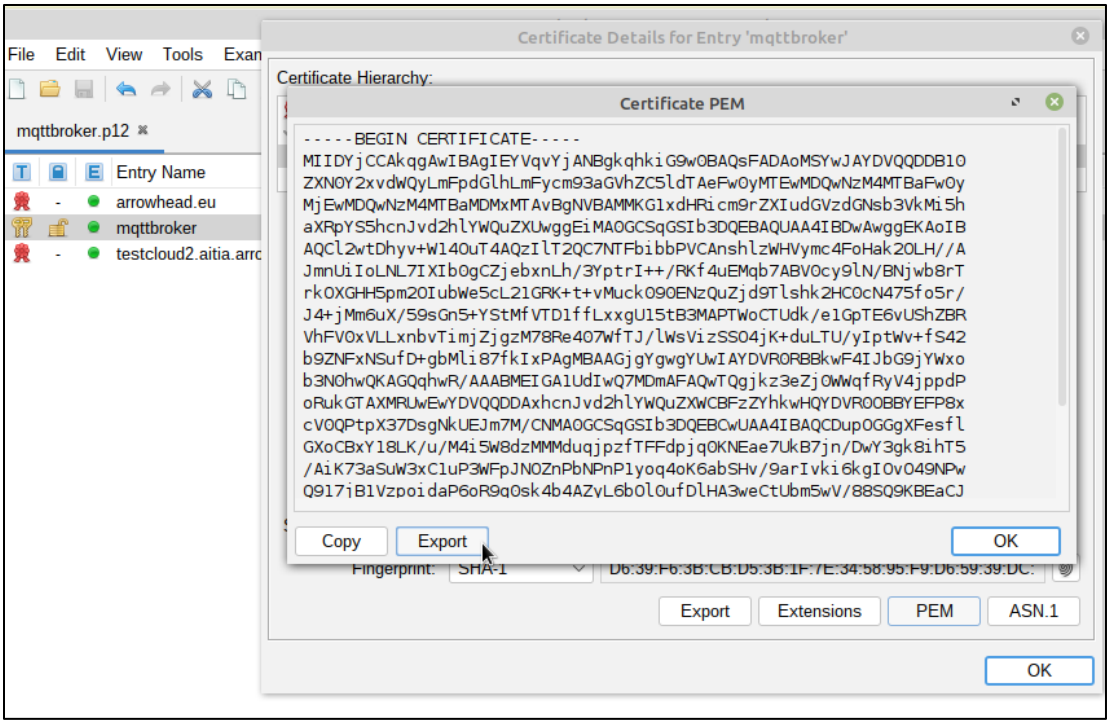

Figure 15 Export certificate to PEM file

Then *mqttbroker's* certificate has been exported, open the newly create file in a text editor. Now we must add the rest of the certificate chain, namely the intermediate (testcloud2.aitia.arrowhead.eu) and root (arrowhead.eu) certificates. Start by double-clicking on the entry testcloud2.aitia.arrowhead.eu and then click "PEM". Select all the text by typing Ctrl-A and then click "Copy". Go to the **mqttbroker.pem** file and paste the intermediate certificate we just copied at the end of the file. Repeat this process with the root certificate (arrowhead.eu) as well. Now double check that **mqttbroker.pem** contains three different certificates and save it.

4.3 Truststore file

The truststore file must contain two entries: the intermediate and the root certificates. The easiest way to create this is to simply copy the mqttbroker.pem file, and then delete the first certificate (*mqttbroker*). Then save the remaining two certificates (testcloud2.aitia.arrowhead.eu and arrowhead.eu) as **ca.pem** in a suitable folder.

Intermediate: testcloud2.aita.arrowhead.eu
Root: arrowhead.eu

Schema 2 CA certificate chain file

5. Mosquitto configuration files

Eclipse Mosquitto use sample text-based configuration files. On Ubuntu and other Debian based Linux flavours, the main configuration is in **/etc/mosquitto/mosquitto.conf** and administrators can add their own configuration files in **/etc/mosquitto/conf.d/** that will be read automatically when Mosquitto starts up.

5.1 Unsecure configuration

For testing etc, unsecure configuration can be used. Simply add these lines of text in the file **/etc/mosquitto/conf.d/arrowhead.conf**

```
port 1883
protocol mqtt

allow_anonymous true
```

5.2 Secure configuration

For real installations, secure configuration should be used. This includes adding certificates and username and passwords. Add these lines of text in **/etc/mosquitto/conf.d/arrowhead.conf**

```
port 8883
protocol mqtt

cafile /etc/mosquitto/ca_certs/ca.pem
certfile /etc/mosquitto/ca_certificates/mqttbroker.pem
keyfile /etc/mosquitto/ca_certificates/mqttbroker.key
require_certificate true
use_identity_as_username true

allow_anonymous true #set to false after testing
#password_file /etc/mosquitto/passwords
```

For managing passwords and other options, please refer to Mosquitto's official config file documentation.

5.3 Testing of TLS-enabled MQTT

To test the the Mosquitto broker with the Arrowhead-compliant certificates, we can use the **mosquitto_sub** and **mosquitto_pub** applications that comes with Mosquitto.

In one terminal:

```
ea@broker:/etc/mosquitto $ sudo mosquitto_pub -h 127.0.0.1 -p 8883 -t "test/topic" -m
"Hello, world!" --cafile ca_certificates/ca.pem --cert certs/mqttbroker.pem --key
certs/mqttbroker.key
```

In another terminal:

```
ea@broker:/etc/mosquitto $ sudo mosquitto_sub -h 127.0.0.1 -p 8883 -t "test/topic" --
cafile ca_certificates/ca.pem --cert certs/mqttbroker.pem --key certs/mqttbroker.key
Hello, world!
```

6. Conclusion

This document has provided documentation over how to generate an Arrowhead-compliant MQTT broker certificate in the PKCS #12 format, and how to convert it into the PEM format. Furthermore, configuration parameters for Eclipse Mosquitto have been presented. When combined, it is possible to setup a secure MQTT broker for use within an Eclipse Arrowhead local cloud.

References

[1] Eclipse Arrowhead Framework, www.arrowhead.eu
[2] Arrowhead certificate generation, https://github.com/arrowhead-f/client-skeleton-java-spring
[3] KeyStore Explorer, https://keystore-explorer.org/
[4] Eclipse Mosquitto MQTT broker, https://mosquitto.org/
[5] Eclipse Arrowhead GitHub repository, https://www.github.com/arrorhead-f/

Appendix 1: ca.pem

```
-----BEGIN CERTIFICATE-----
MIIDMzCCAhugAwIBAgIEXNliGTANBgkqhkiG9w0BAQsFADAXMRUwEwYDVQQDDAxh
cnJvd2hlYWQuZXUwHhcNMTkwNTEzMTIyNDU3WhcNMjkwNTEzMTIyNDU3WjAoMSYw
JAYDVQQDDB10ZXN0Y2xvdWQyLmFpdGlhLmFycm93aGVhZC5ldTCCASIwDQYJKoZI
hvcNAQEBBQADggEPADCCAQoCggEBAJV8P9a3Qjl3jcm0kLwKMqwKux6CHVyojfRO
Zu0A6NLp0BZjcyzsH66Xj22Fub6gcUnMx7sg3DsYANXyTYZTaRCn/058BaLpD8we
2Q0+XT2xEKsh8PtMkuj3Ebge93W6E5a/fvl+Xx3Ggi8SCC9OscG3hzN8cpXt7vPg
AU6k/TgsAAnAnCbVqlrevMSAy6Opq5g2RM795J/mvom9seGmL0xOilXTlGK9gZBB
HQfcY7sQc0/SJOUwNVQc9xNgRyf85EVgLFq2kkW4Tjyc8JNa6PgjUTF67NJQasXV
BY+EydvqwZsdeKnaHrjRdIQxQ8uZLZXmnVSPtQhX3cG/+untN88CAwEAAaN2MHQw
DwYDVR0TBAgwBgEB/wIBAjBCBgNVHSMEOzA5gBSaipkeMUwesuYttprtxx4xyt07
/qEbpBkwFzEVMBMGA1UEAwwMYXJyb3doZWFkLmV1ggRc1T6tMB0GA1UdDgQWBBQE
ME0II5M93mY9Flqn0cleI6aXTzANBgkqhkiG9w0BAQsFAAOCAQEAP4kKGckmZk/t
+OrZ8D1P0hhBbAyNWlDijVexoP8JGxo1blEbftfEJ3FdYL+CKFJ1wa7l+jhvAiW9
EiIk6Z/DvH5b9NMxyc72kTy4P0yYmXuJ2/x/rA2hdBwFuSx4KYjFOrMaSGwuCXae
ec61odXhtxC9Y7HGoIao+pNjD1qa82sgzRA6ZQSMyTGI1NaVfBco6VMb1Hu9U6i0
HdbUtu9O3VmkgAI/U7nF/qDcuBHaHXEco4ApTh+PWo6BLmqaBOdcpgRD0F7vc5y/
JviipqjU4oHiX7CQShdGUC4lkPZmXFlfsoXACIpcPDF/49BX1OAy4GNCl1PeyU7A
6STgROIXpw==
-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
MIIC3jCCAcagAwIBAgIEXNU+rTANBgkqhkiG9w0BAQsFADAXMRUwEwYDVQQDDAxh
cnJvd2hlYWQuZXUwHhcNMTkwNTEwMDkwNDQ1WhcNMjkwNTEwMDkwNDQ1WjAXMRUw
EwYDVQQDDAxhcnJvd2hlYWQuZXUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQCuB4z+wchXDKdfy9YFZha2U0khBAWuHYerBlBLM0Oqvr4c/YYcZpNwTWY2
tk6UXPUTQ8gI9V6Ob7DRXoAfLDhCRGKySN0BlGnjUJkItP25Sj0RfiTL3b8fFEIT
Z8pg+6pAfeFQgV0yz+ziyL+0uu69VZPv+RAEf1GKgshGGLJw3sOlcIdKuZaEAaA2
b0nDUn229VpKXb9cg47Ae1Yb0sJcTkyIBuhkQKln3uhG2xct9nDfVal05+229AJQ
Ly1f0UfEofvD/OLjFG3umF857T1Vr5azj8zFOvNi503gV458lH3wKC+9UHUf46sg
Hc8Tyrz1q8VAIamJe7BLUeRFwGvBAgMBAAGjMjAwMA8GA1UdEwQIMAYBAf8CAQMw
HQYDVR0OBBYEFJqKmR4xTB6y5i22mu3HHjHK3Tv+MA0GCSqGSIb3DQEBCwUAA4IB
AQCkFsqyeAjztDBkTQrPxAB0Vvx6KPINApHGIHkJj/9crKXZEQcNJcJr35hfLcgv
hSsmLMdeRFCeaG5QLmUKI6GFYIbX+6nawMLGzIPUTOGetNeuMauDXkq09Hu/UmjN
AOgoD5vWdtyTbItv21enJnUelClAJ7VXti2QpyRM2puPHpZMNi4FWgLGPo6hq5ka
d7KomzW8JLh2Vd67v/6mXGpST4EzyRe+Yb2FJZUmhxVWt68/MFaflPQ2toPIsIpW
5m4OS+rT7t+uxPKWU/ogCK7BOUfE/qf3Al/osWkNKnFQDtO/7x7InEmRoP9EUv07
JY4vK/+k5fJUZHpJzpuKxbBo
-----END CERTIFICATE-----
```

Appendix 2: mqttbroker.key


```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCl2wtDhyv+W14O
uT4AQzIlT2QC7NTFbibbPVCAnshlzWHVymc4FoHak2OLH//AJmnUiIoLNL7IXIb0
gCZjebxnLh/3YptrI++/RKf4uEMqb7ABV0cy9lN/BNjwb8rTrkOXGHH5pm2OIubW
e5cL21GRK+t+vMuck090ENzQuZjd9Tlshk2HC0cN475fo5r/J4+jMm6uX/59sGn5
+YStMfVTD1ffLxxgU15tB3MAPTWoCTUdk/e1GpTE6vUShZBRVhFV0xVLLxnbvTim
jZjgzM78Re407WfTJ/lWsVizSSO4jK+duLTU/yIptWv+fS42b9ZNFxNSufD+gbMl
i87fkIxPAgMBAAECggEABbVBsHvE9FaRqKEQdLLbVXRjhURAWGEfnjkAfcS5MeO9
I735CnGf23VqHh4mT6ciykk+sv8IN94TWPPWHmNQrB5mxD1VCv1sIWGMBF/XE+1P
GeS2kxbuLbdtfezjBqD/O4KAXa6wI42o8yXCv4o3Ru9DhhOaUFOzjtOLvfU9QRpC
tvt36g8hOiBTjKciJ62Ilb6sOGoAg76k/hMCkP4tcg4fw74LAb0ZfittZ55ukcYn
KphJZOekN9E5zZ7MZPwBpJx4dZgc/PxxcJw1JqGEaIxPX+/tuz8r5F8z+I7QHJEk
jabDhTBRYh12xSnfHK7LsM2gpsmbMAoKcixEpYWqpQKBgQDh1cZPlRXDDY106yNL
MEysvQ5gpRyKU3nFLMs9m2a4BGTMMdtfXjILrxR6KB02UT7DriNhP1vw90MQrsD/
1RG6+3nGLgjwV1wSkFSj88LIw4uDGRN1JoHN/V6Nr+tH0K2o+i0uvF2pHWlwGdoB
KyxM/jMJ0H7w1Prqmw4U4/lkKwKBgQC8AlPWgCuCYhyNVKTEZPAUwDPV1/qmwg8P
3T2kFSvV00iO+9WcdWCv1qT9Jumqh+lD5L9qOtV/CK6wQPfr70EIBjtzUCjDaYth
GdZ3L5aIgzdkGbeYTuiqLj0+7JPdrbrrTwIOMEHXNEQqW2ZAckadgAATrTfEyPfx
4zlc77GybQKBgQCNhhhG2xjOKXoSYSC3B8elIL0X16DxFqri49ViADb47tf0FvJw
62jVZZyhiO31kiiRHFnToY8OjQWcOMd+C5qW4O1JQqQbacOrZoOEU0gn5FNTjq1g
5wbFfhh5FluNn7qUgGM8PJM9Xc8rjyfNP679ePenGN/UMMQlrZG/ibZKDQKBgEL6
wUuF1fpZYbqoERs2qqF55/2jOFT4hc0tEKzILhm7b/oglfVri8qZOaZGmrB6QwxB
DkCWPUFIopOAUdLoJc7qCXwHbr5Pd6xcPB+yAHsZ5D3xO4szcYhPeNpWBkfPgOwb
9auAXbSBlldQwJjtHWM3Gs/nwhqFTnAKd4E4k6HxAoGBAJRykzyJkNSOW+Dfsd+r
Cx6SD9DsC3D6CqPSSMRqka06OzgYh3nnobABqg85w5OCBXgdCfFYW/2mt3WjBvng
+OOJ0JKiUIJpE5al4FW3eKLYRw8wT+d71JRTrYHESaWLSKaIJdxXgbuS0InMER0B
Dut5QkDEFJMp0kKIRKHP1MRm
-----END PRIVATE KEY-----
```

Appendix 3: mqttbroker.pem

```
-----BEGIN CERTIFICATE-----
MIIDYjCCAkqgAwIBAgIEYVqvYjANBgkqhkiG9w0BAQsFADAoMSYwJAYDVQQDDB10
ZXN0Y2xvdWQyLmFpdGlhLmFycm93aGVhZC5ldTAeFw0yMTEwMDQwNzM4MTBaFw0y
MjEwMDQwNzM4MTBaMDMxMTAvBgNVBAMMKG1xdHRicm9rZXIudGVzdGNsb3VkMi5h
aXRpYS5hcnJvd2hlYWQuZXUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQCl2wtDhyv+W14OuT4AQzIlT2QC7NTFbibbPVCAnshlzWHVymc4FoHak2OLH//A
JmnUiIoLNL7IXIb0gCZjebxnLh/3YptrI++/RKf4uEMqb7ABV0cy9lN/BNjwb8rT
rkOXGHH5pm2OIubWe5cL21GRK+t+vMuck090ENzQuZjd9Tlshk2HC0cN475fo5r/
J4+jMm6uX/59sGn5+YStMfVTD1ffLxxgU15tB3MAPTWoCTUdk/e1GpTE6vUShZBR
VhFV0xVLLxnbvTimjZjgzM78Re407WfTJ/lWsVizSSO4jK+duLTU/yIptWv+fS42
b9ZNFxNSufD+gbMli87fkIxPAgMBAAGjgYgwgYUwIAYDVR0RBBkwF4IJbG9jYWxo
b3N0hwQKAGQqhwR/AAABMEIGA1UdIwQ7MDmAFAQwTQgjkz3eZj0WWqfRyV4jppdP
oRukGTAXMRUwEwYDVQQDDAxhcnJvd2hlYWQuZXWCBFzZYhkwHQYDVR0OBBYEFP8x
cV0QPtpX37DsgNkUEJm7M/CNMA0GCSqGSIb3DQEBCwUAA4IBAQCDupOGGgXFesfl
GXoCBxY18LK/u/M4i5W8dzMMMduqjpzfTFFdpjq0KNEae7UkB7jn/DwY3gk8ihT5
/AiK73aSuW3xC1uP3WFpJNOZnPbNPnP1yoq4oK6abSHv/9arIvki6kgIOvO49NPw
Q917jB1VzpoidaP6oR9q0sk4b4AZyL6bOl0ufDlHA3weCtUbm5wV/88SQ9KBEaCJ
LLTYp447Tg0tAk/A9+dIH2jz3/eqXoL7WzxkUT0DWF3rKc5l/vxQvvsyXHYUZWHY
ZGgTKieSEOS/UElRjhky92NT4Rf+e0gFbAofGRx29Rd1efDc+QJFWYC2GzhDiplO
Js9zn2W6
-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
MIIDMzCCAhugAwIBAgIEXNliGTANBgkqhkiG9w0BAQsFADAXMRUwEwYDVQQDDAxh
cnJvd2hlYWQuZXUwHhcNMTkwNTEzMTIyNDU3WhcNMjkwNTEzMTIyNDU3WjAoMSYw
JAYDVQQDDB10ZXN0Y2xvdWQyLmFpdGlhLmFycm93aGVhZC5ldTCCASIwDQYJKoZI
hvcNAQEBBQADggEPADCCAQoCggEBAJV8P9a3Qjl3jcm0kLwKMqwKux6CHVyojfRO
Zu0A6NLp0BZjcyzsH66Xj22Fub6gcUnMx7sg3DsYANXyTYZTaRCn/058BaLpD8we
2Q0+XT2xEKsh8PtMkuj3Ebge93W6E5a/fvl+Xx3Ggi8SCC9OscG3hzN8cpXt7vPg
AU6k/TgsAAnAnCbVqlrevMSAy6Opq5g2RM795J/mvom9seGmL0xOilXTlGK9gZBB
HQfcY7sQc0/SJOUwNVQc9xNgRyf85EVgLFq2kkW4Tjyc8JNa6PgjUTF67NJQasXV
BY+EydvqwZsdeKnaHrjRdIQxQ8uZLZXmnVSPtQhX3cG/+untN88CAwEAAaN2MHQw
DwYDVR0TBAgwBgEB/wIBAjBCBgNVHSMEOzA5gBSaipkeMUwesuYttprtxx4xyt07
/qEbpBkwFzEVMBMGA1UEAwwMYXJyb3doZWFkLmV1ggRc1T6tMB0GA1UdDgQWBBQE
ME0II5M93mY9Flqn0cleI6aXTzANBgkqhkiG9w0BAQsFAAOCAQEAP4kKGckmZk/t
+OrZ8D1P0hhBbAyNWlDijVexoP8JGxo1blEbftfEJ3FdYL+CKFJ1wa7l+jhvAiW9
EiIk6Z/DvH5b9NMxyc72kTy4P0yYmXuJ2/x/rA2hdBwFuSx4KYjFOrMaSGwuCXae
ec61odXhtxC9Y7HGoIao+pNjD1qa82sgzRA6ZQSMyTGI1NaVfBco6VMb1Hu9U6i0
HdbUtu9O3VmkgAI/U7nF/qDcuBHaHXEco4ApTh+PWo6BLmqaBOdcpgRD0F7vc5y/
JviipqjU4oHiX7CQShdGUC4lkPZmXFlfsoXACIpcPDF/49BX1OAy4GNCl1PeyU7A
6STgROIXpw==
-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
MIIC3jCCAcagAwIBAgIEXNU+rTANBgkqhkiG9w0BAQsFADAXMRUwEwYDVQQDDAxh
cnJvd2hlYWQuZXUwHhcNMTkwNTEwMDkwNDQ1WhcNMjkwNTEwMDkwNDQ1WjAXMRUw
EwYDVQQDDAxhcnJvd2hlYWQuZXUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQCuB4z+wchXDKdfy9YFZha2U0khBAWuHYerBlBLM0Oqvr4c/YYcZpNwTWY2
tk6UXPUTQ8gI9V6Ob7DRXoAfLDhCRGKySN0BlGnjUJkItP25Sj0RfiTL3b8fFEIT
Z8pg+6pAfeFQgV0yz+ziyL+0uu69VZPv+RAEf1GKgshGGLJw3sOlcIdKuZaEAaA2
b0nDUn229VpKXb9cg47Ae1Yb0sJcTkyIBuhkQKln3uhG2xct9nDfVal05+229AJQ
Ly1f0UfEofvD/OLjFG3umF857T1Vr5azj8zFOvNi503gV458lH3wKC+9UHUf46sg
```

```
Hc8Tyrz1q8VAIamJe7BLUeRFwGvBAgMBAAGjMjAwMA8GA1UdEwQIMAYBAf8CAQMw
HQYDVR0OBBYEFJqKmR4xTB6y5i22mu3HHjHK3Tv+MA0GCSqGSIb3DQEBCwUAA4IB
AQCkFsqyeAjztDBkTQrPxAB0Vvx6KPINApHGIHkJj/9crKXZEQcNJcJr35hfLcgv
hSsmLMdeRFCeaG5QLmUKI6GFYIbX+6nawMLGzIPUTOGetNeuMauDXkq09Hu/UmjN
AOgoD5vWdtyTbItv21enJnUelClAJ7VXti2QpyRM2puPHpZMNi4FWgLGPo6hq5ka
d7KomzW8JLh2Vd67v/6mXGpST4EzyRe+Yb2FJZUmhxVWt68/MFaflPQ2toPIsIpW
5m4OS+rT7t+uxPKWU/ogCK7BOUfE/qf3Al/osWkNKnFQDtO/7x7InEmRoP9EUv07
JY4vK/+k5fJUZHpJzpuKxbBo
-----END CERTIFICATE-----
```