

EMQ X Cloud Webinar

Ensure MQTT Security with TLS/SSL in EMQ X

February 9th

9:00am EST / 3:00pm CET / 2:00pm UTC



Speaker:
Kary Ware, Sales Engineer @EMQ



Agenda

Introduction to EMQ X Cloud and on-prem products

Overview of SSL/TLS security

How to verify certificates and certificate chains using openssl

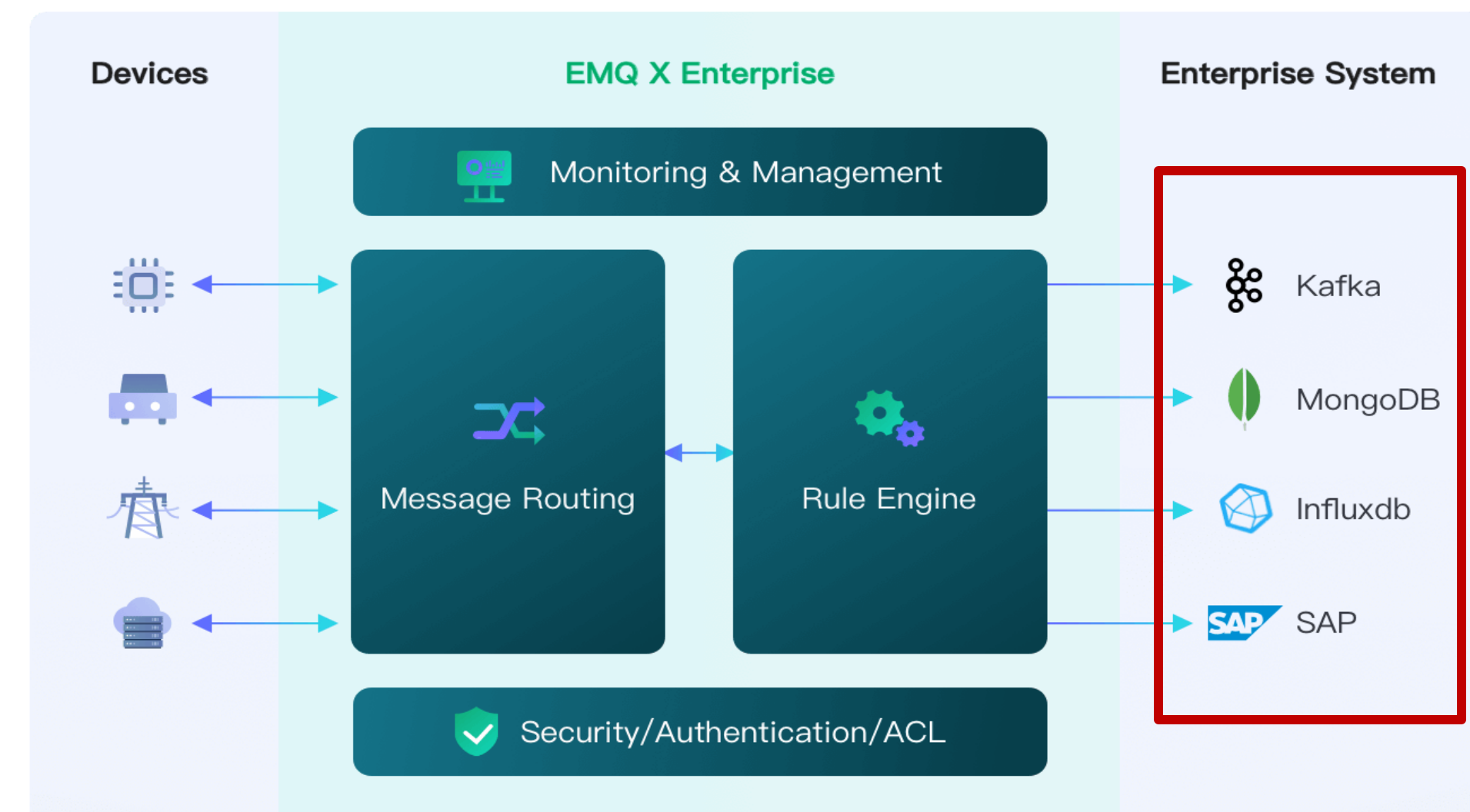
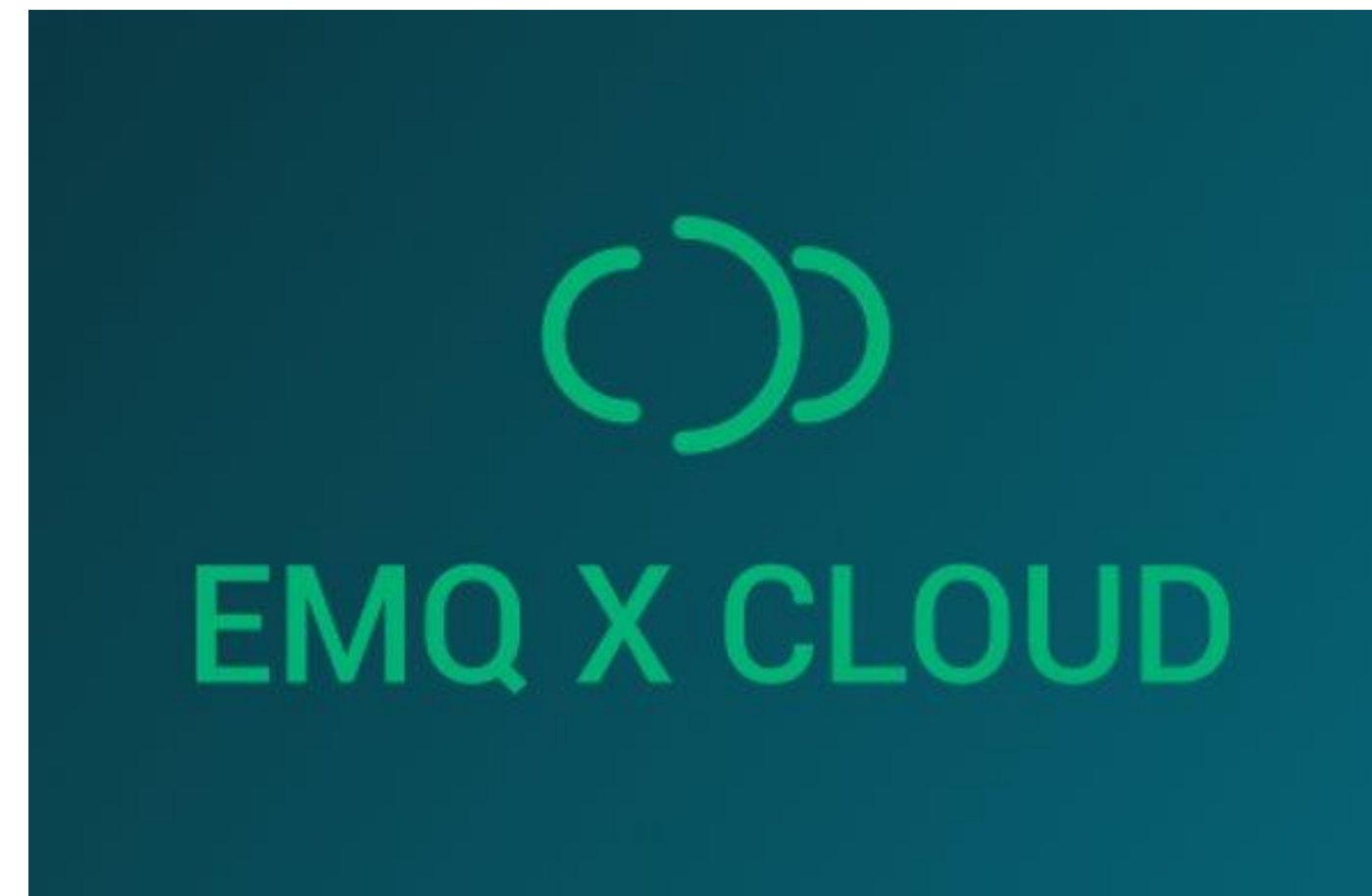
How to map certificate files to EMQ X listeners (cloud and on-prem)

Example: Mapping Let's Encrypt certificates to EMQ X

Demo: Generating self-signed certificates and mapping to EMQ X
Separate tutorial video so you can try it yourselves.

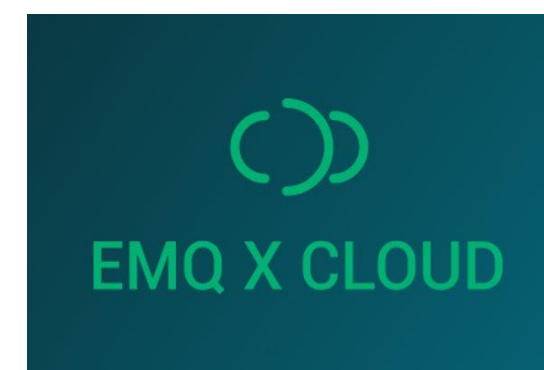
Summary and Q & A

Introduction to EMQ X Cloud and on-prem products



EMQ X Cloud

All plans offer high performance fully 100% MQTT compliant message brokers
Rule Engine: Process messages in real-time



Standard



Start at **\$ 0.18** per hour

Save 15% if you prepay annually

Create Now

- ✓ Single node
- ✓ Up to 10,000 connections
- ✓ Contains 100G traffic
- ✓ Support WebHook
- ✓ Support MQTT Bridge
- ✓ SLA 99%
- ✓ 8/5 technical support

Professional



Start at **\$ 0.99** per hour

Save 15% if you prepay annually

Create Now

- ✓ Multiple node cluster
- ✓ Exclusive resources
- ✓ Up to 100,000 connections
- ✓ Contains up to 1T traffic
- ✓ Support Rule Engine
- ✓ Support VPC Peering
- ✓ SLA 99.99%
- ✓ 24/7 technical support

TLS/SSL

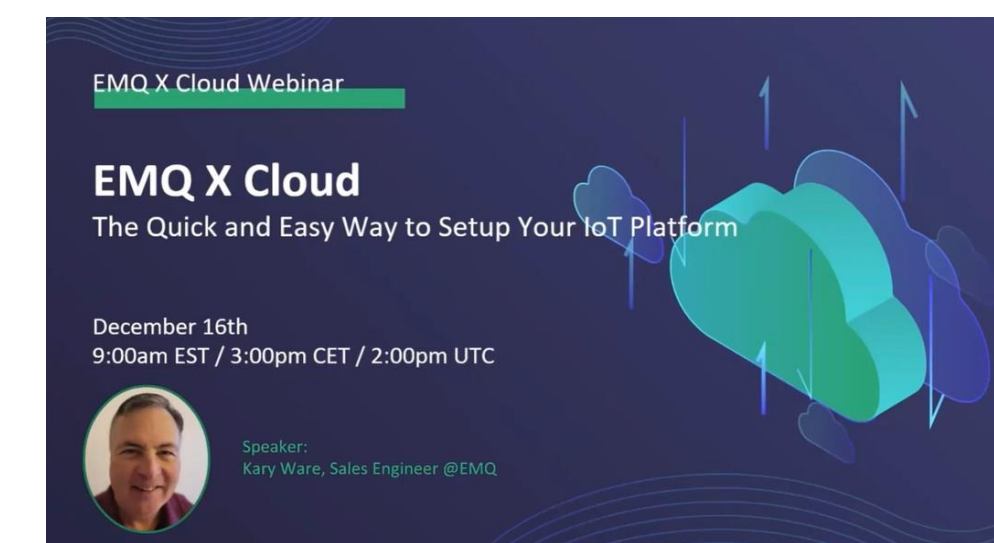
Unlimited



Contact business for customized solutions

Contact Us

- ✓ Physical resource isolation
- ✓ Device management
- ✓ Device model
- ✓ Device Shadow
- ✓ Cloud Edge Collaboration
- ✓ SLA 99.99%
- ✓ Consulting service



www.youtube.com/watch?v=-ybaUISiSdE

EMQ X on-prem

Both are high performance fully 100% MQTT compliant message brokers

Rule Engine: Process messages in real-time

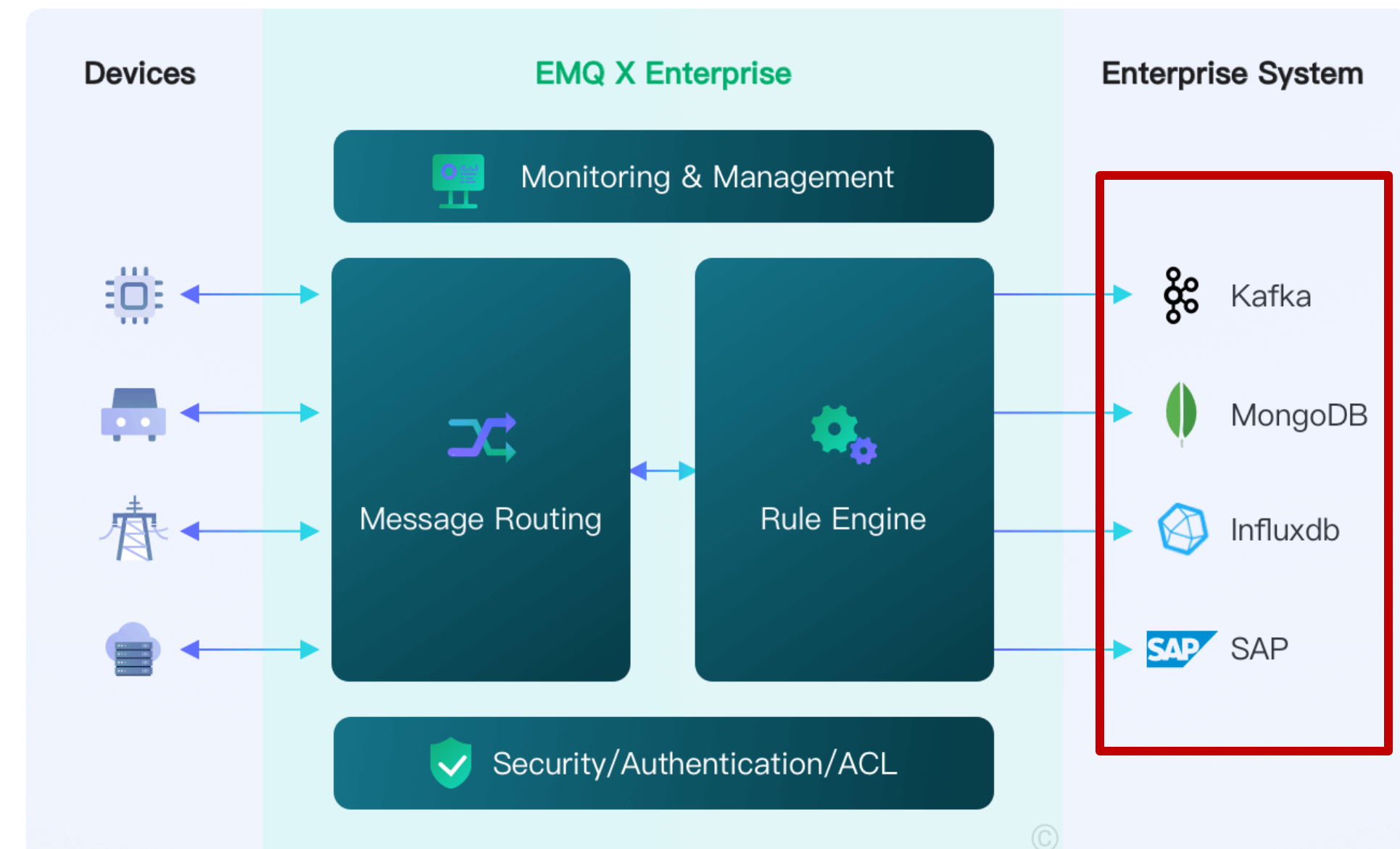
EMQ X Broker/Community Edition

- Open source
- High performance
- Real-time message processing
- 100K connections per node
- TLS/SSL Security



EMQ X Enterprise

- Based on Broker
- 1M connections per node
- Data persistence to many popular databases
- Kafka bridge

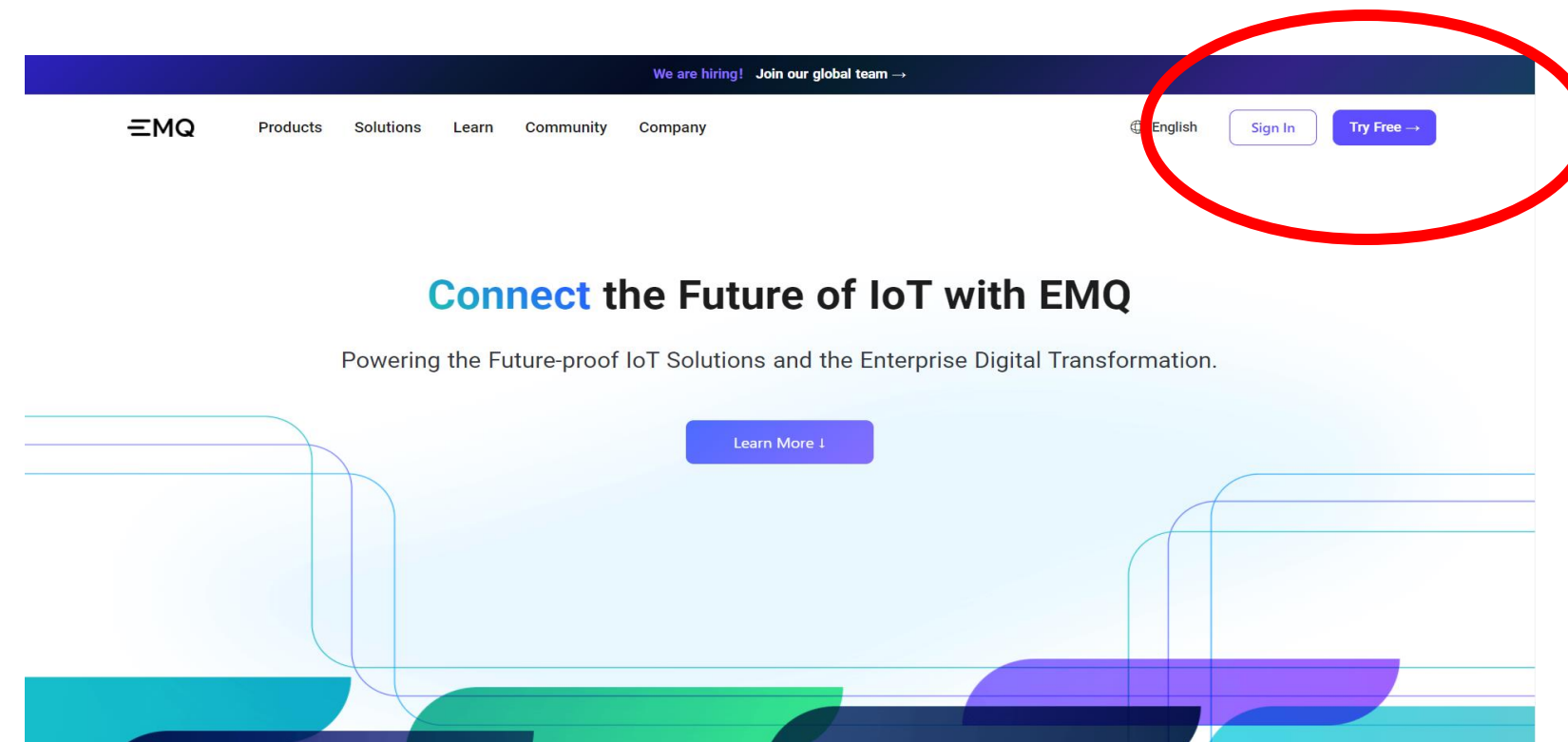


Get a free trial

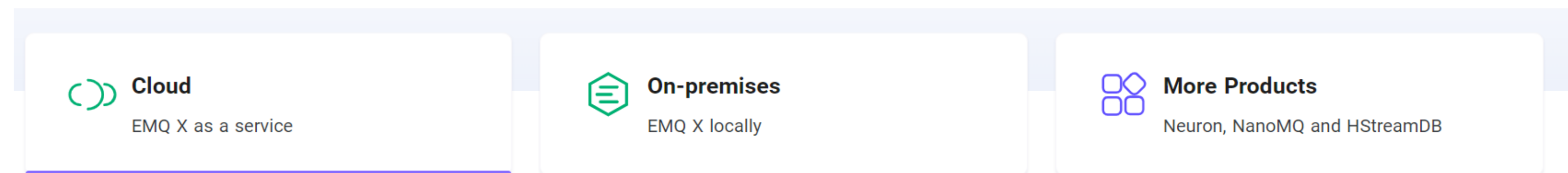
Try Free →

Go to www.emqx.com

and click **Try Free**



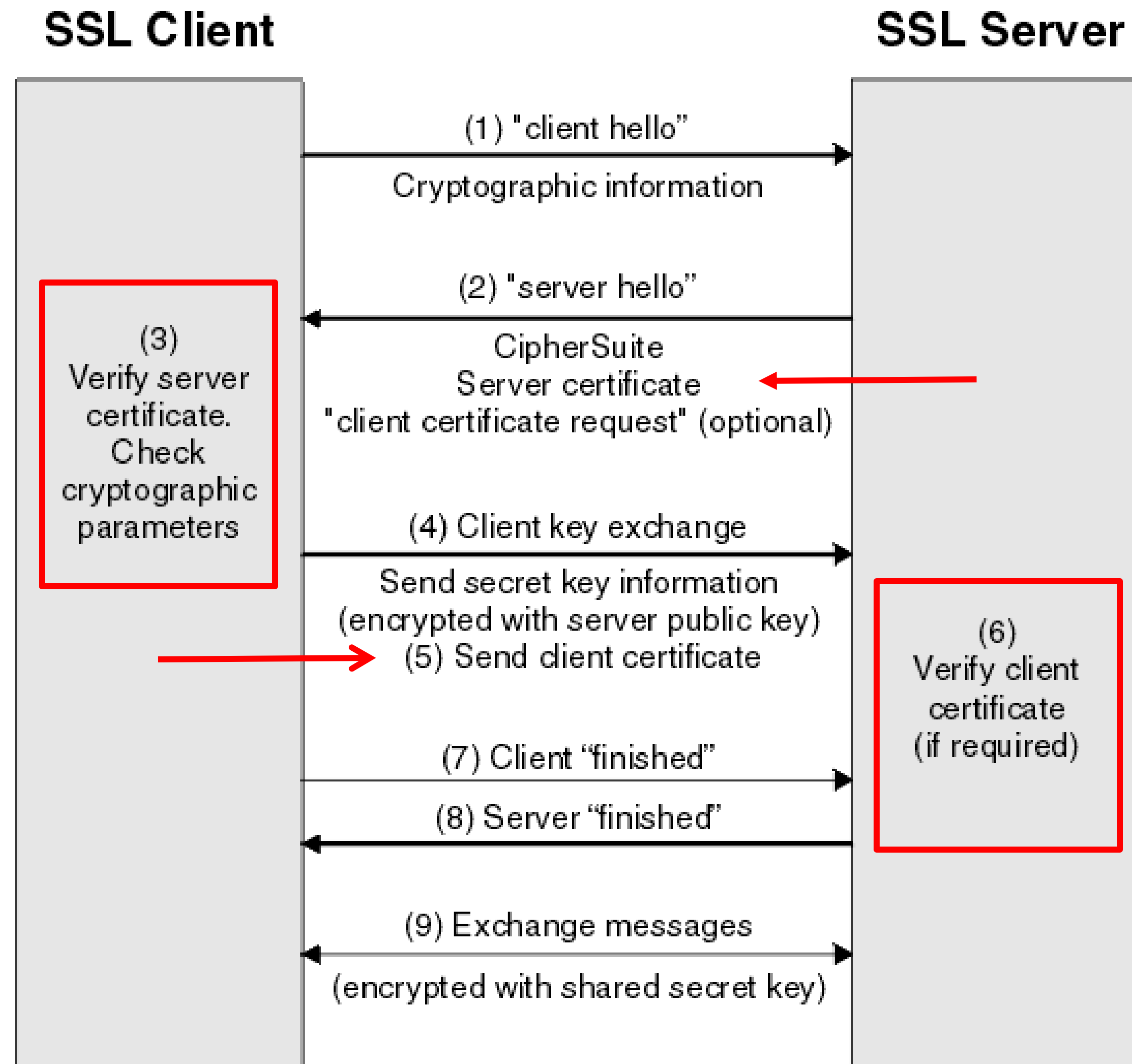
Choose **Cloud** or On-Premises



Follow the instructions

You don't need a credit card!

Overview of SSL/TLS security



Two main levels of security in MQTT and EMQ X



Username/password and client ID

- Client sends username/password and client ID in MQTT CONNECT packet.
- EMQ X broker validates client username/password or client ID using internal or external database such as MongoDB, MySQL, PostgreSQL, Redis
- This can be turned off for testing purposes

Transport Layer Security (TLS): encryption + authentication using trust certificates

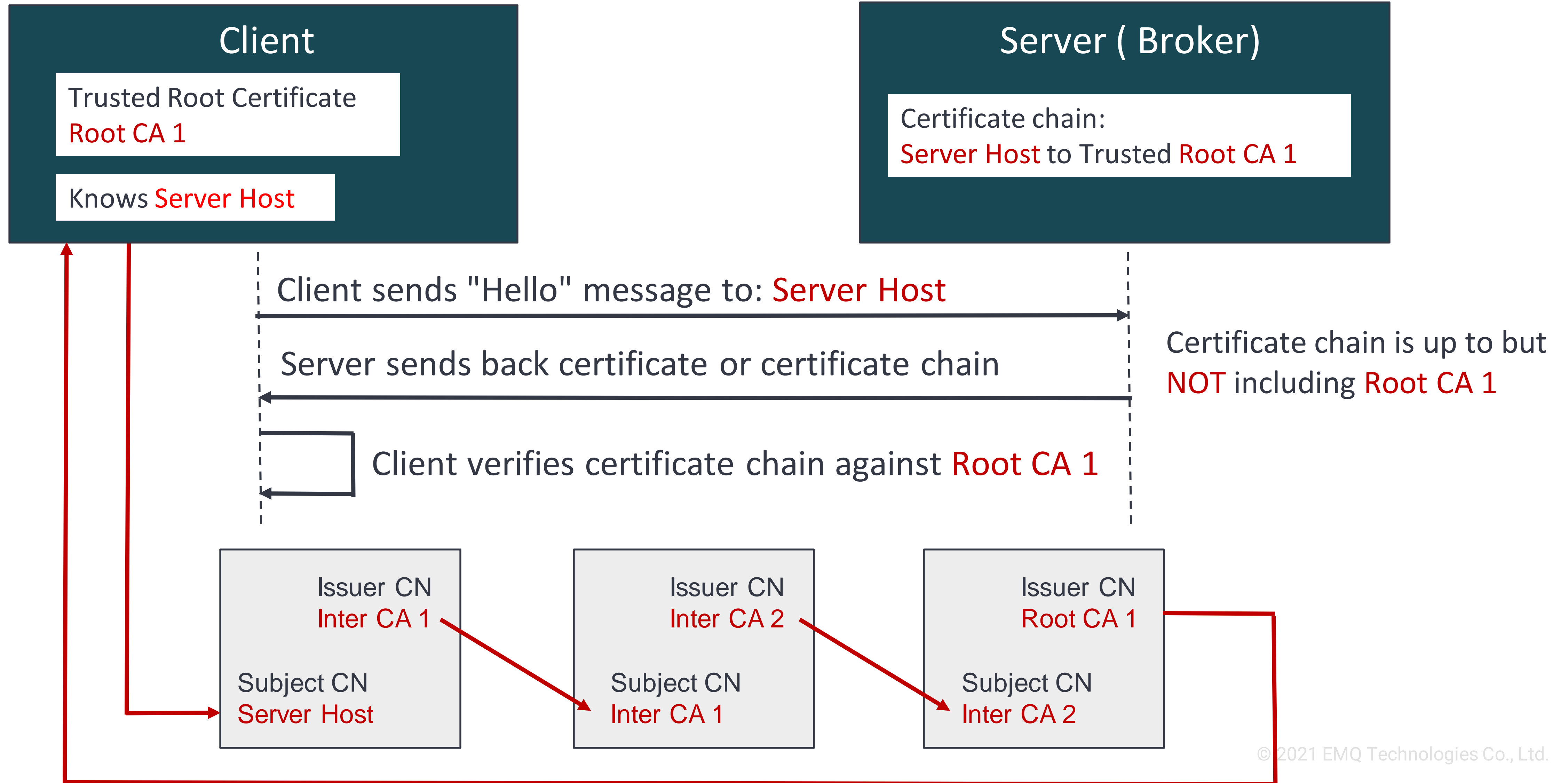
- One-way TLS: Client verifies identity of the server
- Two-way TLS (mTLS): Server also verifies the identity of the client

TLS security and certificates are the focus of this webinar

One-way TLS example



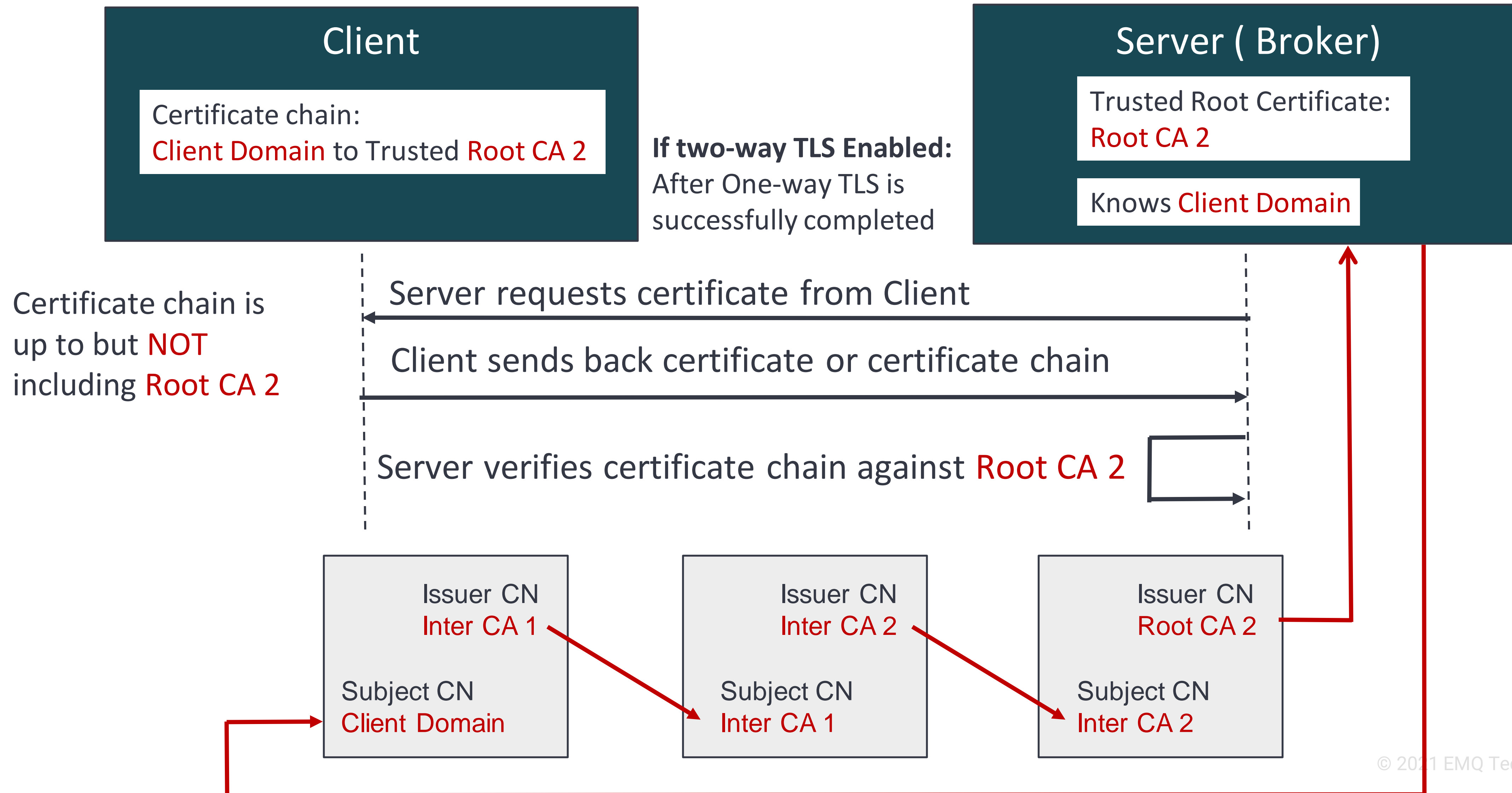
Client verifies identity of the server (broker) using trusted certificate chain



Two-way TLS example



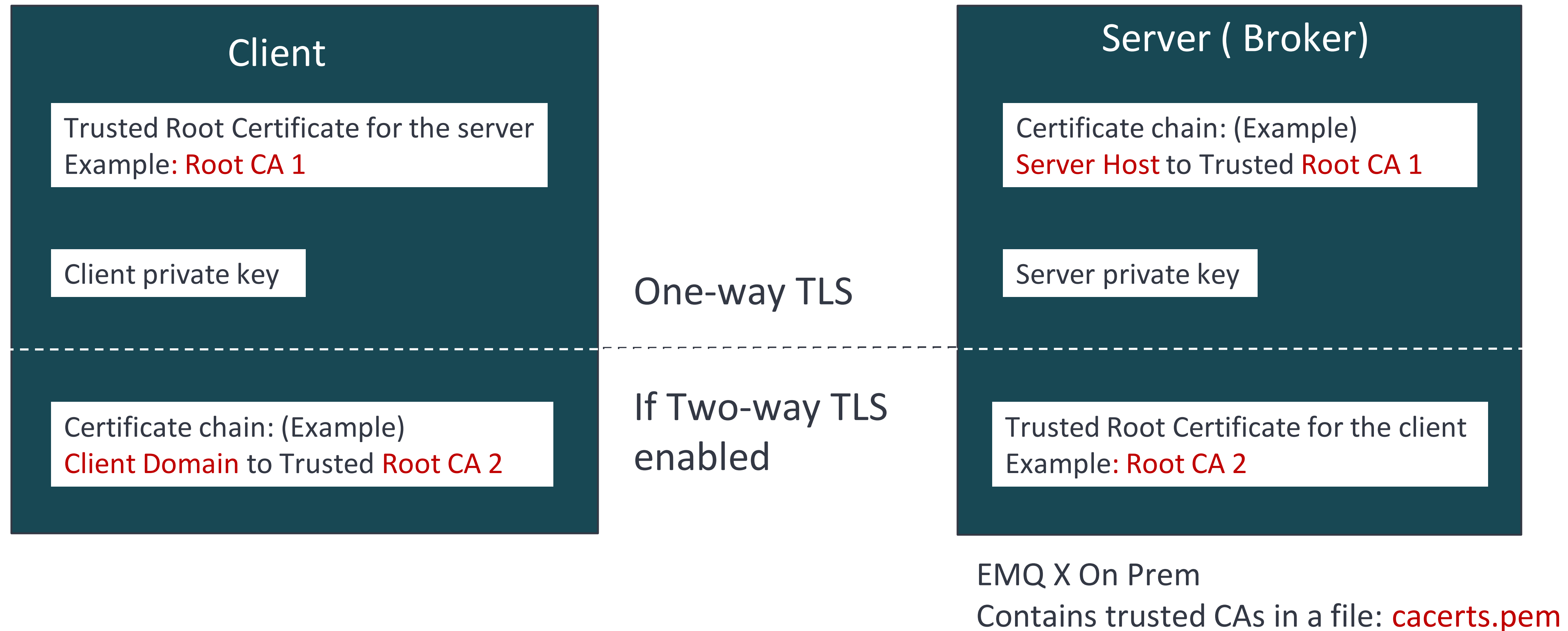
Server also verifies identity of the client using trusted certificate chain



Setting up clients and server for TLS



You will need to provide the clients and server with the following information



You provide this information through public key certificate files and private key files

TLS certificate file examples



Trusted CA Let's Encrypt

cert.pem chain.pem fullchain.pem privkey.pem isrgrootx1.pem

Self-signed These will be generated in the demo and tutorial

ca.pem client-fullchain.pem client.key client.pem server-fullchain.pem server.key server.pem

Generated by EMQ X on-prem Self-signed

cacert.pem cert.pem client-cert.pem client-key.pem key.pem

Located in emqx/etc/certs directory

Next you will see how to examine and understand what these files contain.

How to verify certificates and certificate chains using openssl

Part 1

Part 1:

Examine the certificate files before they are mapped to the EMQ X broker



openssl.org

Part 2 will show how to test the certificates on the running broker

OpenSSL version should be at least 1.1.1

Determine if a file is a certificate or key

Display contents of file

```
cat file_name
```

Example

```
cat server.pem
```

-----BEGIN CERTIFICATE-----

```
MIIDYzCCAkugAwIBAgIHFKIXcQaZEzANBgkqhkiG9w0BAQsFADBtMQswCQYDVQQG
EwJTRTESMBAGA1UECAwJU3....
```

Display the number of certificates in the chain

```
cat server-fullchain.pem | grep -i begin
```

-----BEGIN CERTIFICATE-----

-----BEGIN CERTIFICATE-----

Self signed files used in the demo

```
ca.pem  client-fullchain.pem  client.key  client.pem
server-fullchain.pem  server.key  server.pem
```

```
cat server.key
```

-----BEGIN RSA PRIVATE KEY-----

```
MIIEpAIBAAKCAQEA+R7ZTkUUqHV8me6zRba3BS7yb
o0NNtAr1IUxM9G5wBsGnT0E
F1wOswpueljEjmGHxFh6tpWlsc....
```

```
cat server-fullchain.pem | grep -ic begin
```

2

Verify that certificate and key pair match

Gets public key contained in private key and compares hash against public key certificate file

```
openssl pkey -in KEY_FILENAME -pubout -outform pem | sha256sum  
openssl x509 -in CERT_FILENAME -pubkey -noout -outform pem | sha256sum
```

Example

```
openssl pkey -in server.key -pubout -outform pem | sha256sum  
openssl x509 -in server.pem -pubkey -noout -outform pem | sha256sum
```

```
./verify.sh server
```

```
./key-cert-verify.sh server.key server.pem
```

```
key - server.key: a3d3fd1961aeefbf568a3e436926adaa494d51048e39771e306e1628fea009cc  
cert - server.pem: a3d3fd1961aeefbf568a3e436926adaa494d51048e39771e306e1628fea009cc  
Result: OK
```

```
./key-cert-verify.sh client.key server.pem
```

```
key - client.key: 8f5cf72e1d3d27397ee7f74c9aecece39173c0581bfb13d9eb0edf220d393772  
cert - server.pem: a3d3fd1961aeefbf568a3e436926adaa494d51048e39771e306e1628fea009cc  
Result: Fail - Not a valid certificate and private key pair
```

Trace certificate chain in a file

Displays all the subjects and issuers in a certificate file

```
openssl crl2pkcs7 -nocrl -certfile CERT_FILENAME | openssl pkcs7 -print_certs -noout
```

Example

```
openssl crl2pkcs7 -nocrl -certfile server-fullchain.pem | openssl pkcs7 -print_certs -noout
```

```
./trace.sh server-fullchain.pem
```

subject=C = SE, ST = Stockholm, O = MyOrgName, OU = MyService, **CN** = localhost

issuer=C = SE, ST = Stockholm, O = MyOrgName, OU = MyIntermediateCA, **CN** = MyIntermediateCA-1

subject=C = SE, ST = Stockholm, O = MyOrgName, OU = MyIntermediateCA, **CN** = MyIntermediateCA-1

issuer=C = SE, ST = Stockholm, L = Stockholm, O = MyOrgName, OU = MyRootCA, **CN** = MyRootCA



Determine if a root CA is in EMQ X cacerts.pem file

Displays all the subjects and issuers in a certificate file

```
openssl crl2pkcs7 -nocrl -certfile cacerts.pem | openssl pkcs7 -print_certs -noout
```

```
./trace.sh cacerts.pem
```

```
/opt/emqx/lib/certifi-2.8.0/priv/cacerts.pem
```

subject=C = GR, O = Hellenic Academic and Research Institutions CA, CN = HARICA TLS RSA Root CA 2021

issuer=C = GR, O = Hellenic Academic and Research Institutions CA, CN = HARICA TLS RSA Root CA 2021

subject=C = GR, O = Hellenic Academic and Research Institutions CA, CN = HARICA TLS ECC Root CA 2021

issuer=C = GR, O = Hellenic Academic and Research Institutions CA, CN = HARICA TLS ECC Root CA 2021

```
./trace.sh /opt/emqx/lib/certifi-2.8.0/priv/cacerts.pem | grep "ISRG Root"
```

Root CA for Let's Encrypt

subject=C = US, O = Internet Security Research Group, CN = ISRG Root X1

issuer=C = US, O = Internet Security Research Group, CN = ISRG Root X1

Show certificate file contents

Displays the contents of all certificates in a file

```
openssl crl2pkcs7 -nocrl -certfile CERT_FILENAME | openssl pkcs7 -print_certs -noout -text
```

Example

Same as previous example but with **-text** option

```
openssl crl2pkcs7 -nocrl -certfile server-fullchain.pem | openssl pkcs7 -print_certs -noout -text
```

```
./show.sh server-fullchain.pem
```

Result

Certificate:

Data:

Version: 3 (0x2)

Serial Number: 6265117935573267 (0x16421771069913)

Signature Algorithm: sha256WithRSAEncryption

Issuer: C=SE, ST=Stockholm, O=MyOrgName, OU=MyIntermediateCA,

CN=MyIntermediateCA-1

Validity

Not Before: Jan 14 16:18:27 2022 GMT

Not After : Jan 12 16:18:27 2032 GMT

Subject: C=SE, ST=Stockholm, O=MyOrgName, OU=MyService, CN=localhost

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public-Key: (2048 bit)

Modulus:

00:f9:1e:d9:4e:45:14:a8:75:7c:99:ee:b3:45:b6:

....

Note:

This command works for files with one certificate but **NOT** with multiple certificates:

```
openssl x509 -text -noout -in CERT_FILENAME
```

Usage examples....

Display X509 extensions

```
openssl crl2pkcs7 -nocrl -certfile server-fullchain.pem | openssl pkcs7 -print_certs -noout -text | grep -iA 4 X509 --color
```

```
./show.sh server-fullchain.pem | grep -iA4 X509 --color
```

--colour works too

X509v3 extensions:

X509v3 Subject Alternative Name:

DNS:s2da4b72-internet-facing-e....daf.elb.eu-west-1.amazonaws.com

← Subject Alternative Name

Signature Algorithm: sha256WithRSAEncryption

89:74:3f:5d:78:15:85:20:4b:c2:8c:8b:c2:ea:4c:77:0e:e9:

2d:11:29:9c:63:1d:4e:eb:0a:0d:6f:c1:a3:72:38:39:35:d3:

--

X509v3 extensions:

X509v3 Key Usage: critical

Certificate Sign, CRL Sign

X509v3 Basic Constraints: critical

CA:TRUE

X509v3 Subject Key Identifier:

65:F1:3A:AA:67:67:A5:D7:DC:06:85:5E:A9:0E:3D:25:B7:AD:34:91

X509v3 Authority Key Identifier:

DirName:/C=SE/ST=Stockholm/L=Stockholm/O=MyOrgName/OU=MyRootCA/CN=MyRootCA

serial:50:66:52:FC:DC:68:9A:98:73:34:D6:BC:06:BC:73:92:BA:F2:03:46

Signature Algorithm: sha256WithRSAEncryption

Display certificate validity

Display validity dates of all certificates in the file

```
openssl crl2pkcs7 -nocrl -certfile server-fullchain.pem | openssl pkcs7 -print_certs -noout -text | grep -iA 2 validity
```

```
./show.sh server-fullchain.pem | grep -iA2 validity --color
```

Validity

Not Before: Jan 14 16:18:27 2022 GMT

Not After : Jan 12 16:18:27 2032 GMT

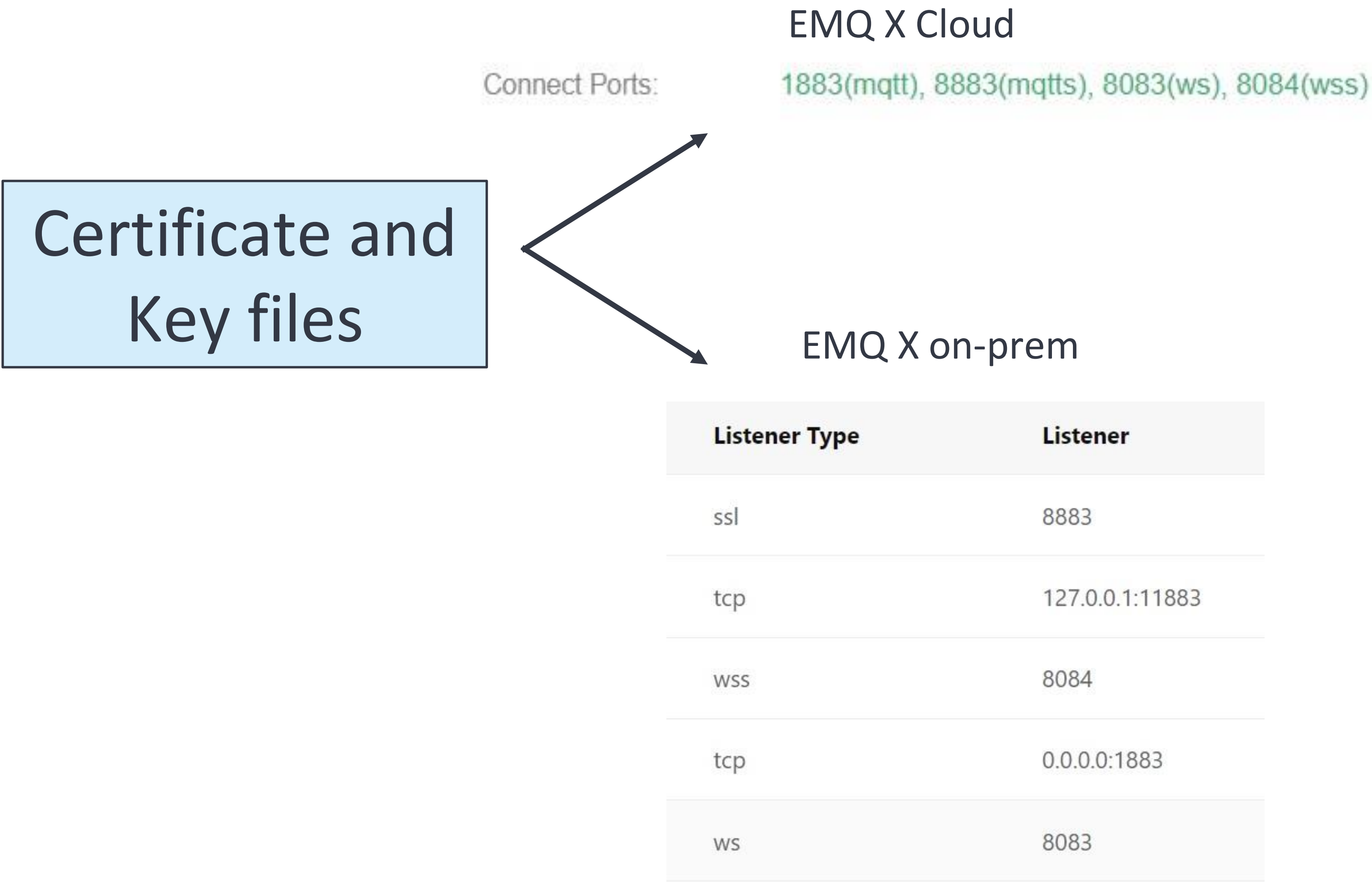
--

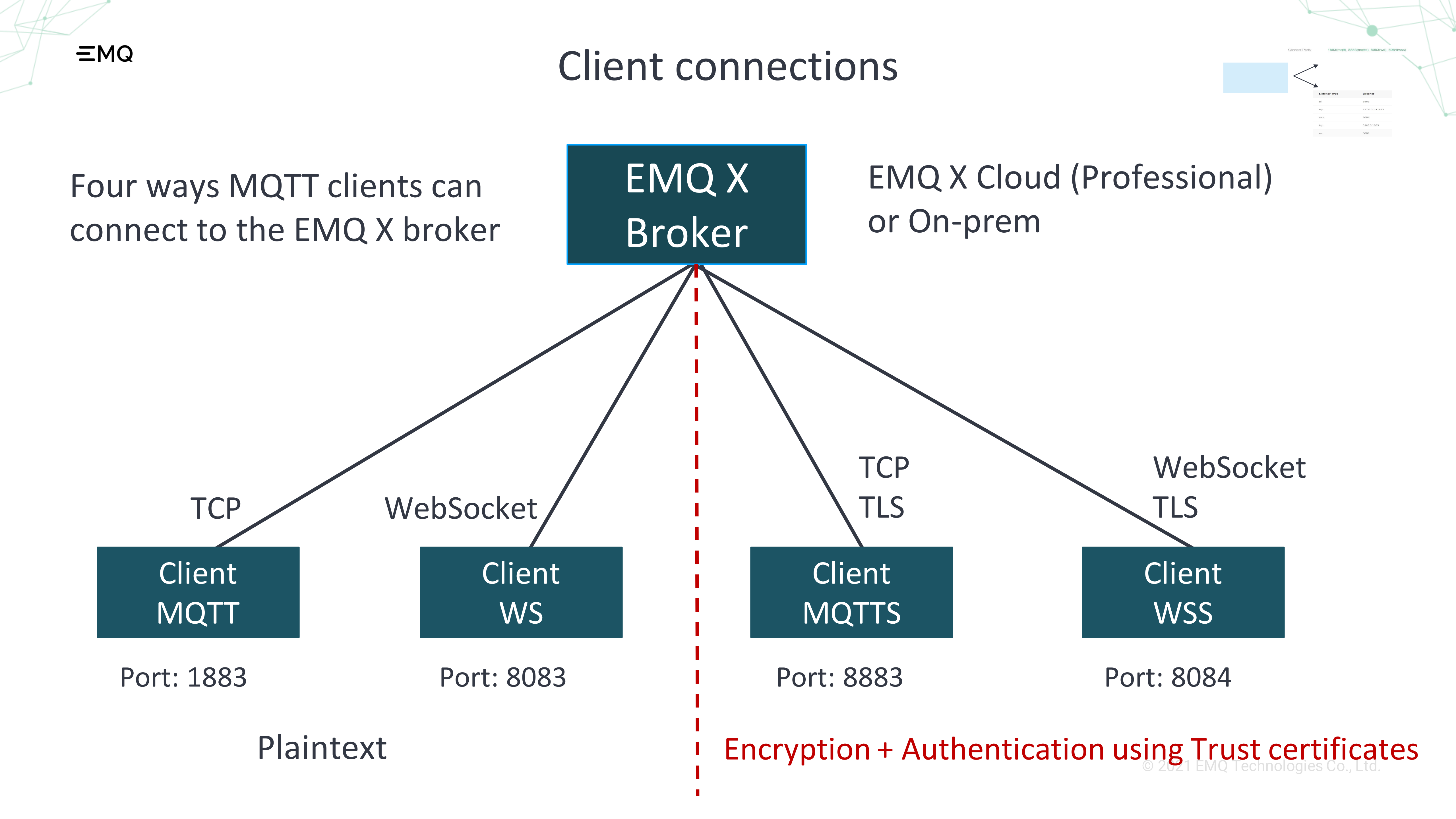
Validity

Not Before: Jan 14 16:18:26 2022 GMT

Not After : Jan 12 16:18:26 2032 GMT

How to map certificate files to EMQ X listeners







EMQ X dashboard (on-prem)

Two ways to connect to the EMQ X dashboard

📘 localhost:18083/#/monitor

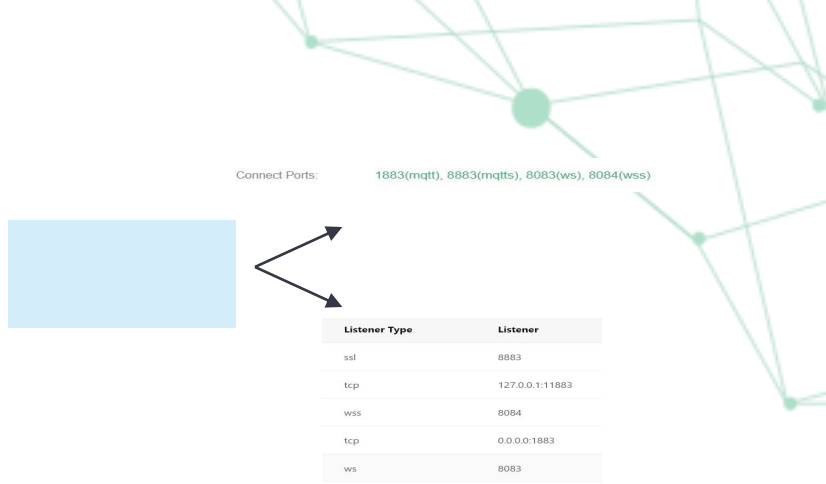
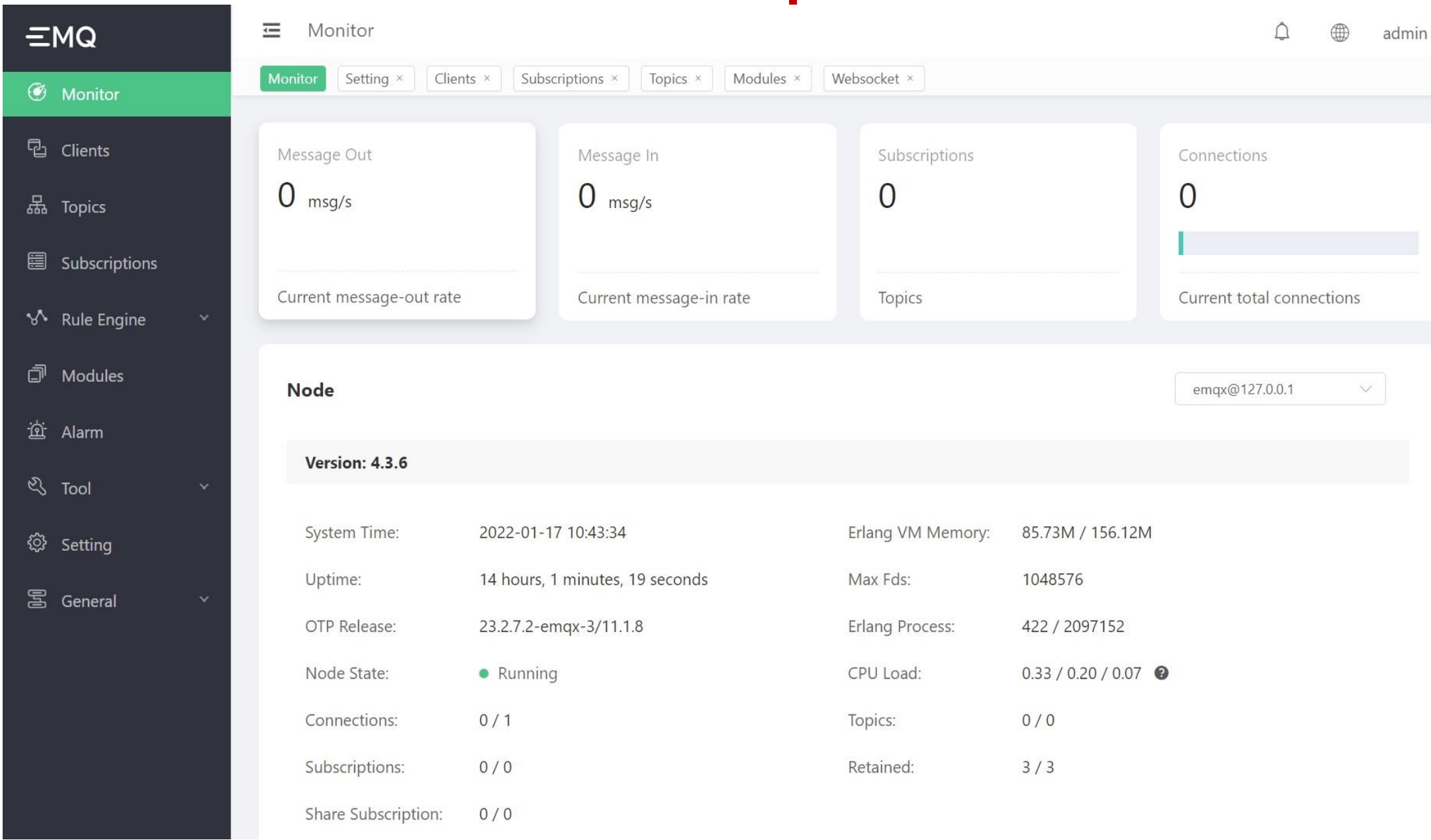
HTTP
Port: 18083

Plaintext

🔒 localhost:18084/#/monitor

HTTPS
Port: 18084

Encryption + Authentication
using Trust certificates



EMQ X Cloud

Deployment > Overview > Connect Ports

Connect Ports: 1883(mqtt), 8883(mqtts), 8083(ws), 8084(wss)

For MQTTS (SSL) and WSS to work properly:

The EMQ X broker needs to have
the correct certificate files.

EMQ X on-prem

Dashboard > Settings > Listeners

Listener Name	Listener Type	Listener
external	<u>ssl</u>	8883
internal	tcp	127.0.0.1:11883
external	<u>wss</u>	8084
external	tcp	0.0.0.0:1883
external	ws	8083

Mapping of EMQ X Cloud certificate files

Maps to:

File containing the server (broker) certificate

Together they need to form a valid chain
(Can overlap)

File containing all intermediate server CAs (if any), but **NOT** the root CA

File containing the server (broker) private key

Example

server.pem

subject: ServerHost
issuer: MyInterCA-1

server-chain.pem

subject: MyInterCA-1
issuer: MyRootCA

server.key

Server Private Key

For two-way TLS

File containing the **client** root CA

ca.pem

subject: MyRootCA
issuer: MyRootCA

TLS/SSL Config

* Certificate body

1

upload PEM-encoded

Certificate chain

1

upload PEM-encoded

* Certificate private key

1

upload PEM-encoded

* Client CA certificate

1

upload PEM-encoded

Connect Ports:

Listener Type	Listener
ssl	8883
tcp	127.0.0.1:11803
ws	8084
tcp	0.0.0.0:1883
ws	8083

EMQ X Parameter

Maps to:

Example

*.keyfile

File containing the server (broker)
private key

server.key

Server Private Key

*.certfile

File containing certificate chain
from server, all (if any) intermediate
CAs, but **NOT** the root CA

server-fullchain.pem

subject: ServerHost
issuer: MyInterCA-1

subject: MyInterCA-1
issuer: MyRootCA

*.cacertfile

For two-way TLS

File containing root CAs for all **clients**.
Needed only if the root CA is not included
in the EMQ X cacerts.pem file

ca.pem

subject: MyRootCA
issuer: MyRootCA

EMQ X on-prem configuration files example

Enterprise edition

SSL (MQTTS)
Configurations

WSS
Configurations

HTTPS
Configurations

listeners.conf

listener.ssl.external.keyfile

listener.ssl.external.certfile

listener.ssl.external.cacertfile

listener.wss.external.keyfile

listener.wss.external.certfile

listener.wss.external.cacertfile

plugins/emqx_dashboard.conf

dashboard.listener.https.keyfile

dashboard.listener.https.certfile

dashboard.listener.https.cacertfile

Server Private Key

subject: ServerHost

issuer: MyInterCA-1

subject: MyInterCA-1

issuer: MyRootCA

subject: MyRootCA

issuer: MyRootCA

Root CA for Clients

Community Edition: emqx.conf

server.key

server-fullchain.pem

ca.pem

server.key

server-fullchain.pem

ca.pem

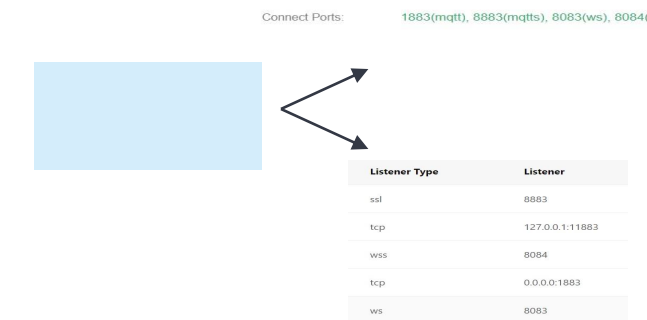
server.key

server-fullchain.pem

ca.pem

© 2021 EMQ Technologies Co., Ltd.

Mapping EMQ X self-generated on-prem files



Listener Type	Listener
ssl	8883
tcp	127.0.0.1:11803
ws	8084
tcp	0.0.0.0:1883
ws	8083

Located in emqx/certs/ cacert.pem cert.pem client-cert.pem client-key.pem key.pem

No intermediate CAs

Root CA is **same** for both server and client

Server files

key.pem

Server Private Key

Maps to

***.keyfile**

cert.pem

subject: Server
issuer: RootCA

Maps to

***.certfile**

cacert.pem

subject: RootCA
issuer: RootCA

Maps to

Needed only for
Two-way TLS
***.cacertfile**

Files for Clients

client-key.pem

Client Private Key

cacert.pem

subject: RootCA
issuer: RootCA

Root CA of **server**

Needed only for **Two-way TLS**

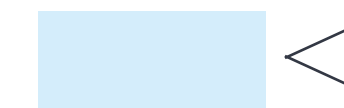
client-cert.pem

subject: Client
issuer: RootCA

Root CA of **client**

Mapping EMQ X self-generated on-prem files EMQ X Cloud

Connect Ports: 1883(mqtt), 8083(mqtts), 8083(wss), 8084(wss)



Listener Type	Listener
ssl	8083
tcp	127.0.0.1:1883
wss	8084
tcp	0.0.0.0:1883
wss	8083

cert.pem

subject: Server
issuer: RootCA

Maps to

* Certificate body

1

upload PEM-encoded

Certificate chain

1

upload PEM-encoded

* Certificate private key

1

upload PEM-encoded

* Client CA certificate

1

upload PEM-encoded

Files for **Clients**

client-key.pem

Client Private Key

cacert.pem

subject: RootCA
issuer: RootCA

Root CA of **server**

Needed only for **Two-way TLS**

client-cert.pem

subject: Client
issuer: RootCA

emqx/certs/

cacert.pem cert.pem key.pem
client-cert.pem client-key.pem

key.pem

Server Private Key

Maps to

cacert.pem

subject: RootCA
issuer: RootCA

Two-Way TLS

Maps to

On-prem

For One-Way TLS: *.verify = verify_none

For Two-Way TLS: *.verify = verify_peer

*.fail_if_no_peer_cert = true

Cloud

TLS/SSL Config ?

★ TLS/SSL type

one-way

one-way

two-way

File	Parameter	One-Way TLS	Two-Way TLS
listeners.conf/emqx.conf	listener.ssl.external.verify	verify_none	verify_peer
SSL	listener.ssl.external.fail_if_no_peer_cert		true
listeners.conf/emqx.conf	listener.wss.external.verify	verify_none	verify_peer
WSS	listener.wss.external.fail_if_no_peer_cert		true
emqx_dashboard.conf	listener.https.external.verify	verify_none	verify_peer
HTTPS	listener.https.external.fail_if_no_peer_cert		true

Example:
Mapping
Lets Encrypt
certificates to
EMQ X



Let's Encrypt

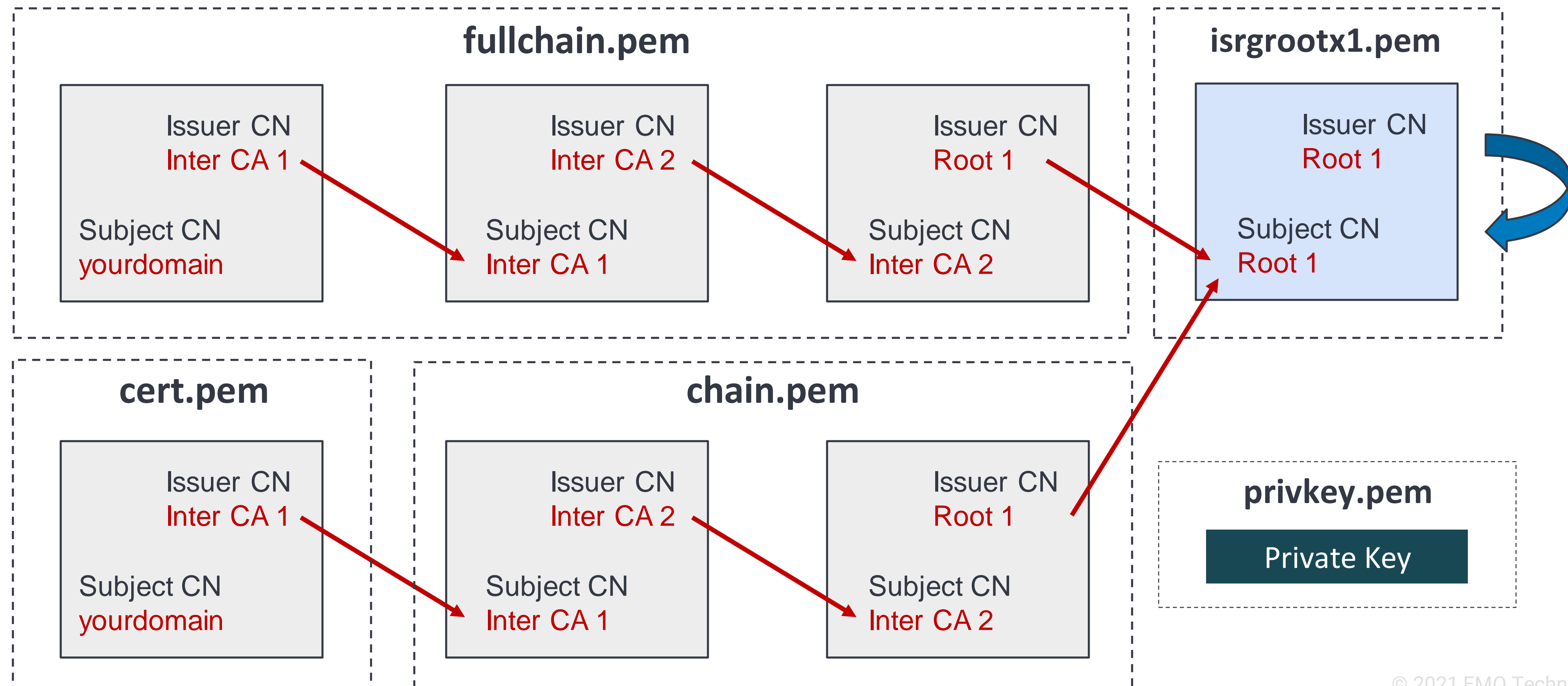
letsencrypt.org

Let's Encrypt files

Files located at: /etc/letsencrypt/live/**yourdomain**/ `cert.pem` `chain.pem` `fullchain.pem` `privkey.pem`

Root certificate file not included but can be downloaded manually

Root certificate is contained in EMQ X (on-prem) `cacerts.pem` file



Mapping Let's Encrypt files to EMQ X on-prem

cert.pem chain.pem fullchain.pem privkey.pem

privkey.pem

Server Private Key

Maps to

EMQ X Parameters
listeners.conf
plugins/emqx_dashboard.conf

***.keyfile**

fullchain.pem

subject: yourdomain
issuer: R3

subject: R3
issuer: ISRG Root X1

subject: ISRG Root X1
issuer: DST Root CA X3

Maps to

***.certfile**

One-Way TLS

Client needs to have either
ISRG Root X1
DST Root CA X3

Two-Way TLS

Clients need corresponding
fullchain.pem for client
privkey.pem for client

EMQ X cacerts.pem file
contains both
ISRG Root X1
DST Root CA X3

Mapping Let's Encrypt files to EMQ X Cloud

cert.pem chain.pem fullchain.pem privkey.pem

cert.pem

subject: yourdomain
issuer: R3

Maps to

* Certificate body

1

upload PEM-encoded

chain.pem or fullchain.pem

subject: R3
issuer: ISRG Root X1

subject: ISRG Root X1
issuer: DST Root CA X3

Maps to

Certificate chain

1

upload PEM-encoded

privkey.pem

Server Private Key

Maps to

* Certificate private key

1

upload PEM-encoded

xxx.pem

subject: Client RootCA
issuer: Client RootCA

Root CA of **client**

Two-Way TLS

Maps to

* Client CA certificate

1

upload PEM-encoded

Files for **Clients**

One-Way TLS

Client needs to have either
ISRG Root X1
DST Root CA X3

Two-Way TLS

EMQ X contains both
ISRG Root X1
DST Root CA X3

How to verify certificates and certificate chains using openssl

Part 2

Part 2:

Verify that certificates are correct
on the running broker.



openssl.org

OpenSSL version should be at least 1.1.1

openssl s_client

Connects to the running broker as a client using SSL/TLS

Useful for troubleshooting and verifying TLS certificates are correct

Example

```
openssl s_client -CAfile CA_FILE -connect HOST:PORT
```

For EMQ X, PORT can be **8883** (SSL), **8084** (WSS), **18084** (HTTPS)

```
openssl s_client -CAfile ca.pem -connect localhost:8883
```

Prints out TLS connection messages

Verify the certificate chain on the server

```
openssl s_client -CAfile ca.pem -connect localhost:8883 2>&1 | grep -iA5 "certificate chain"
```

Certificate chain

0 s:C = SE, ST = Stockholm, O = MyOrgName, OU = MyService, CN = localhost

i:C = SE, ST = Stockholm, O = MyOrgName, OU = MyIntermediateCA, CN = MyIntermediateCA-1

1 s:C = SE, ST = Stockholm, O = MyOrgName, OU = MyIntermediateCA, CN = MyIntermediateCA-1

i:C = SE, ST = Stockholm, L = Stockholm, O = MyOrgName, OU = MyRootCA, CN = MyRootCA

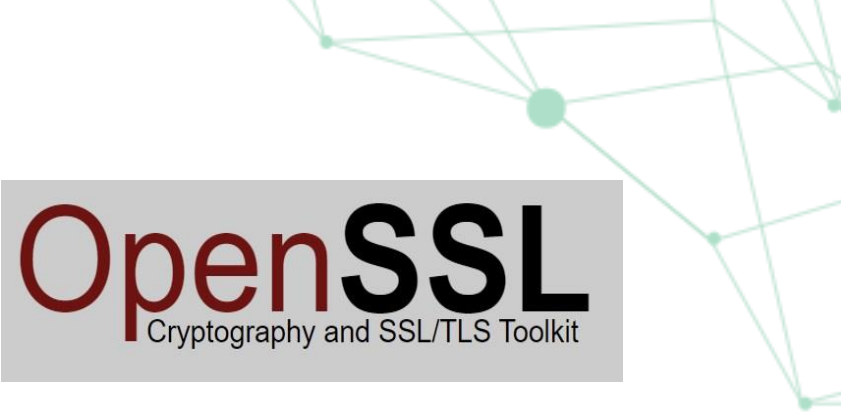
```
grep iAx
```

```
x = (num certs) x 2 + 1
```

s_client writes to both stdout and stderr

Use **2>&1** to force all output to stdout before piping to grep

Otherwise, the output will be: All of stderr plus grep of only stdout



openssl s_client verbose

To turn on verbose output

For EMQ X, PORT can be **8883** (SSL), **8084** (WSS), **18084** (HTTPS)

```
openssl s_client -CAfile CA_FILE -verify_hostname HOST -tlsextdebug -state -debug -connect HOST:PORT
```

```
openssl s_client -CAfile ca.pem -verify_hostname localhost -tlsextdebug -state -debug -connect localhost:8883
```

Can use the following scripts

	Port	EMQ X On-Prem	EMQ X Cloud
SSL	8883	./sclient-ssl.sh	./sclient-ssl.sh cloud
WSS	8084	./sclient-wss.sh	./sclient-wss.sh cloud
HTTPS	18084	./sclient-https.sh	./sclient-https.sh cloud

Then examine the output or grep to find some errors....

Common certificate errors

Certificate chain is broken

```
Cannot verify certificate chain  
Verify return code: 20 (unable to get local issuer certificate)
```

Server hostname does not match certificate

```
Cannot verify hostname  
Verification error: Hostname mismatch
```

Can grep for lines of interest

```
./sclient-wss.sh 2>&1 | grep -i -e err -e verif
```

grep -e option allows for multiple search terms

No certificates

```
SSL_connect:error in SSLv3/TLS write client hello  
write:errno=104  
Verification: OK  
Verify return code: 0 (ok)
```

```
./sclient-wss.sh 2>&1 | grep -i "certificate chain"
```

Returns blank

Hands-on Tutorial

Overview

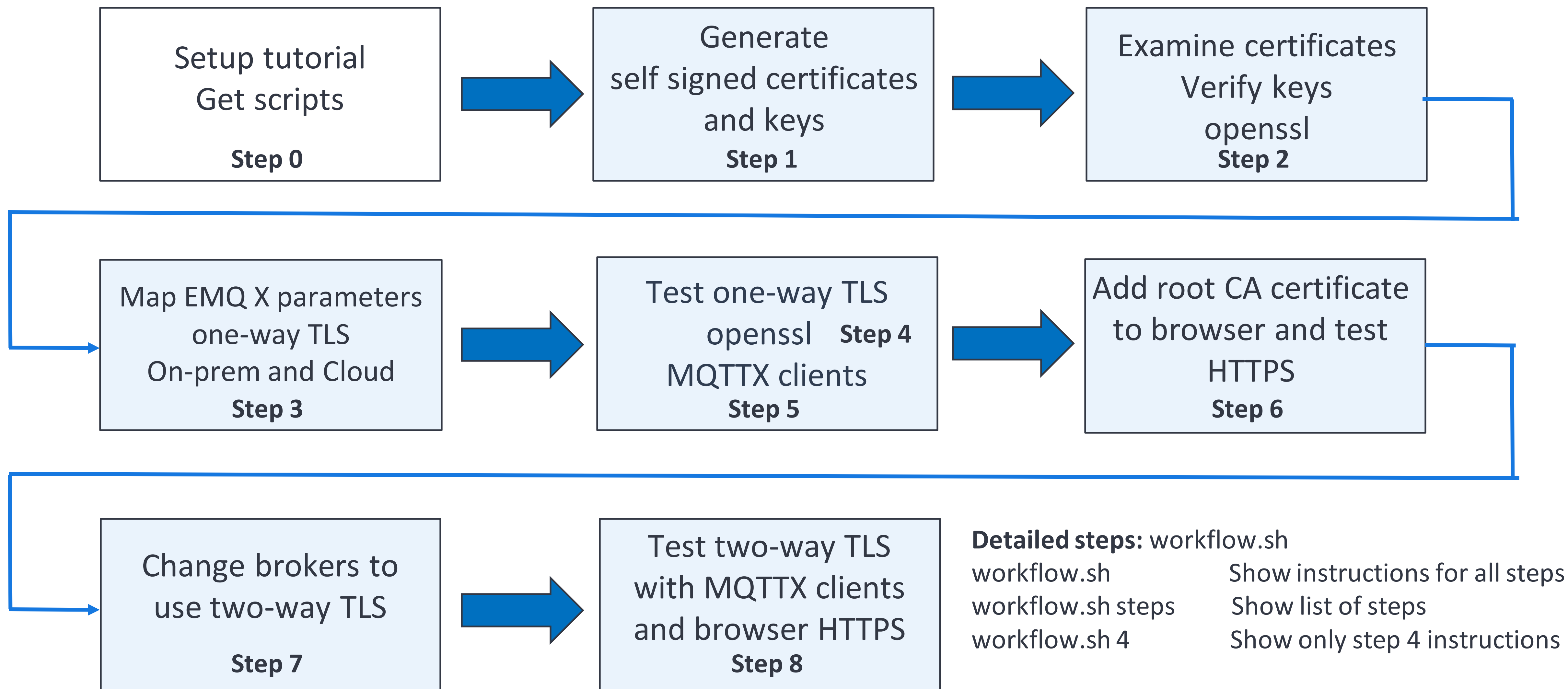
Generating self-signed certificates
and mapping to EMQ X Cloud and on-prem

Verify that they work

One-way TLS and Two-way TLS

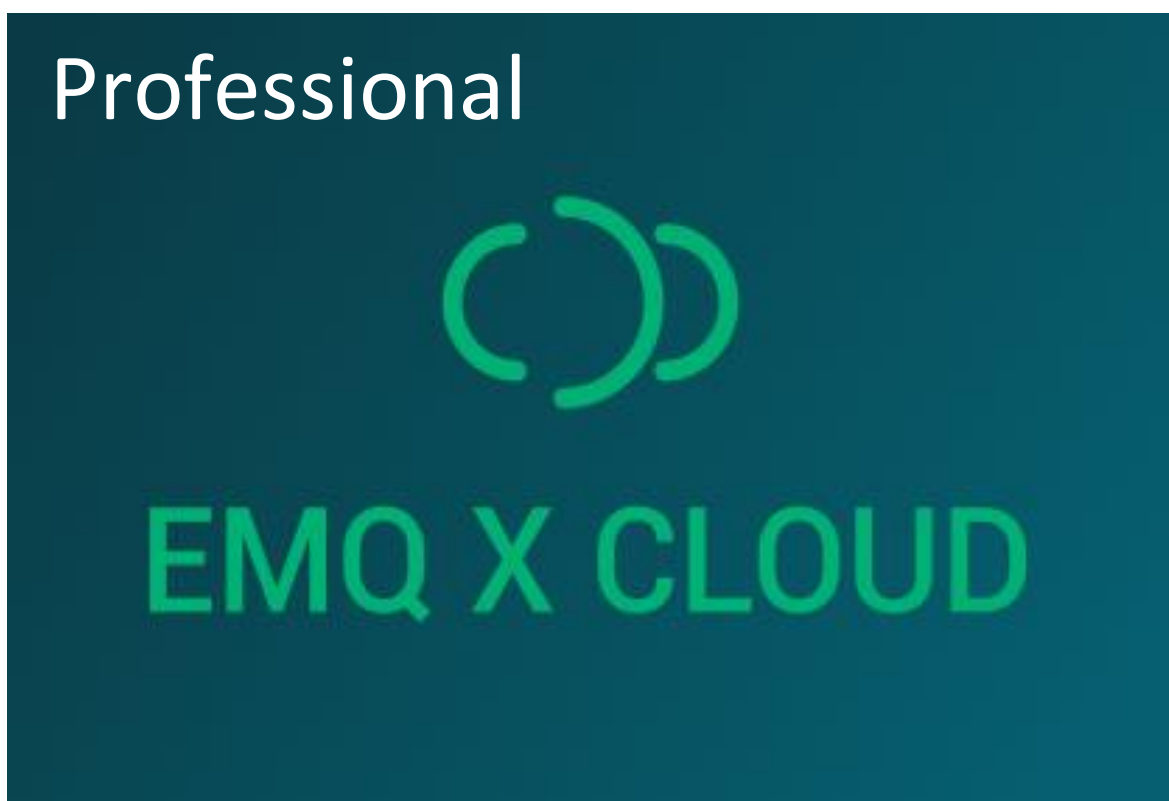
Separate Video

Tutorial Workflow



Demo and tutorial setup

Demo



Docker Ubuntu image
+
EMQ X Ubuntu install

≠

Available Downloads

Version
v4.4.0

OS
Ubuntu / Ubuntu 20.04

Available Downloads

Version
v4.4.0

OS
Docker

EMQ X Docker Install

Alpine Linux
Smaller

Windows Laptop PC

Docker
Ubuntu 20.04
EMQ X Enterprise

 **MQTT X**
MQTT Clients

You can also install EMQ X directly on Linux

Demo

Generate self-signed certificates
Map to EMQ X Cloud and on-prem
One-way TLS

Welcome to join EMQ X Community



<https://slack-invite.emqx.io/>



<https://github.com/emqx/emqx>

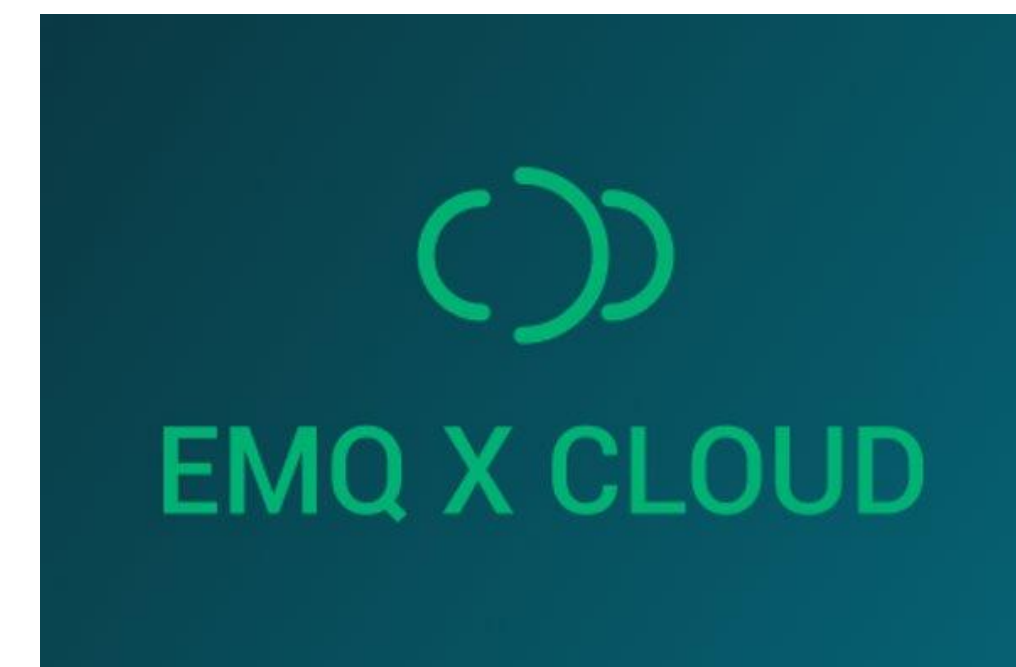
Forum



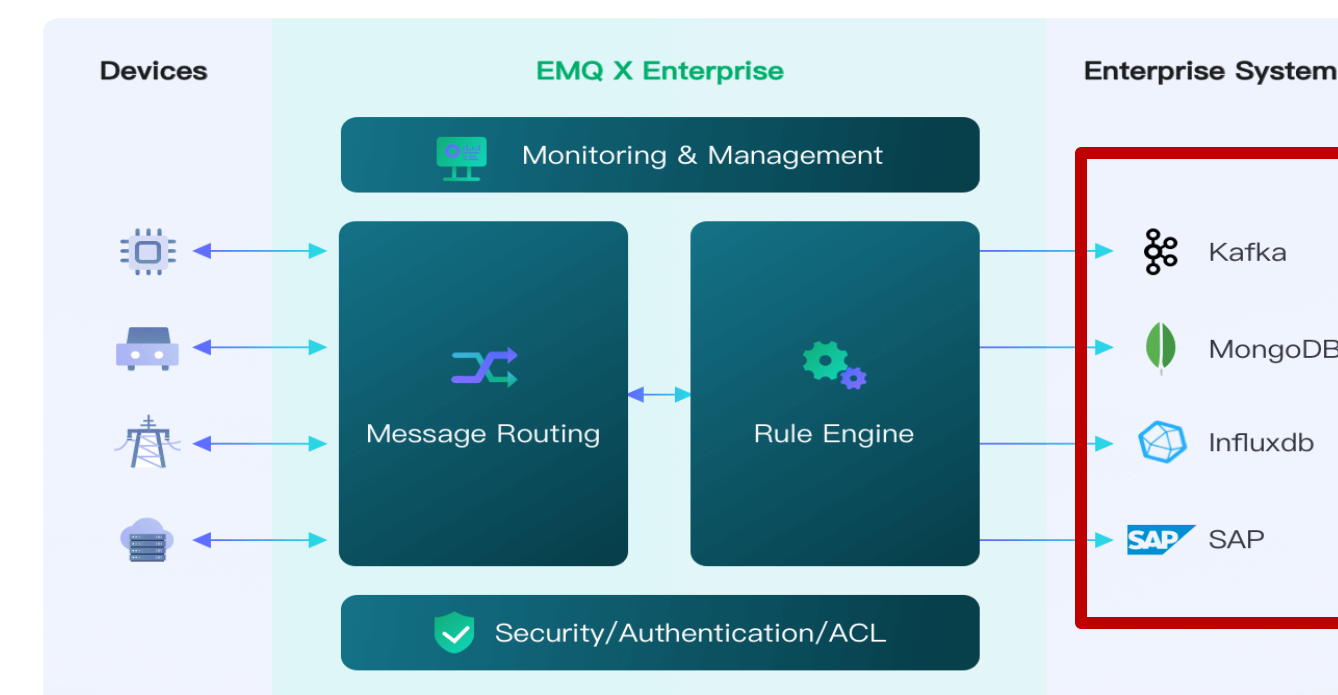
<https://github.com/emqx/emqx/discussions>

Happy to discuss with YOU

Questions and Answers



Q & A



Sign up for your free trial today! www.emqx.com

And try out the demo yourself

Try Free →