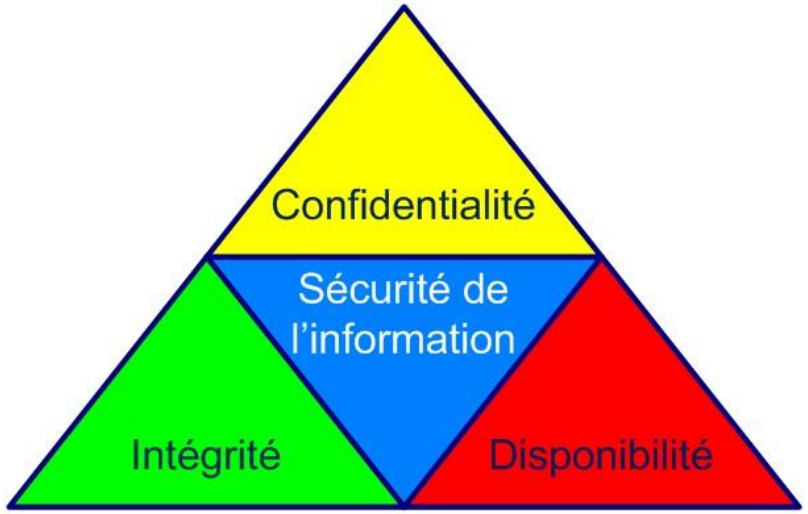
NORME ISO 27001

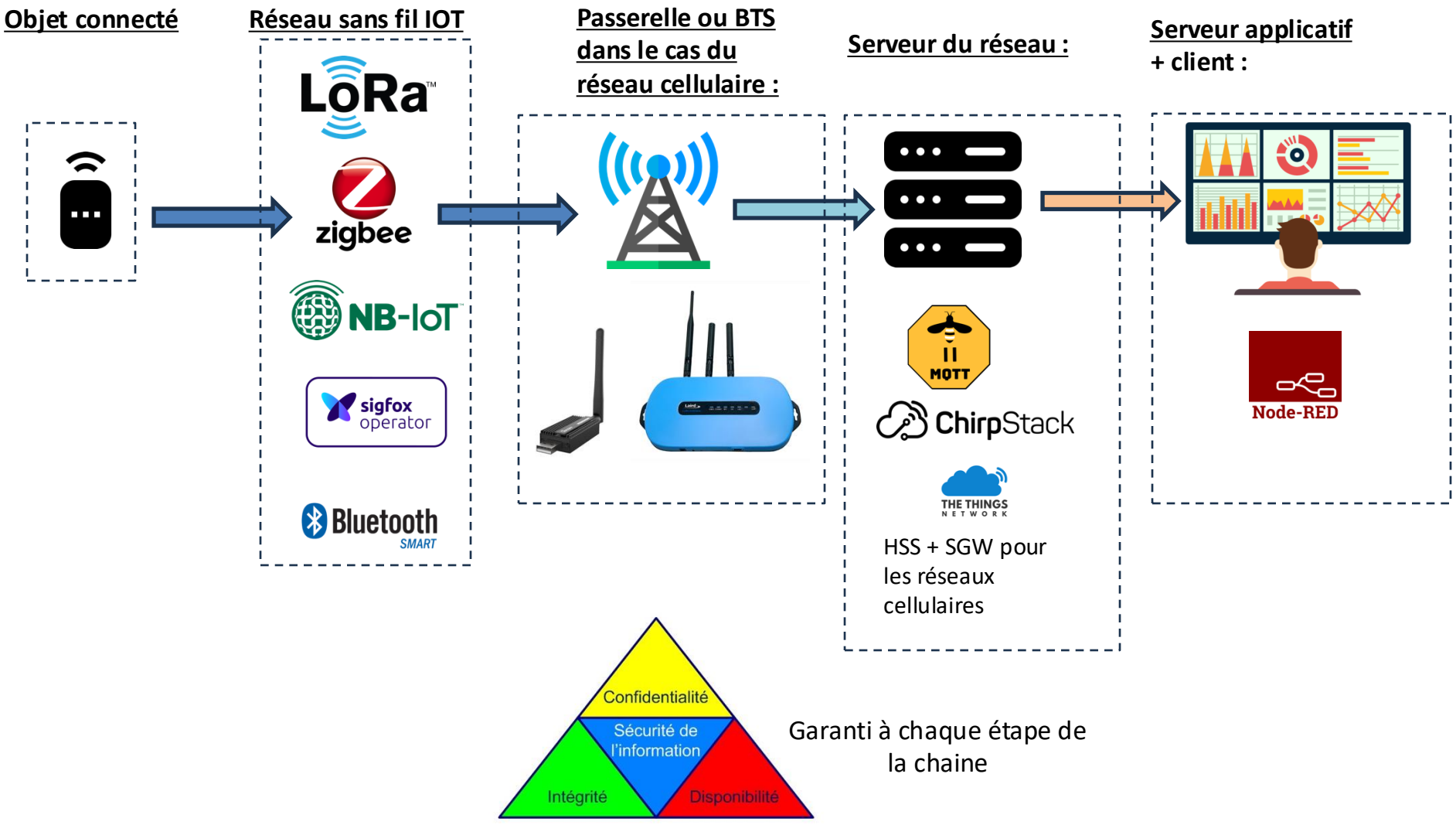
L'**ISO/IEC 27001** est une norme publiée par l'**Organisation Internationale de Normalisation (ISO)** et la **Commission Électrotechnique Internationale (IEC)**. Elle fournit un cadre pour aider les organisations à protéger leurs **informations sensibles** contre les cybermenaces et les violations de données.

Objectif principal : Garantir la **confidentialité, l'intégrité et la disponibilité** des informations, en mettant en place des **contrôles de sécurité adaptés**.



On parle souvent de la triade de la sécurité

SYNOPTIQUE DE LA CHAINE DE VALEUR DES OBJETS CONNECTÉS



VULNÉRABILITÉ SUR LE MQTT

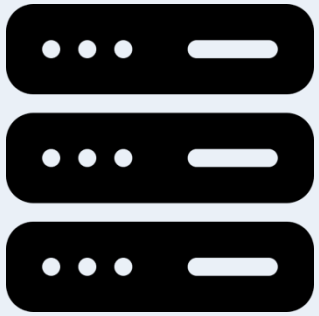
Attaques potentielles

MQTT

mosquitto_sub -h <broker> -t '#' -v

```
mosquitto_sub -h 192.168.1.25 -t '#' -v
```

```
garden/bed2/temperature 17
garden/bed1/temperature 18
garden/bed1/humidity 33.1
home/bedroom 21
```



Serveur du réseau

internet

MQTTS avec Certificats TLS

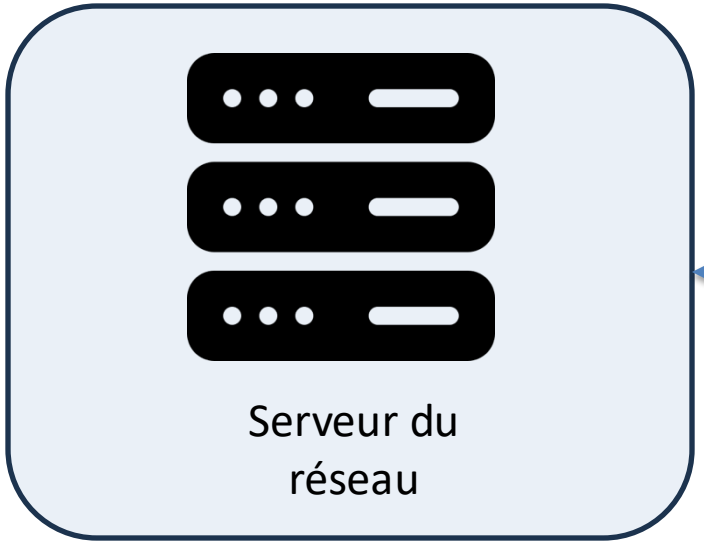
AUTHENTIFICATION DU CLIENT PAR MOT DE PASSE

Authenticité

Attaques potentielles

MQTT

mosquitto_sub -h 192.168.1.25 -u ciel –P passciel -t ‘home/bedroom’

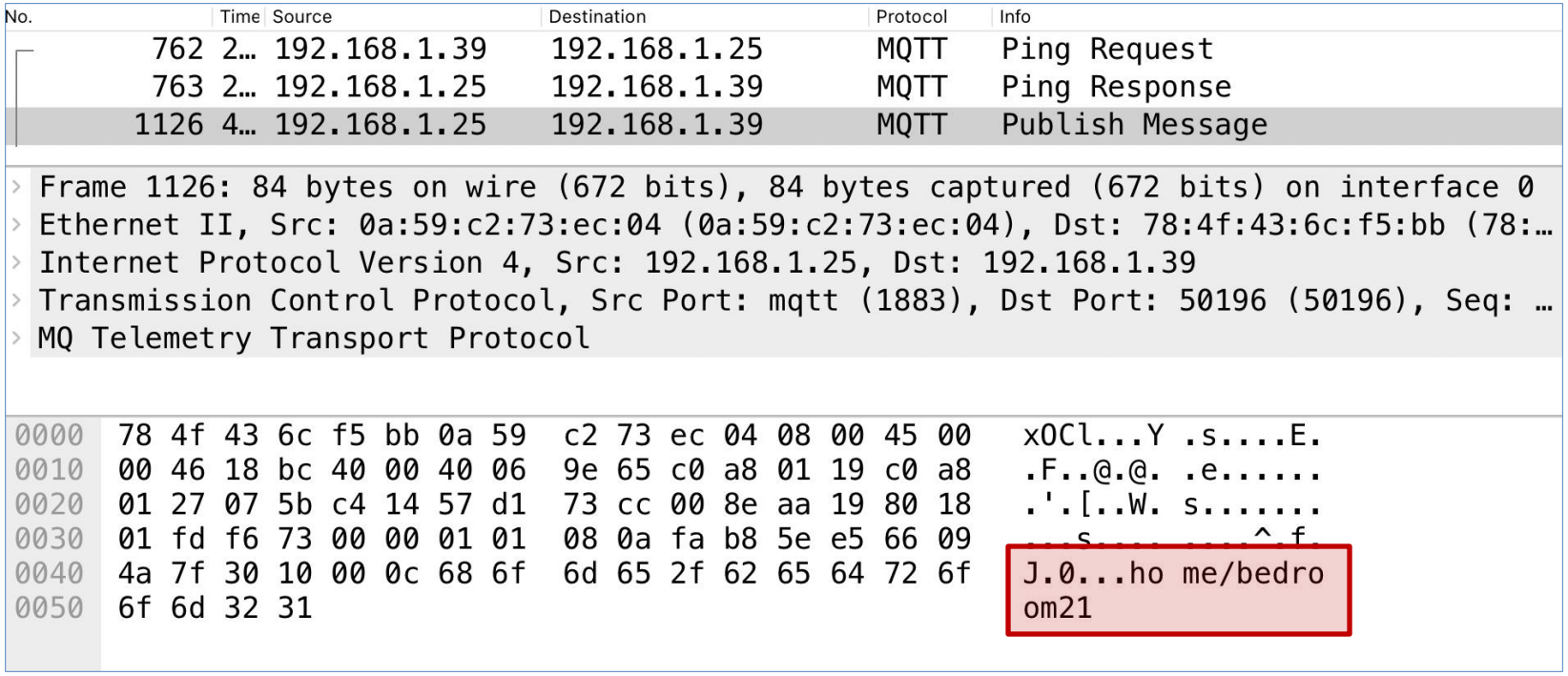


internet

MQTTS avec
Certificats TLS

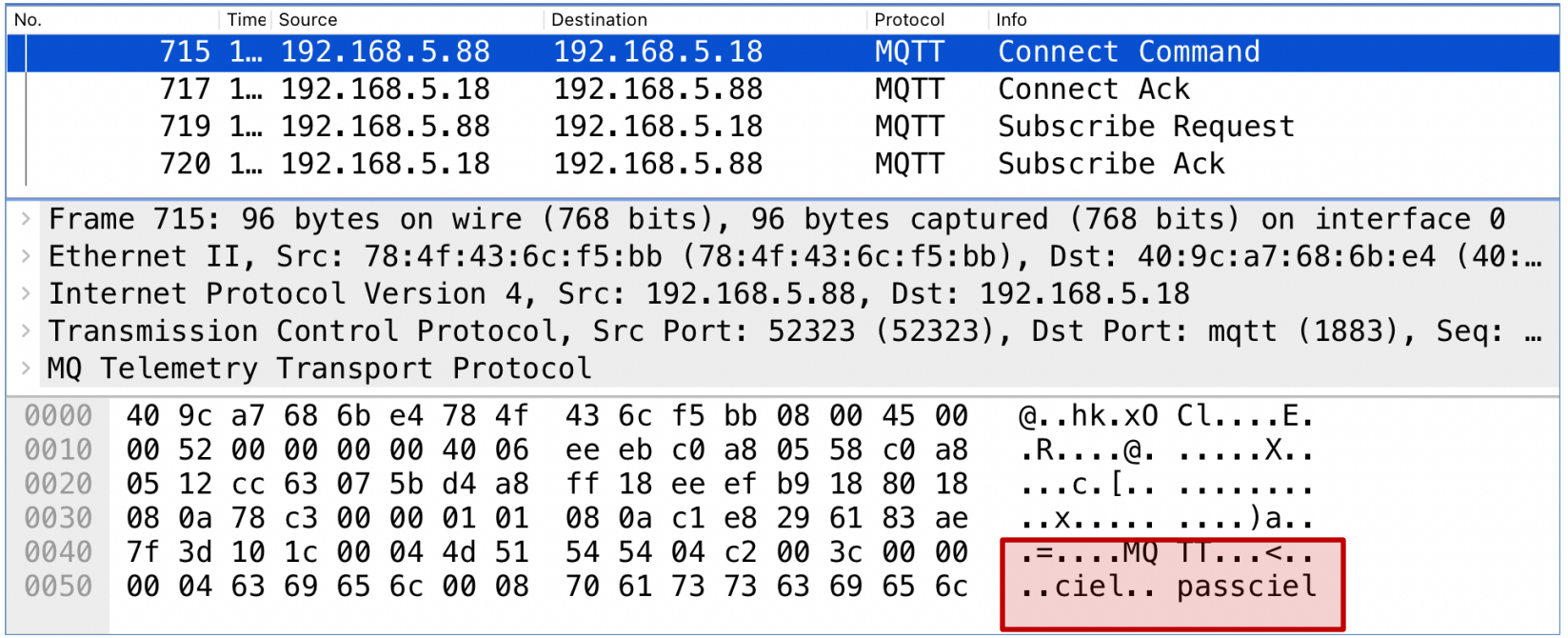
LES DONNÉES MQTT SONT EN CLAIRES

mosquitto_sub -h 192.168.1.25 -u ciel –P passciel -t ‘home/bedroom’



LES IDENTIFIANT/PASSWORD SONT EN CLAIRES

mosquitto_sub -h 192.168.1.18 -u ciel –P passciel -t ‘home/bedroom’

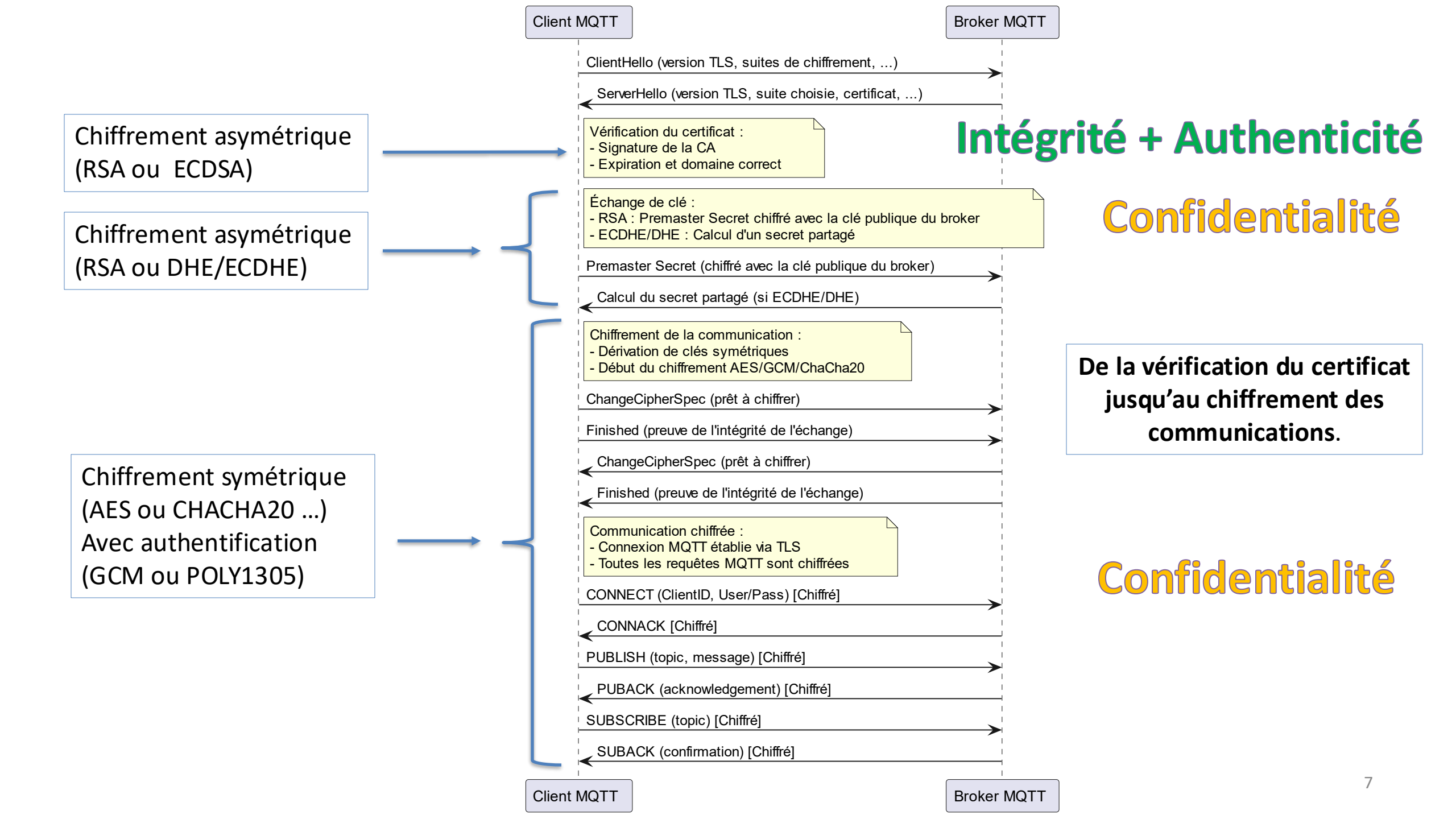


No.	Time	Source	Destination	Protocol	Info
715	1…	192.168.5.88	192.168.5.18	MQTT	Connect Command
717	1…	192.168.5.18	192.168.5.88	MQTT	Connect Ack
719	1…	192.168.5.88	192.168.5.18	MQTT	Subscribe Request
720	1…	192.168.5.18	192.168.5.88	MQTT	Subscribe Ack

```
› Frame 715: 96 bytes on wire (768 bits), 96 bytes captured (768 bits) on interface 0
› Ethernet II, Src: 78:4f:43:6c:f5:bb (78:4f:43:6c:f5:bb), Dst: 40:9c:a7:68:6b:e4 (40:…
› Internet Protocol Version 4, Src: 192.168.5.88, Dst: 192.168.5.18
› Transmission Control Protocol, Src Port: 52323 (52323), Dst Port: mqtt (1883), Seq: …
› MQ Telemetry Transport Protocol
```

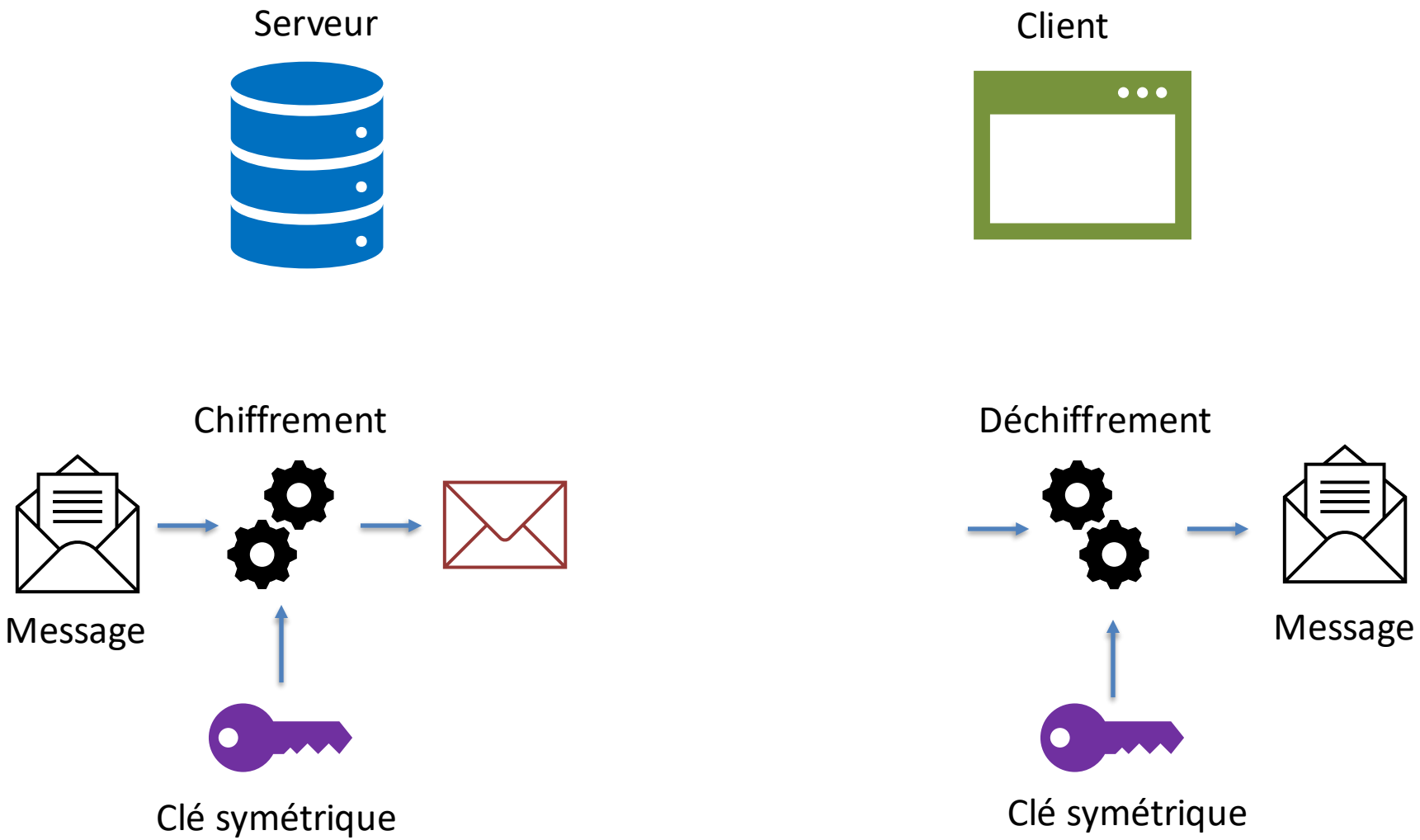
```
0000  40 9c a7 68 6b e4 78 4f  43 6c f5 bb 08 00 45 00   @..hk.xO Cl....E.
0010  00 52 00 00 00 00 40 06  ee eb c0 a8 05 58 c0 a8   .R....@. .....X..
0020  05 12 cc 63 07 5b d4 a8  ff 18 ee ef b9 18 80 18   ...c.[.. ........
0030  08 0a 78 c3 00 00 01 01  08 0a c1 e8 29 61 83 ae   ..x..... ....)a..
0040  7f 3d 10 1c 00 04 4d 51  54 54 04 c2 00 3c 00 00   .=....MQ TT...<..
0050  00 04 63 69 65 6c 00 08  70 61 73 73 63 69 65 6c   ..ciel.. passciel
```

Il va donc falloir chiffrer la communication. On utilise des certificats TLS


Client MQTT
Broker MQTT
ClientHello (version TLS, suites de chiffrement, ...)
ServerHello (version TLS, suite choisie, certificat, ...)
Chiffrement asymétrique (RSA ou ECDSA)
Vérification du certificat :
- Signature de la CA
- Expiration et domaine correct
Intégrité + Authenticité
Chiffrement asymétrique (RSA ou DHE/ECDHE)
Échange de clé :
- RSA : Premaster Secret chiffré avec la clé publique du broker
- ECDHE/DHE : Calcul d'un secret partagé
Confidentialité
Premaster Secret (chiffré avec la clé publique du broker)
Calcul du secret partagé (si ECDHE/DHE)
Chiffrement de la communication :
- Dérivation de clés symétriques
- Début du chiffrement AES/GCM/ChaCha20
ChangeCipherSpec (prêt à chiffrer)
Finished (preuve de l'intégrité de l'échange)
ChangeCipherSpec (prêt à chiffrer)
Finished (preuve de l'intégrité de l'échange)
De la vérification du certificat jusqu'au chiffrement des communications.
Chiffrement symétrique (AES ou CHACHA20 …) Avec authentification (GCM ou POLY1305)
Communication chiffrée :
- Connexion MQTT établie via TLS
- Toutes les requêtes MQTT sont chiffrées
Confidentialité
CONNECT (ClientID, User/Pass) [Chiffré]
CONNACK [Chiffré]
PUBLISH (topic, message) [Chiffré]
PUBACK (acknowledgement) [Chiffré]
SUBSCRIBE (topic) [Chiffré]
SUBACK (confirmation) [Chiffré]
Client MQTT
Broker MQTT

Confidentialité

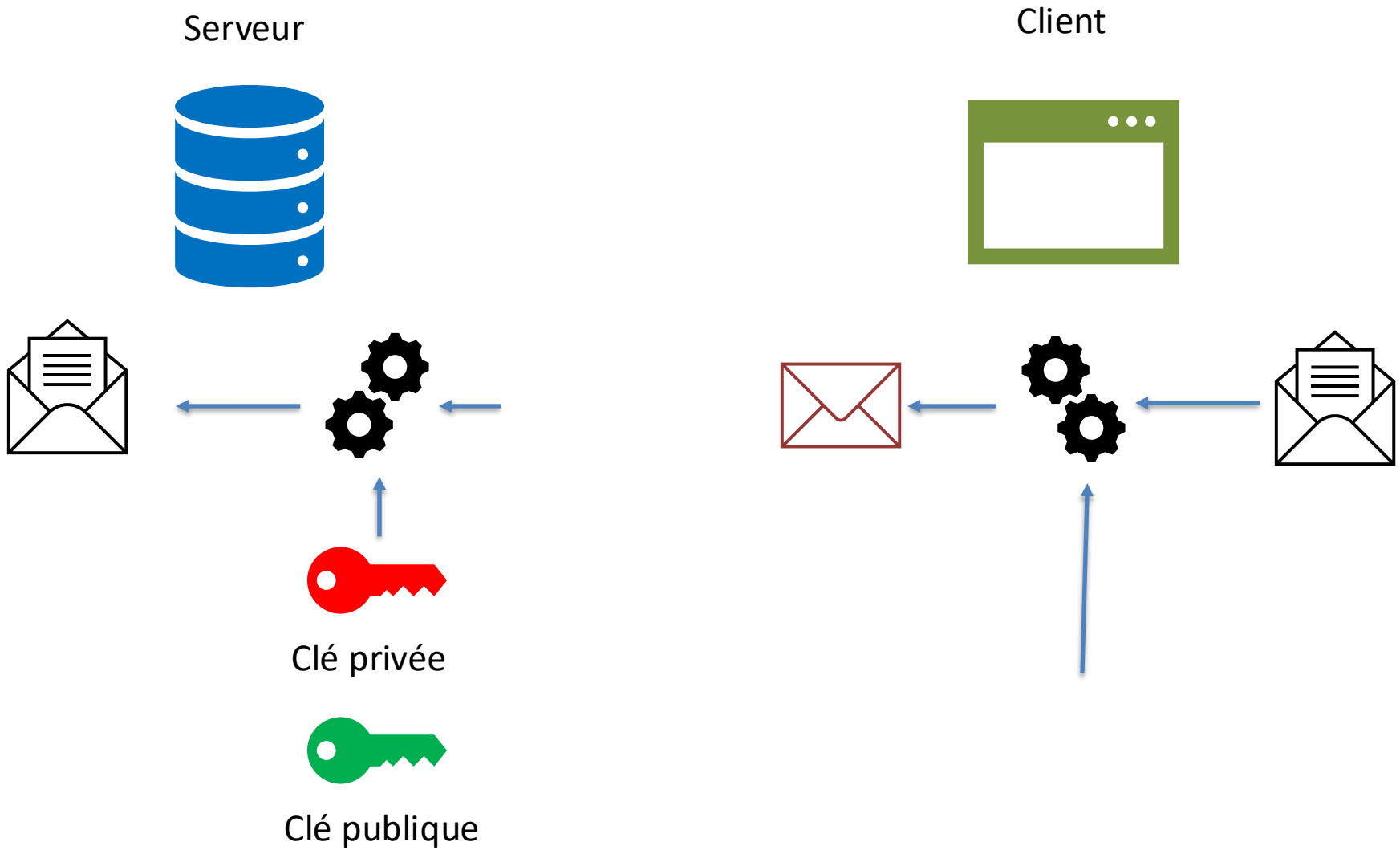
CHIFFREMENT SYMÉTRIQUE



Exemple d'algorithme de chiffrement symétrique : AES

Confidentialité

CHIFFREMENT ASYMÉTRIQUE

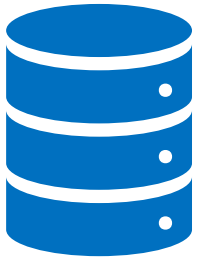


Exemple d'algorithme de chiffrement asymétrique : RSA

HACHAGE

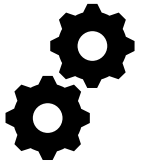
Intégrité

Serveur

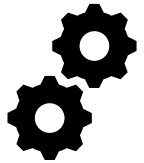


Objectif : vérifier l'intégrité

Client



Algorithme de hachage
Ex : SHA256

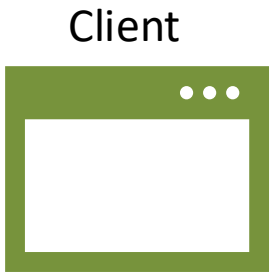


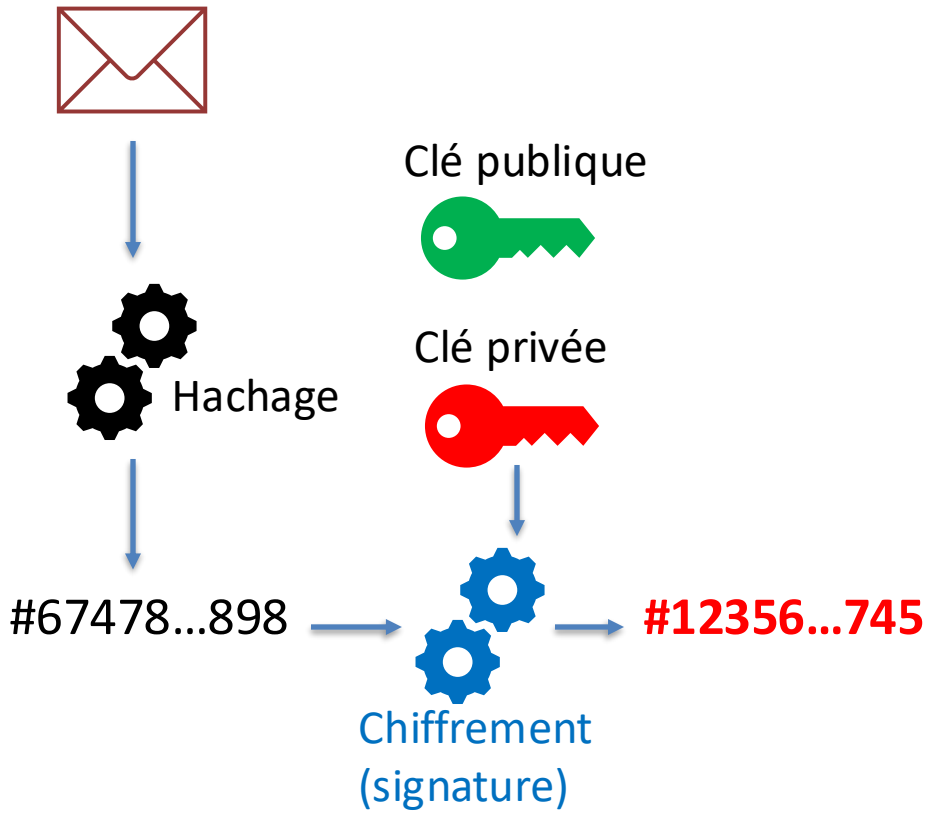
#67478…898

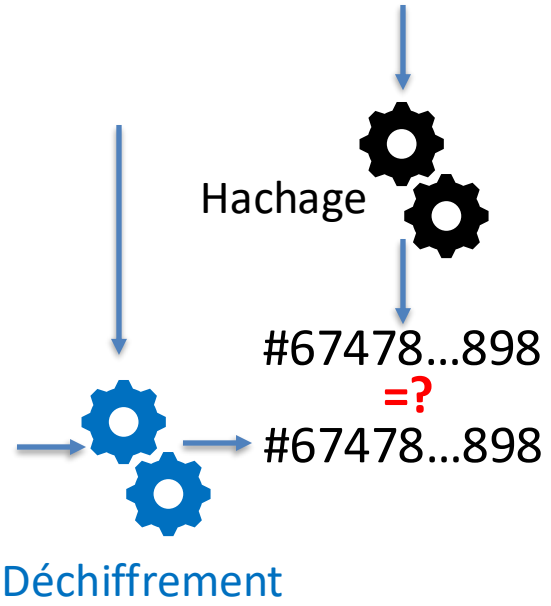
#67478…898
=?

SIGNATURE PAR CHIFFREMENT AVEC CLÉ PRIVÉE

Intégrité + Authenticité







Intégrité
+
Authenticité

CONSTITUTION D'UN CERTIFICAT TLS

Clé publique

- **Nom de Domaine.**
- **Détails de l'Organisation.**
- **Autorité de Certification (CA).**
- **Période de Validité .**
- **Numéro de série.**
- **Signature de l'Autorité de Certification.**
- **Algorithmes de chiffrement.**

Intégrité
+
Authenticité

CA : Certificate Authority

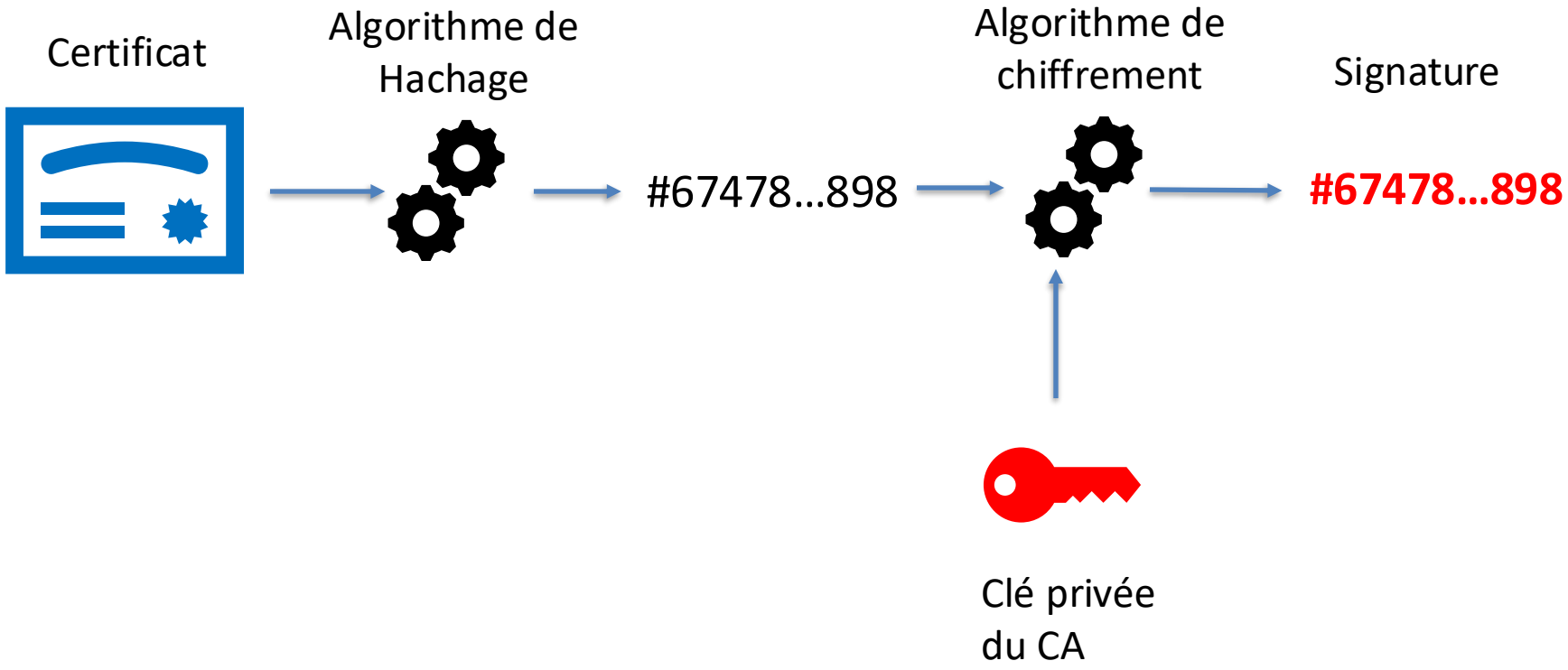
Un **CA (Certificate Authority)** est une entité de confiance qui délivre des certificats numériques permettant d'authentifier l'identité d'un serveur, d'un utilisateur ou d'un appareil. Il joue un rôle clé dans les infrastructures à clé publique (PKI) en garantissant la sécurité des communications via des certificats SSL/TLS.

Exemples :

- **Let's Encrypt.**
- **DigiCert**.
- **GlobalSign.**
- **Entrust.**

Signature d'un certificat

Intégrité
+
Authenticité



Intégrité
+
Authenticité

Vérification de la signature d'un certificat

Client

Magasin de certificats de CA

Clé publique let's encrypt

Clé publique Digicert

Clé publique Entrust

Sur Windows

Les certificats des CA sont stockés dans le **Magasin de certificats Windows** accessible via :

- Exécuter certmgr.msc

Emplacement physique :
C:\Windows\System32\certmgr.msc

Sur Linux

Les certificats racine des CA sont stockés dans :
/etc/ssl/certs/
/usr/local/share/ca-certificates/

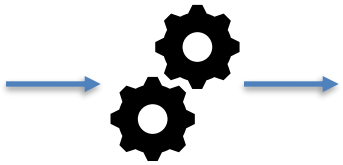
Intégrité + Authenticité

Vérification de la signature d'un certificat

Serveur

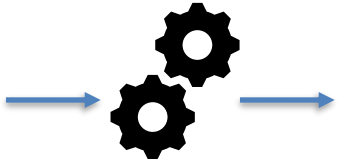
#5468…834

Client



#5975…263

=?



#5975…263

Clé publique du CA

Certificat signé par let's encrypt.
Pour rappel, il est constitué de la clé publique du site web et de données. Il est signé par le CA.
(c'est un hash chiffré avec la clé privée du CA)

GÉNÉRATION D'UN CERTIFICAT AUTO-SIGNÉ

1. Création un certificat CA :

Intégrité + Authenticité

openssl req -new -x509 -days 1826 -extensions v3_ca -keyout ca.key -out ca.crt

2. Création d'une clé privée pour le serveur :

openssl genrsa -out server.key 2048

Confidentialité

3. Création d'une demande de signature de certificat (CSR) :

openssl req -out server.csr -key server.key -new

4. Signature de la Demande de Signature de Certificat (CSR) et génération d'un certificat SSL/TLS signé :

openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days 360

Intégrité + Authenticité

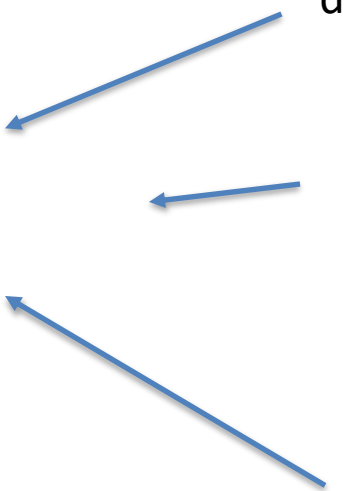
INTÉGRATION DES CERTIFICATS

Dans le fichier de configuration : /etc/mosquitto/mosquitto.conf

Côté Broker :

listener 8883
cafile /ca.crt
certfile /server.crt
keyfile /server.key

Côté Client :
cafile /ca.crt

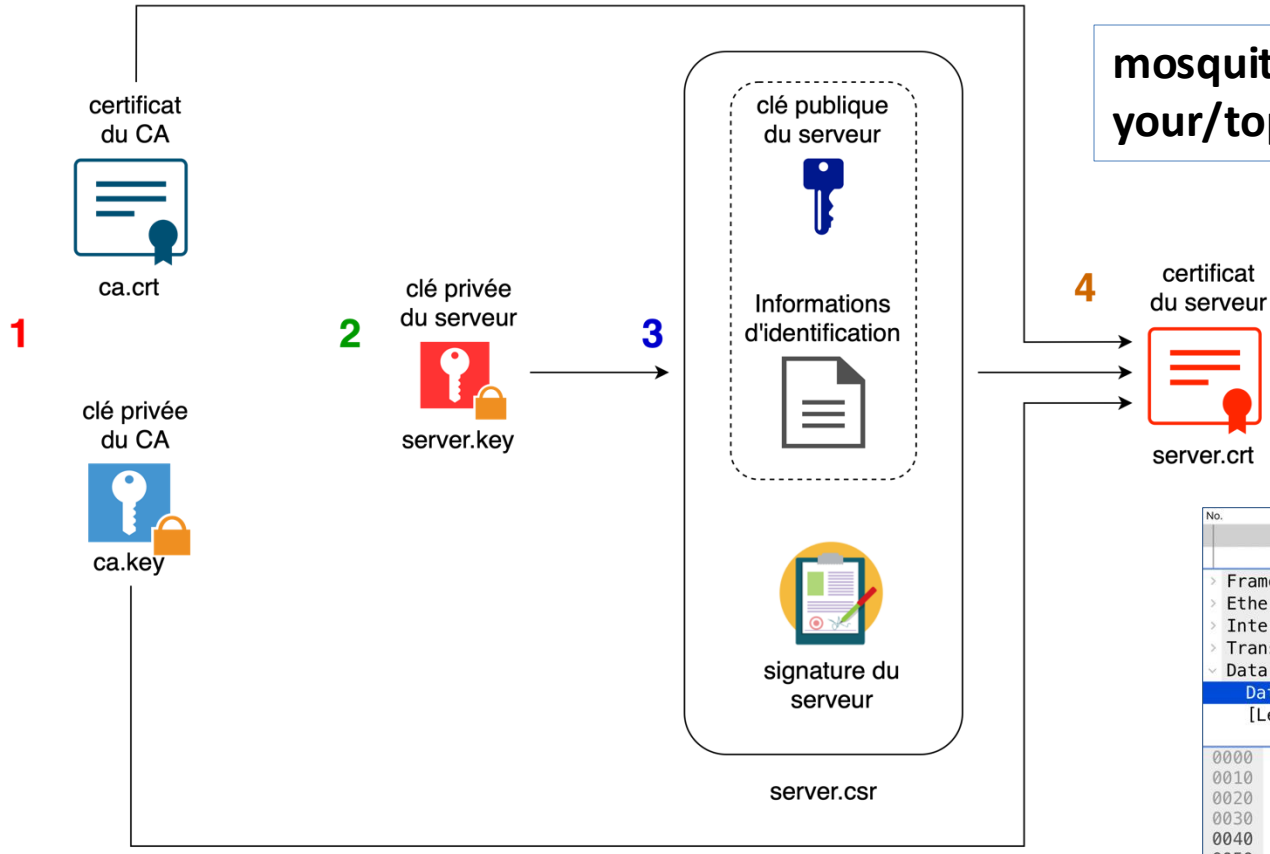


Certificat du CA (constitué de la clé publique du serveur et des données du certificat)

Certificat du serveur (constitué de la clé publique du serveur et des données du certificat)

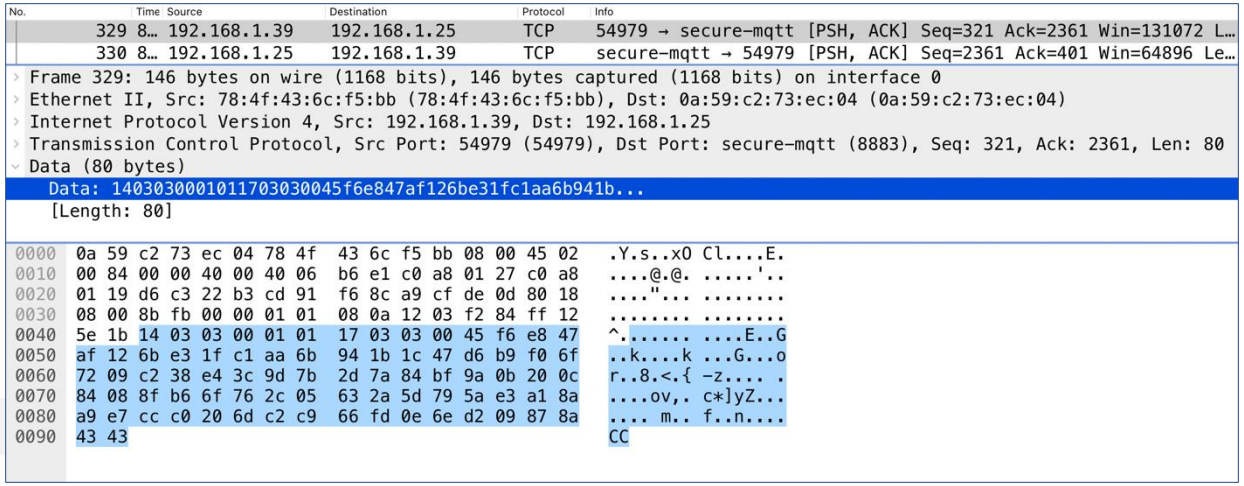
Clé privée du serveur

GÉNÉRER ET IMPLANTER DES CERTIFICATS SSL



mosquitto_pub -h 192.168.1.25 -p 8883 --cafile /ca.crt -t your/topic -m "message"

CHIFFREMENT DES DONNÉES



1 : `openssl req -new -x509 -days 1826 -extensions v3_ca -keyout ca.key -out ca.crt`

2 : `openssl genrsa -out server.key 2048`

3 : `openssl req -out server.csr -key server.key -new`

4 : `openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days 360`

CERTIFICAT SERVER.CRT

C’est codé en base 64 :


```
-----BEGIN CERTIFICATE-----
MIIDZjCCAk4CFAImW8oGSJGT34eEsQGHXgVXEoyQMA0GCSqGSIb3DQEBCwUAMFUx
CzAJBgNVBAYTAkZSMRMwEQYDVQQIDApTb21lLVN0YXRlMQ4wDAYDVQQHDAVQQVJJ
UzEhMB8GA1UECgwYSW50ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkMB4XDTI1MDIyMDE2
MDQyOFoXDTI2MDIxNTE2MDQyOFowgYkxCzAJBgNVBAYTAkZSMRMwEQYDVQQIDApT
b21lLVN0YXRlMQ4wDAYDVQQHDAVQQVJJUzEPMA0GA1UECgwGTmV3dG9uMQ0wCwYD
VQQLDARDSUVMMRYwFAYDVQQDDA0xOTIuMTY4LjEuMTk0MR0wGwYJKoZIhvcNAQkB
Fg5jaWVsQG5ld3Rvbi5mcjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AN82X2I8aMr67WMpXcDlfzurcta/v+N4FHBegJzdJhMZN2zz5teOsW3GHlrdD44h
3SGR+pgjk7VSKGw739I85PKpss3od0NMDP8qmyttl8HlO7g/JvU1ebTBC1JyqJQP
z+l/tVpSXUFRRuIiuk/b/SoKXUs7qDsKpVRZw2YzzpCiVA64Ak7Y/8zMfY75z8r5
4slTd5VLiUnK9+UtjdTvKOcxX0qyfkNZcOdxChJXNtq5abMZV8P7f3wtTnCUxGLK
uGFCKJdeh8/vK3GDgwar/N7+u9lSohCex+ygjdeHd2aqfK/182B78le7DYha63dW
KOqkdie5yh7tZY/fbj8Hcx8CAwEAATANBgkqhkiG9w0BAQsFAAOCAQEAP+YmPanr
pq3QClK5xItEe4h90hzwnnwmR7SdTOTkyXcfo0UgLvg/nPkgPe5/AgcuKZBNnEJ7
MSRcp/fpVq5EiZslWCQhLPPVvFO1Z+kzyYBjtDRUjL/lpGY5iPiAClLch9hEsfXT
e+KTEGqb73PzOZYl3OgbQawSdVCrNfL2l46Uala7iAWN4tZpDRgsLBGnVXTbvSTA
3hJqJY4GD670ztghBs6OsKA3TIKDFhJ5U7t5h6pzSxCgDWo06OHQsPcN81OMYJ5J
7zQMPjAZyZ/snOk3AMu6X2o8jSOZ40snlmUCVjkYBz5KgvfGooBO0JJerrOj4/r7
sg+K8RAhSzdARA==
-----END CERTIFICATE-----
```

openssl x509 -in server.crt -text –noout

```
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            02:26:5b:ca:06:48:91:93:df:87:84:b1:01:87:5e:05:57:12:8c:90
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = FR, ST = Some-State, L = PARIS, O = Internet Widgits Pty Ltd
        Validity
            Not Before: Feb 20 16:04:28 2025 GMT
            Not After : Feb 15 16:04:28 2026 GMT
        Subject: C = FR, ST = Some-State, L = PARIS, O = Newton, OU = CIEL, CN = 192.168.1.194, emailAddress = ciel@newton.fr
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:df:36:5f:62:3c:68:ca:fa:ed:63:29:5d:c0:e5:
                    7f:3b:ab:72:d6:bf:bf:e3:78:14:70:5e:80:9c:dd:
                    26:13:19:37:6c:f3:e6:d7:8e:b1:6d:c6:1e:5a:dd:
                    0f:8e:21:dd:21:91:fa:98:23:93:b5:52:28:6c:3b:
                    df:d2:3c:e4:f2:a9:b2:cd:e8:77:43:4c:0c:ff:2a:
                    9b:2b:6d:97:c1:e5:3b:b8:3f:26:f5:35:79:b4:c1:
                    0b:52:72:a8:94:0f:cf:e9:7f:b5:5a:52:5d:41:51:
                    46:e2:22:ba:4f:db:fd:2a:0a:5d:4b:3b:a8:3b:0a:
                    a5:54:59:c3:66:33:ce:90:a2:54:0e:b8:02:4e:d8:
                    ff:cc:cc:7d:8e:f9:cf:ca:f9:e2:c9:53:77:95:4b:
                    89:49:ca:f7:e5:2d:8d:d4:ef:28:e7:31:5f:4a:b2:
                    7e:43:59:70:e7:71:0a:12:57:36:da:b9:69:b3:19:
                    57:c3:fb:7f:7c:2d:4e:70:94:c4:62:ca:b8:61:42:
                    28:97:5e:87:cf:ef:2b:71:83:83:06:ab:fc:de:fe:
                    bb:d9:52:a2:10:9e:c7:ec:a0:8d:d7:87:77:66:aa:
                    7c:af:f5:f3:60:7b:f2:57:bb:0d:88:5a:eb:77:56:
                    28:ea:a4:76:27:b9:ca:1e:ed:65:8f:df:6e:3f:07:
                    73:1f
                Exponent: 65537 (0x10001)
```

Informations sur le CA

Informations sur le serveur

Chiffré = message ^d modulo n

n est le Modulus, un nombre semi-premier

d est l'exposant public. C'est 65537 presque tout le temps. Facile à encoder, peu de bits à 1 : 1000000000000001

Chiffré = message ^65537 modulo modulus

Suite ...

openssl x509 -in server.crt -text –noout

La signature du certificat :

```
Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        3f:e6:26:3d:a9:eb:a6:ad:d0:0a:52:b9:c4:8b:44:7b:88:7d:
        d2:1c:f0:9e:7c:26:47:b4:9d:4c:e4:e4:c9:77:1f:a3:45:20:
        2e:f8:3f:9c:f9:20:3d:ee:7f:02:07:2e:29:90:4d:9c:42:7b:
        31:24:5c:a7:f7:e9:56:ae:44:89:9b:25:58:24:21:2c:f3:d5:
        bc:53:b5:67:e9:33:c9:80:63:b4:34:54:8c:bf:e5:a4:66:39:
        88:f8:80:0a:52:dc:87:d8:44:b1:f5:d3:7b:e2:93:10:6a:9b:
        ef:73:f3:39:96:25:dc:e8:1b:41:ac:12:75:50:ab:35:f2:f6:
        97:8e:94:6a:56:bb:88:05:8d:e2:d6:69:0d:18:2c:2c:11:a7:
        55:74:db:bd:24:c0:de:12:6a:25:8e:06:0f:ae:f4:ce:d8:21:
        06:ce:8e:b0:a0:37:4c:82:83:16:12:79:53:bb:79:87:aa:73:
        4b:10:a0:0d:6a:34:e8:e1:d0:b0:f7:0d:f3:53:8c:60:9e:49:
        ef:34:0c:3e:30:19:c9:9f:ec:9c:e9:37:00:cb:ba:5f:6a:3c:
        8d:23:99:e3:4b:27:96:65:02:56:39:18:07:3e:4a:82:f7:c6:
        a2:80:4e:d0:92:5e:ae:b3:a3:e3:fa:fb:b2:0f:8a:f1:10:21:
        4b:37:40:44
```

CLÉ PRIVÉE SERVER.KEY

-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDfNl9iPGjK+u1j
KV3A5X87q3LWv7/jeBRwXoCc3SYTGTds8+bXjrFtxh5a3Q+OId0hkfqYI5O1Uihs
O9/SPOTyqbLN6HdDTAz/KpsrbZfB5Tu4Pyb1NXm0wQtScqiUD8/pf7VaUl1BUUbi
IrpP2/0qCl1LO6g7CqVUWcNmM86QolQOuAJO2P/MzH2O+c/K+eLJU3eVS4lJyvfl
LY3U7yjnMV9Ksn5DWXDncQoSVzbauWmzGVfD+398LU5wlMRiyrhhQiiXXofP7ytx
g4MGq/ze/rvZUqlQnsfsol3Xh3dmqnyv9fNge/JXuw2IWut3VijqpHYnucoe7WWP
324/B3MfAgMBAAECggEAEf8sSZtdR6Bbq3cWXAsk6wazx6TGL+zqmPTFoeAixb9u
vJYdspuKqrMil2jjbC5GyglfFk0oB6KMJ83cCfNm8jnEdN2dYAutsJUa1bGMXtHA
suZEmdip62z1ILyl4uozdTwavyGM60tGT81FAFNYYh9bzkkNE1WUZuyl2lOsz/Rg
DNUrmyy/juDUqphSjw3oS5HdmoVNanz41RCAfjylFNEReAELex48o/T8ifjFqU7W
PyYmjk1lDO3mqI6vH8+PyicHkv0f+/3YeJ1blcjdZKzvMk4Im9iKBG0IJzLEDUyJ
aKWHgDML61W7kvQuJY0S/0HJ5rwgyfFbBphvYty2oQKBgQDzjxbhJjiCbvRkzxm+
oIFvgLWsjDSNnNUzNoPsCUVCOrkGNqlzq++p9TnhWPiujWMWZpFPnMaxN+G5dxRV
HY0PU1d2nXdP125T5tbqun/qbICLfvLOMYkCOxG/cRAfq0bKpYbUcHGGwZeO/m1f
/AjC/UUzX2FP93Yrr2qEACLpAwKBgQDqnTh11E9aHUrYDPTw1S3fJ5EjxhcuEmRD
Uu8gynIfMLnu0/ZEuKmL7Uq1RjFlgy/53UZyhfkmItHmcTNSFKmfSBycLHL+Kpte
fvVB6Fjgwv90aTycKXIe/DbL3quCXPvioBfoNFN5bJZFuWtDTpLxcRztfQ5HFh1C
Z/GfFpQ8tQKBgA7jQjzE/1NJwPqghixW11KfhXtknpnBam7U+D9nWapwuHqewMDn
U6EJ8l3J+HmC/vmRj3RUWvjeN5gEpWOGiObU61W+zlo9CutqAt0aRVNpCnp8ag8c
jAls3ura2Gd3Kr6cW5+EuAvKrZ76AmmJmGeC38YCIIqoz7pf4Jzs/TVNAoGBAJnn
6VHipOuIq8BvP399WUo6uoutNyeSCRPhI91u8M5lOJPjmHocaZzRXWDtWxospRxU
fBJEsNzDms6BFBmUVHGY7KvGvOjfKm0i++Dwet13Df1fy9LlnWeWLJwwxnIObW6N
dyFsshXFlg7HRAYzVVHt0fw5PVSEvU0Cww6J0VTBAoGAOXwIMMQGTVGRgwFNUQy7
sqZM1yzbzv2ONet59UAy6+0Flwx3NJa/O9aR71kO8XuEgowiUl+XxBVqZkfwLuVl
a0Y0gBpqextt8W9bZnLOgOJ4g/6KXvRyyik6h3ztNt6VIyjsBSpMIBjg01zxtPSv
seISpicadgsTcGrmBdwySmU=
-----END PRIVATE KEY-----

openssl rsa -in server.key -text -noout

```
modulus:
    00:df:36:5f:62:3c:68:ca:fa:ed:63:29:5d:c0:e5:
    7f:3b:ab:72:d6:bf:bf:e3:78:14:70:5e:80:9c:dd:
    26:13:19:37:6c:f3:e6:d7:8e:b1:6d:c6:1e:5a:dd:
    0f:8e:21:dd:21:91:fa:98:23:93:b5:52:28:6c:3b:
    df:d2:3c:e4:f2:a9:b2:cd:e8:77:43:4c:0c:ff:2a:
    9b:2b:6d:97:c1:e5:3b:b8:3f:26:f5:35:79:b4:c1:
    0b:52:72:a8:94:0f:cf:e9:7f:b5:5a:52:5d:41:51:
    46:e2:22:ba:4f:db:fd:2a:0a:5d:4b:3b:a8:3b:0a:
    a5:54:59:c3:66:33:ce:90:a2:54:0e:b8:02:4e:d8:
    ff:cc:cc:7d:8e:f9:cf:ca:f9:e2:c9:53:77:95:4b:
    89:49:ca:f7:e5:2d:8d:d4:ef:28:e7:31:5f:4a:b2:
    7e:43:59:70:e7:71:0a:12:57:36:da:b9:69:b3:19:
    57:c3:fb:7f:7c:2d:4e:70:94:c4:62:ca:b8:61:42:
    28:97:5e:87:cf:ef:2b:71:83:83:06:ab:fc:de:fe:
    bb:d9:52:a2:10:9e:c7:ec:a0:8d:d7:87:77:66:aa:
    7c:af:f5:f3:60:7b:f2:57:bb:0d:88:5a:eb:77:56:
    28:ea:a4:76:27:b9:ca:1e:ed:65:8f:df:6e:3f:07:
    73:1f
```

```
privateExponent:
    11:ff:2c:49:9b:5d:47:a0:5b:ab:77:16:5c:0b:24:
    eb:06:b3:c7:a4:c6:2f:ec:ea:98:f4:c5:a1:e0:22:
    c5:bf:6e:bc:96:1d:b2:9b:8a:aa:b3:22:97:68:e3:
    6c:2e:46:ca:02:1f:16:4d:28:07:a2:8c:27:cd:dc:
    09:f3:66:f2:39:c4:74:dd:9d:60:0b:ad:b0:95:1a:
    d5:b1:8c:5e:d1:c0:b2:e6:44:99:d8:a9:eb:6c:f5:
    20:bc:a5:e2:ea:33:75:3c:1a:bf:21:8c:eb:4b:46:
    4f:cd:45:00:53:58:62:1f:5b:ce:49:0d:13:55:94:
    66:ec:a5:d8:83:ac:cf:f4:60:0c:d5:2b:9b:2c:bf:
    8e:e0:d4:aa:98:52:8f:0d:e8:4b:91:dd:9a:85:4d:
    6a:7c:f8:d5:10:80:7e:3c:a5:14:d1:11:78:01:0b:
    7b:1e:3c:a3:f4:fc:89:f8:c5:a9:4e:d6:3f:26:26:
    8e:4d:65:0c:ed:e6:a8:8e:af:1f:cf:8f:ca:27:07:
    92:fd:1f:fb:fd:d8:78:9d:5b:95:c8:dd:64:ac:ef:
    32:4e:08:9b:d8:8a:04:6d:08:27:32:c4:0d:4c:89:
    68:a5:87:80:33:0b:eb:55:bb:92:f4:2e:25:8d:12:
    ff:41:c9:e6:bc:20:c9:f1:5b:06:98:6f:62:dc:b6:
    a1
```

n= modulus que l'on a vu dans le certificat.

exposant privé : d

```
publicExponent: 65537
```

exposant public : e

Chiffrement = m ^ e modulo n

m = chiffrement ^ d modulo n

Suite ...

openssl rsa -in server.key -text -noout

```
exponent1:
    0e:e3:42:3c:c4:ff:53:49:c0:fa:a0:86:2c:56:d7:
    52:9f:85:7b:64:9e:99:c1:6a:6e:d4:f8:3f:67:59:
    aa:70:b8:7a:9e:c0:c0:e7:53:a1:09:f2:5d:c9:f8:
    79:82:fe:f9:91:8f:74:54:5a:f8:de:37:98:04:a5:
    63:86:88:e6:d4:eb:55:be:ce:5a:3d:0a:eb:6a:02:
    dd:1a:45:53:69:0a:7a:7c:6a:0f:1c:8c:09:6c:de:
    ea:da:d8:67:77:2a:be:9c:5b:9f:84:b8:0b:ca:ad:
    9e:fa:02:69:89:98:67:82:df:c6:02:20:8a:a8:cf:
    ba:5f:e0:9c:ec:fd:35:4d
```

```
coefficient:
    39:7c:08:30:c4:06:4d:51:91:83:01:4d:51:0c:bb:
    b2:a6:4c:d7:2c:db:ce:fd:8e:35:eb:79:f5:40:32:
    eb:ed:05:97:0c:77:34:96:bf:3b:d6:91:ef:59:0e:
    f1:7b:84:82:8c:22:52:5f:97:c4:15:6a:66:47:f0:
    2e:e5:48:6b:46:34:80:1a:6a:7b:1b:6d:f1:6f:5b:
    66:72:ce:80:e2:78:83:fe:8a:5e:f4:72:ca:29:3a:
    87:7c:ed:36:de:95:23:28:ec:05:2a:4c:20:18:e0:
    d3:5c:f1:b4:f4:af:b1:e2:12:a6:27:1a:76:0b:13:
    70:6a:e6:05:dc:32:4a:65
```

```
exponent2:
    00:99:e7:e9:51:e2:a4:eb:88:ab:c0:6f:3f:7f:7d:
    59:4a:3a:ba:8b:ad:37:27:92:09:13:e1:23:dd:6e:
    f0:ce:65:38:93:e3:98:7a:1c:69:9c:d1:5d:60:ed:
    5b:1a:2c:a5:1c:54:7c:12:44:b0:dc:c3:9a:ce:81:
    14:19:94:54:71:98:ec:ab:c6:bc:e8:df:2a:6d:22:
    fb:e0:f0:7a:dd:77:0d:fd:5f:cb:d2:e5:9d:67:96:
    2c:9c:30:c6:72:0e:6d:6e:8d:77:21:6c:b2:15:c5:
    96:0e:c7:44:06:33:55:51:ed:d1:fc:39:3d:54:84:
    bd:4d:02:c3:0e:89:d1:54:c1
```

Exponent1 et **Exponent2** sont des versions réduites de l'exposant privé utilisées pour accélérer les calculs RSA en travaillant séparément avec les deux facteurs premiers de la clé privée. **Le coefficient** est un multiplicateur qui permet de recombiner efficacement les résultats après ces calculs optimisés. Ces trois valeurs permettent d'utiliser le **théorème des restes chinois (CRT)** pour accélérer le déchiffrement et la signature RSA.