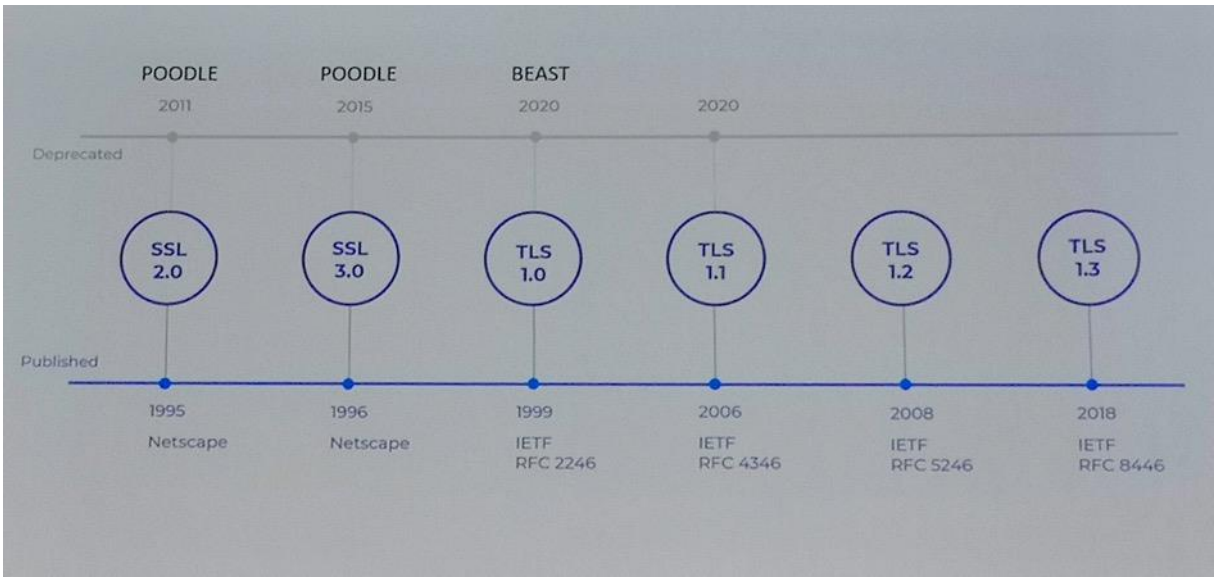
Mieux maitriser TLS, OpenSSL et les certificats

Date : 21 avril 2022
Speaker : Mathieu Humbert (Accenture)
Format : conférence (45mn)

Dans la vie de tous les jours, les dévs ne sont pas confrontés de face au protocile TLS : **Transport Layer Security**. TLS est encore aujourd'hui bien souvent associé au vieil acronyme **SSL** (ancien nom de TLS).
En 1995, le web décolle. Le navigateur phare est Netscape.
SSL 1.0 n'est jamais sorti.
SSL 2.0 est sorti mais renforcé par SSL 3.0 un an plus tard.
Lorsque Netscape a coulé, il a donné SSL à la fondation IETF qui en change le nom après quelques modifications.
Aujourd'hui : 90% du trafic mondial est en HTTPS.
Les versions importantes sont TLS 1.2 et 1.3. Les versions précédentes comprennent des failles de sécurité.

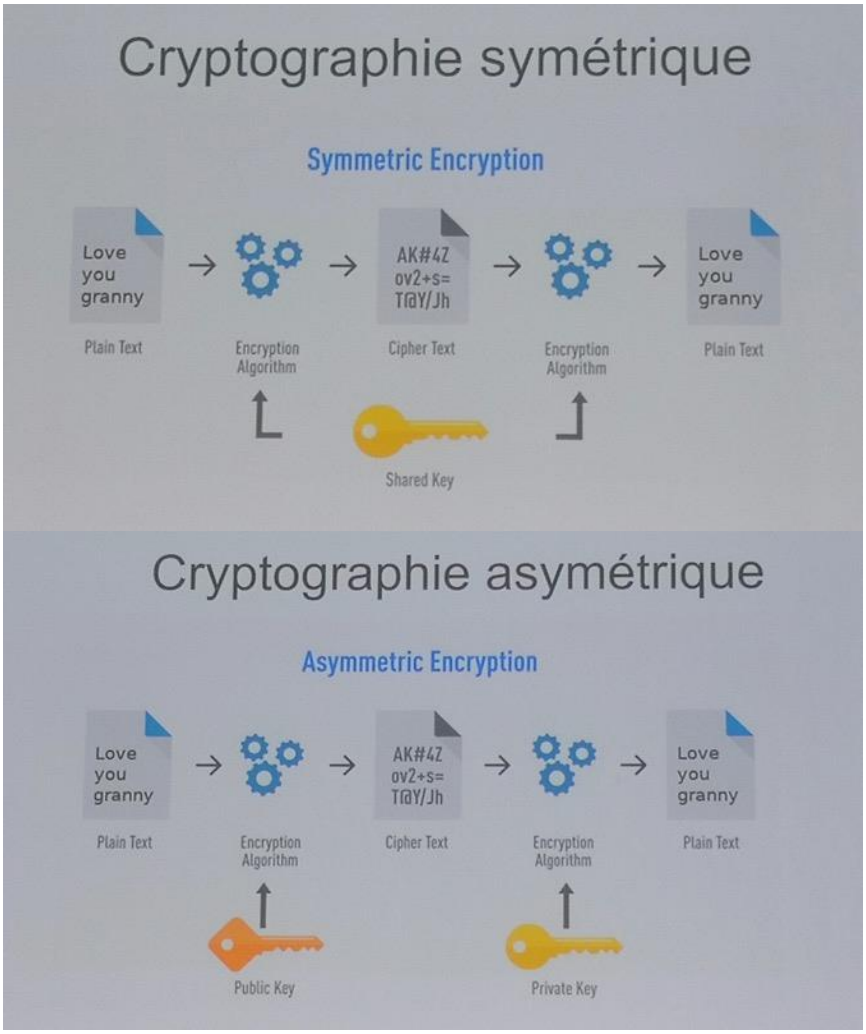


Transport Layer Security

3 piliers du TLS :

1. **Confidentialité**
2. **Authentification** : empêche les attaques par man in the middle
3. **Intégrité** : empêche le bit flipping

TLS s'appuie sur la **cryptographie symétrique** et **asymétrique**.



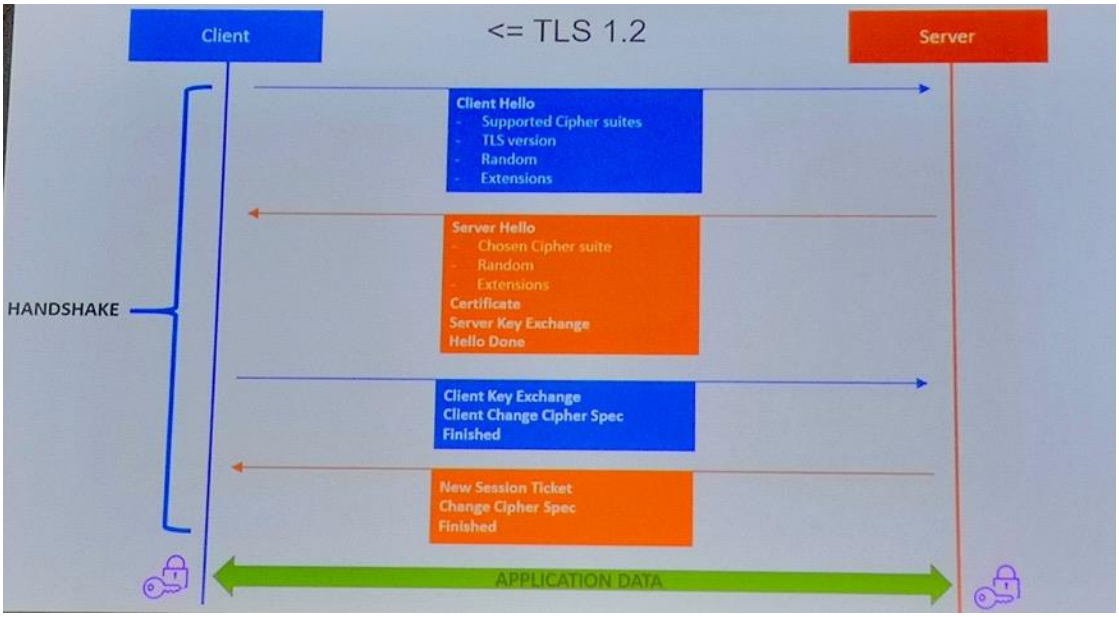
La clé privé permet de déchiffrer du contenu et de créer une signature.

Une requête HTTPS envoyée à un site pour échanger de la data utilise un chiffrement asymétrique à l'initialisation puis bascule sur un chiffrement symétrique.
Le chiffrement asymétrique est lent : on cherche à passer en symétrique dès qu'on peut.

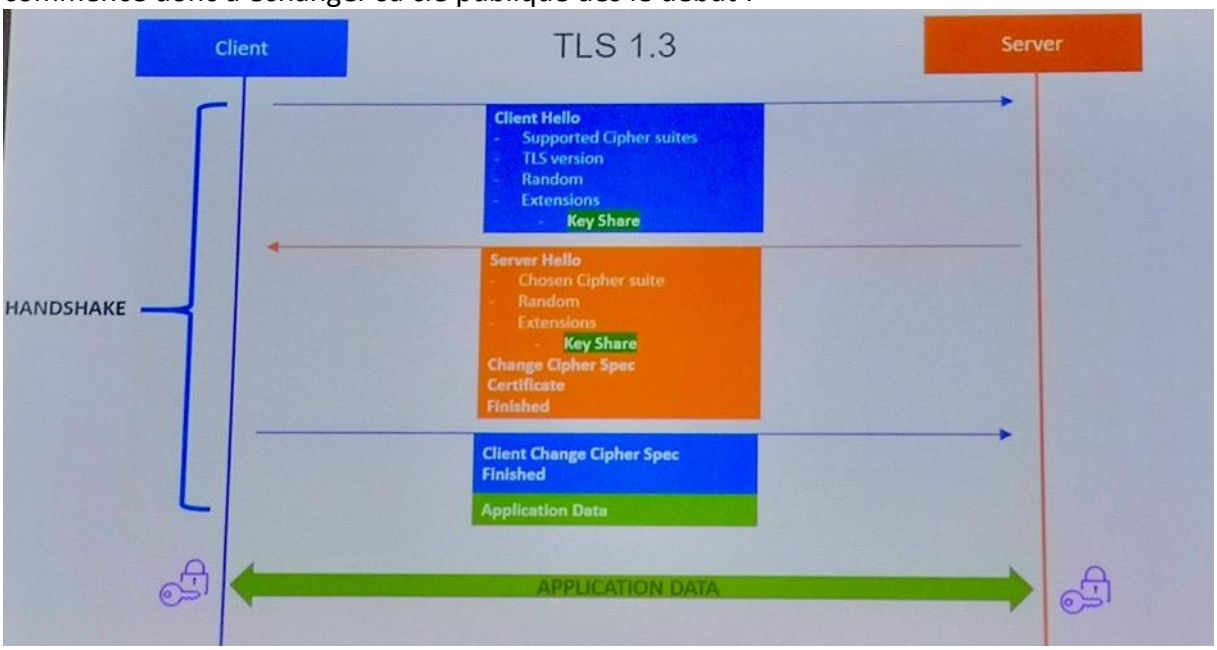
Questions :

- Quel est la clé partagée ?
- Comment se partager les clés ?
- Comment on chiffre le contenu ?
- Comment vérifier le certificat serveur ?

Phase de **healthcheck de TLS 1.2** :



Le client utilise le Supported Cipher suites pour indiquer au serveur les algos qu'il supporte. Dans TLS 1.3, on n'a plus qu'un seul aller/retour dans le meilleur des cas. Le client fait le pari que le serveur le supportera aussi. Si tel n'était pas le cas, on retourne sur 2 A/R. Le client commence donc à échanger sa clé publique dès le début :



Cipher suites

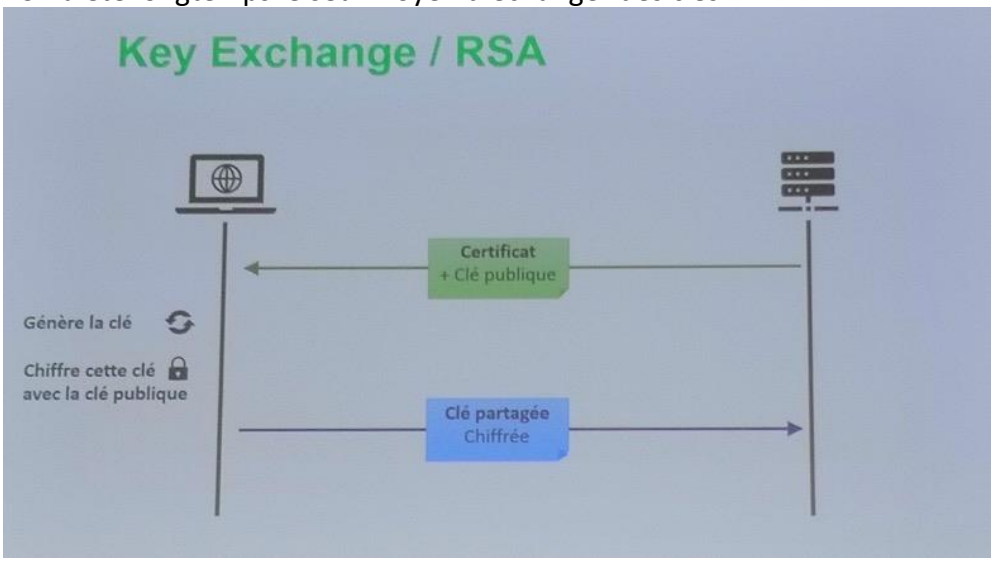
En TLS 1.2, le **cipher suites** conditionne le niveau de sécurité.
Présentation du tableau des ciphers les plus utilisés en 2021.
Dans la chaine on retrouve plusieurs chaînes :

1. Key Echange
    a. Plusieurs modes : RAS, DHE ou ECDHE
2. Authentication

   a. Ex : RSA ou ECDSA
3. Cipher
   a. Algorithm
   b. Strength
   c. Mode of operation : important pour TLS

   Exemple : AES128-GCM
4. MAC

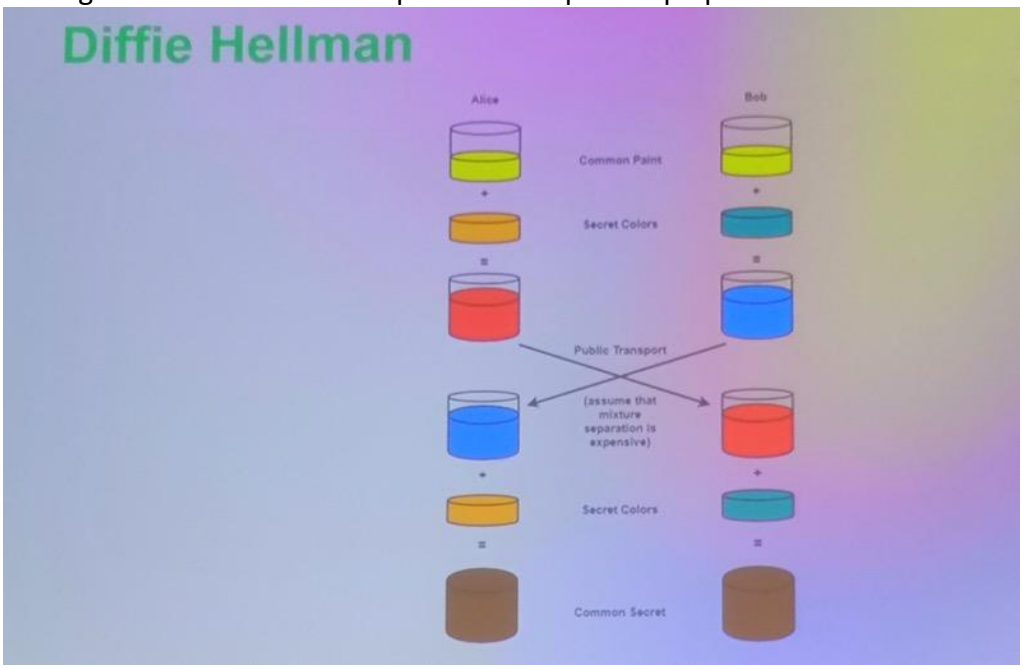
RSA a été longtemps le seul moyen d'échanger des clés



<u>Problème</u> : la clé privée peut se leaker et cela permettrait d'exploiter toutes les conversations en cours et celles enregistrées dans le passé. RSA n'a pas la propriété du **Forward secrecy**.

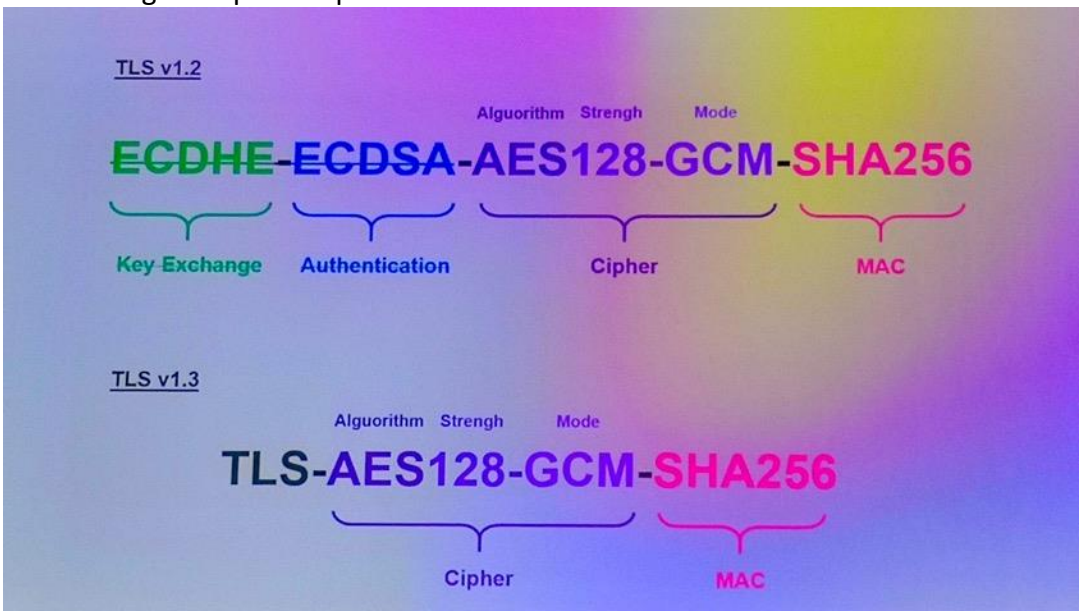
Diffie Hellman

On s'accorde sur une clé partagée sans qu'elle ne circule sur le réseau.
Analogie des couleurs utilisée par Mathieu pour expliquer son fonctionnement :

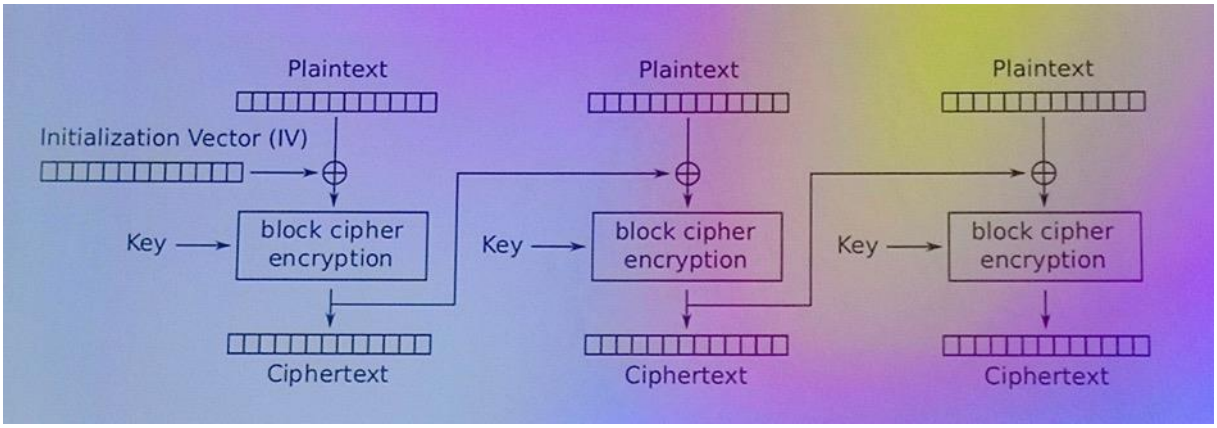


La couleur publique est envoyée sur le réseau. Une fois échangée, on la mélange avec la couleur privée des 2 côté et on arrive à s'accorder sur une couleur commune.
TLS 1.3 impose le Diffile Hellman pour le Forward secrecy

TLS 1.2 vient avec 260 Cypher Suite dont la plupart ne sont plus recommandés en 2022.
TLS 1.3 ne garde que le Cipher et le MAC



L'algo d'encryption prend des chaînes de 128 bits, quel que soit la taille de la donnée à chiffrer.
Découper la donnée n'est pas suffisant car elle conserve les patterns (image du TUX)



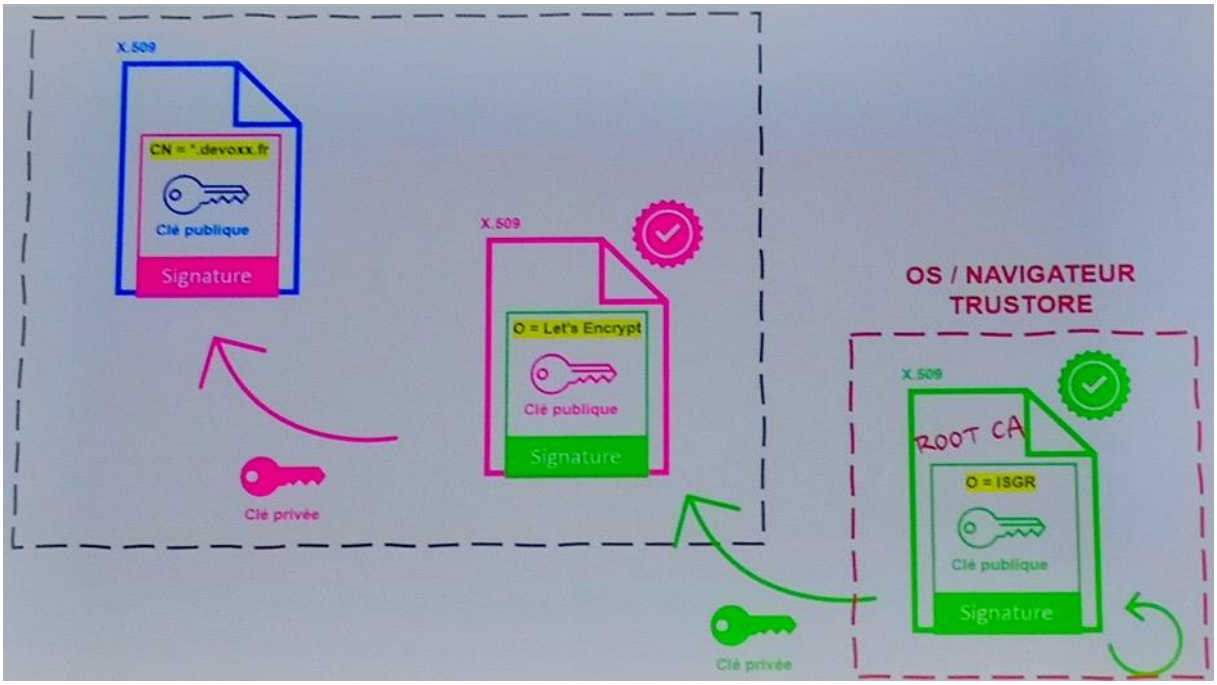
Le check d'intégrité comme SHA256.
MAC : hash du contenu authentifié.
Les questions qu'on peut se poser : faut-il calculer le MAC sur le contenu en clair ou le contenu chiffré ?
En 1995 on a choisi de calculer le MAC sur le contenu en clair.
Avec le recul, cette approche n'est pas la plus sécurisée.
TLS 1.3 change les choses : calcul du MAC sur le contenu chiffré.
Le mode **GCM** est un algorithme : il peut faire le chiffrement et le MAC (intégrité) en même temps. Il permet d'ajouter de l'intégrité sur d'autres contenus.
En TLS 1.3 il ne reste plus que 6 algos de cipher, supportant tous le GCM.
SHA256 reste néanmoins encore utilisé lors de l'initialisation de la connexion.

Certificats

Le certificat et un moyen numérique d'identifié une personne ou une identité.
Le certificat est un fichier au format X509.
Il contient :

- les informations permettant d'identifier la personne. Le Common Name (CN) = *.devoxx.fr.
- la clé publique
- la signature

Le certificat est signé avec la clé privée.
Le certificat de Let's encrypt n'a pas de CN. Il a le super-pouvoir O = Let' Encrypt qui lui permet de générer d'autres certificats.
Le certificat de Let's encrypt est signé par un **Root CA**.
La clé de la Root CA pour le O=ISGR. **Ce certificat se trouve dans les trusts stores des OS, des navigateurs, de la JVM.**



Ce système repose sur une chaine de confiance. On s'attend à ce que les Root CA soient des sociétés très sérieuses.

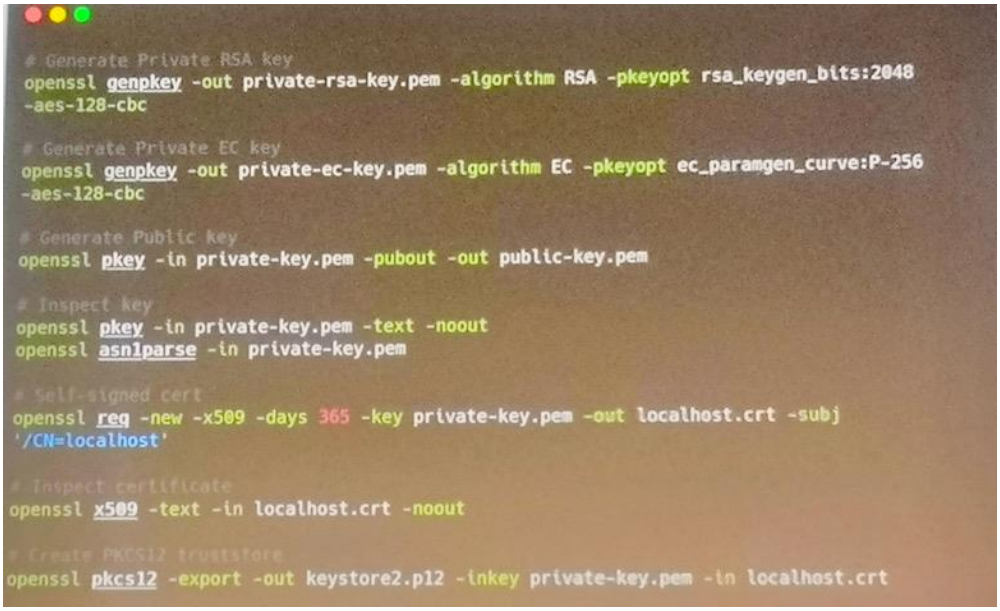
Or, par le passé, on a eu des frayeurs :

- En 2011, **DigiNotar** se fait hacker. La clé privée est volée. Le hacker a généré des certificats pour Windows Updates, …. Il se met en Man in the middle. Faille massivement utilisé en Iran. Depuis cette histoire, des guidelines ont été éditées pour renforcer le niveau de sécurité
- En 2015, **Superfish** : un malware intercepte les communications, les déchiffre avec la clé privée volée, injecte de la pub au navigateur. La clé privée a été intégrée en dur dans le malware et permettait de générer des certificats valides.

La clé privée du Root CA est générée dans des conditions très strictes.
Chaine de confiance systématiquement basée sur 3 ou 4 certificats.

<u>Outils</u> :

- Keytool dans le monde Java
- OpenSSL : Mathieu recommande de prendre le temps de monter en compétences sur cet outil très riche



```
# Generate Private RSA key
openssl genpkey -out private-rsa-key.pem -algorithm RSA -pkeyopt rsa_keygen_bits:2048
-aes-128-cbc

# Generate Private EC key
openssl genpkey -out private-ec-key.pem -algorithm EC -pkeyopt ec_paramgen_curve:P-256
-aes-128-cbc

# Generate Public key
openssl pkey -in private-key.pem -pubout -out public-key.pem

# Inspect key
openssl pkey -in private-key.pem -text -noout
openssl asn1parse -in private-key.pem

# Self-signed cert
openssl req -new -x509 -days 365 -key private-key.pem -out localhost.crt -subj
'/CN=localhost'

# Inspect certificate
openssl x509 -text -in localhost.crt -noout

# Create PKCS12 truststore
openssl pkcs12 -export -out keystore2.p12 -inkey private-key.pem -in localhost.crt
```

Session questions/réponses :

- La chaine de confiance peut être utilisée dans un cadre privé en entreprise. Des middle-box demandent la clé privée lors de leur installation pour analyser le trafic. Il ne faut pas les oublier lorsqu'on veut changer de clé privée. L'IETF recommande d'utiliser un full proxy et d'utiliser un PKI privé.