Infrastructures à clés publiques

Sécurité Informatique (Licence 3)

Abdelmalik Bachir

Systèmes symétriques vs. systèmes asymétriques

Atouts des systèmes symétriques

- rapidité
- coût et complexité réduits

Limitations des systèmes symétriques

- ne permettent pas la signature
- gestion lourde des clés

Atouts des systèmes asymétriques

- permettent la signature numérique
- gestion efficace des clés

Limitations des systèmes asymétriques

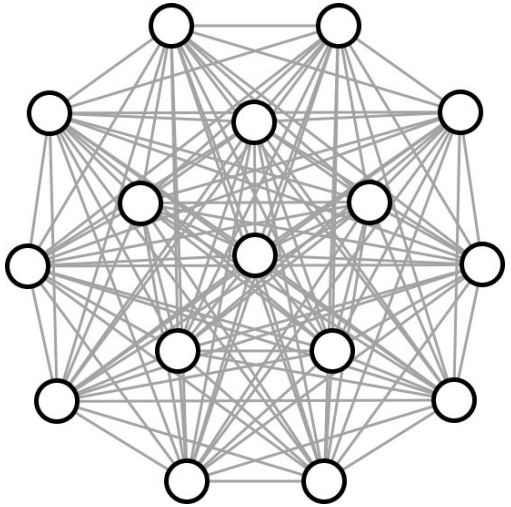
- lenteur
- complexité élevée

Systèmes symétriques vs. systèmes asymétriques

Gestion des clés

Dans un réseau composé de n entités (noeuds), on a besoin de

- n*(n-1)/2 clés avec un système symétrique
  - une clé pour chaque paire de noeuds
- 2n clés avec un système asymétrique
  - une clé publique et une clé privée pour chaque noeud.



Les clés partagées dans un système symétrique

Exemple:

- Pour n = 500 on a besoin de **124750** clés avec un système symétrique
- Pour n = 500 on a besoin de **1000** clés avec un système asymétrique

Combiner les systèmes symétriques et asymétriques

Les deux systèmes ont des atouts et des limitations.

Afin de combiner leurs atouts, on conçoit un système hybride dans lequel :

- la crypto asymétrique est utilisée pour
  - gestion des clés
  - signature numérique
- la crypto symétrique est utilisée pour
  - le chiffrement des message
- les fonctions de hachage pour
  - le contrôle de l'intégrité des messages

Gestion des clés avec crypto asymétrique

Exemple 1 : envoi d'un message confidentiel de a vers b

Version 1 :

- M1 : a -> b : {m}PKb

Cette version est valide mais le chiffrement avec la clé publique est très lent particulièrement quand la taille du message m est grande.

Version 2 :

- M1 : a -> b : a.{Kab}PKb
- M2 : b -> a : {m}Kab

Pour pallier les limitations du chiffrement par une clé asymétrique, on échange d'abord une clé symétrique Kab puis on chiffre avec.

Attention : la version 1 n'est pas considérée solution correcte.

Signature des messages

Exemple 1 : a signe un message et l'envoie à b

Version 1 :

- M1 : a -> b : {m}SKa

Cette version est valide mais la signature de tout le message avec la clé privée de a est très lente particulièrement quand la taille du message m est grande.

Version 2 :

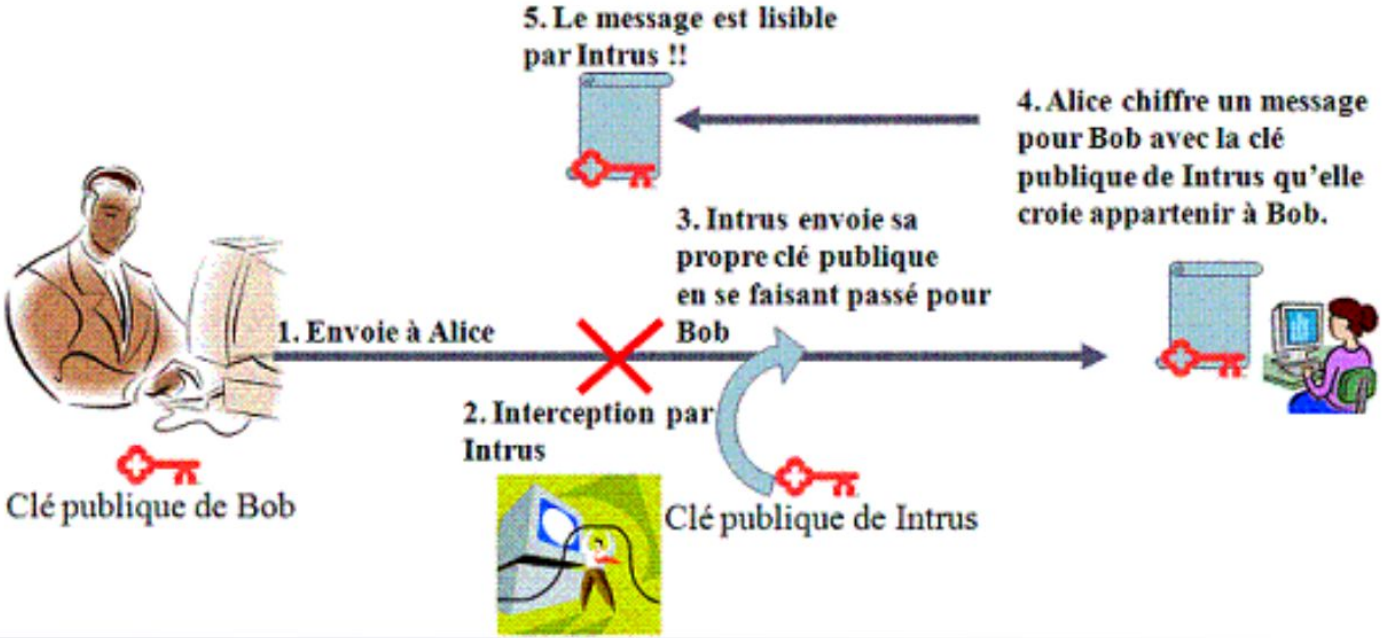
- M1 : a -> b : m.{H(m)}SKa

Pour pallier les limitations de la signature de tout le message (opération lente quand le message es grand), on ne signe que le condensé du message (H(m)).

Attention : la version 1 n'est pas considérée solution correcte.

Limitation d’un système à clés asymétriques

Difficulté principale : comment garantir que la clé publique d’une entité donnée est vraiment celle prétendue ? Attention attaque de type Man In The Middle.



Man in the middle

Limitation d’un système à clés asymétriques

Exemple : si j’ai une clé publique donnée (exemple : PKb), comment je peux etre sure que c’est vraiment la clé publique de b ?
Solution :

1. Connaître b personnellement et c’est lui qui me garantit sa clé publique personnellement (exemple : par téléphone ou tout autre canal sécurisé)
2. Sinon, il faut un garant qui me garantit que la clé publique de b est PKb.

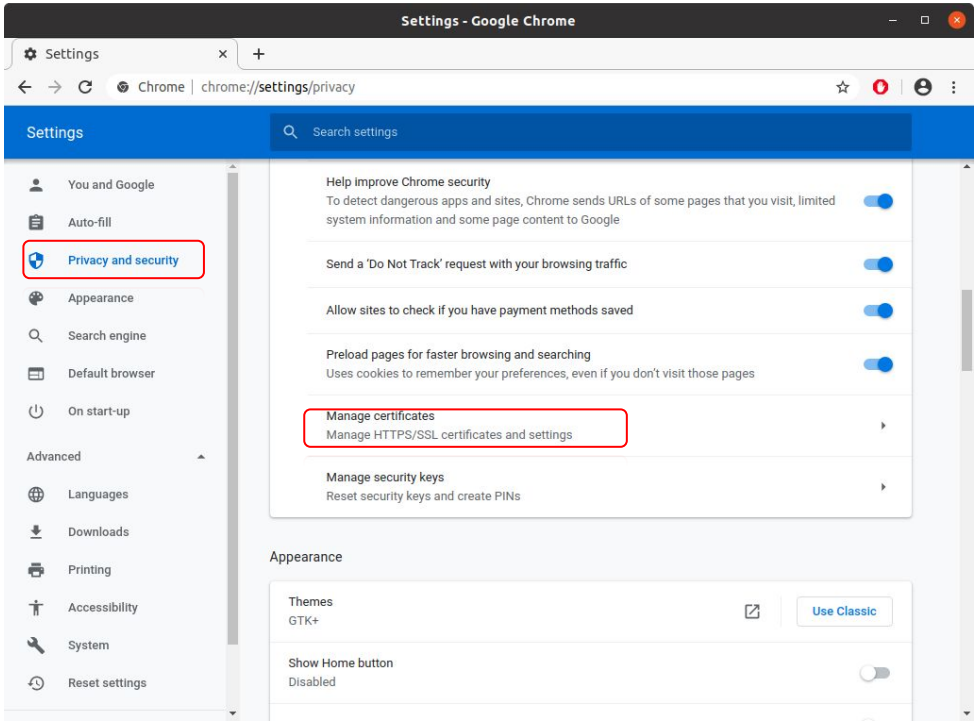
La solution (1) n’est pas pratique, car il y a plusieurs entités qu’on ne connaît pas personnellement. Donc, la solution (2) est plus pratique.

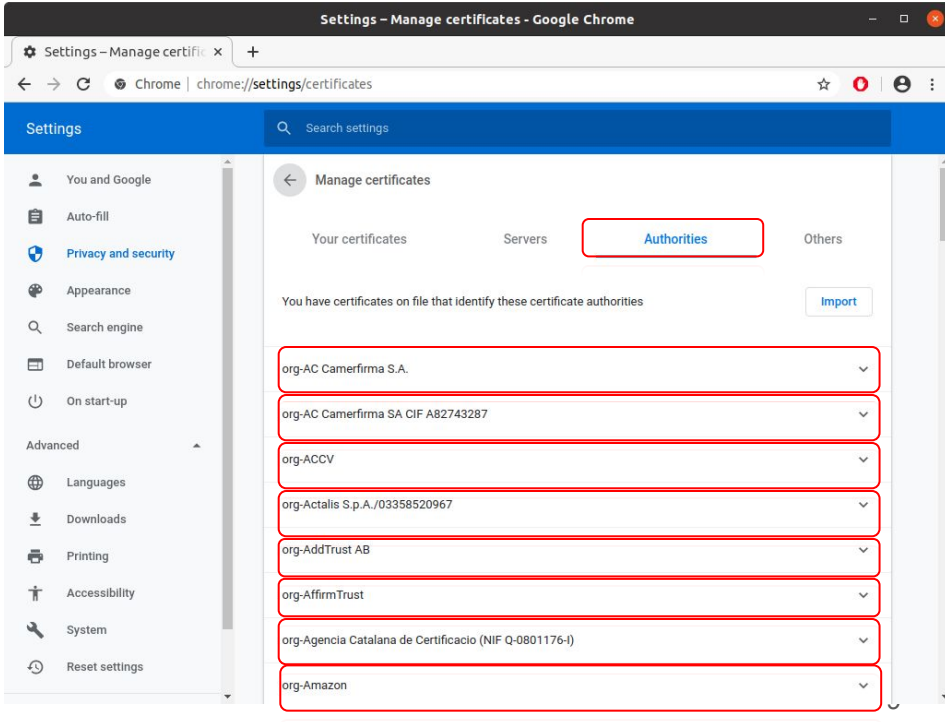
La solution (2) nécessite une autorité qui garantit (**certifie)** la correspondance entre les identités (a, b, …) et les clés publiques (PKa, PKb, …) correspondantes.

Cette autorité est appelée **autorité de certification**

Autorité de certification numérique

Les autorités de certification sont des entités dont on connaît les clés publiques. Les autorités de certification sont **pré-installées** dans nos systèmes et navigateurs.



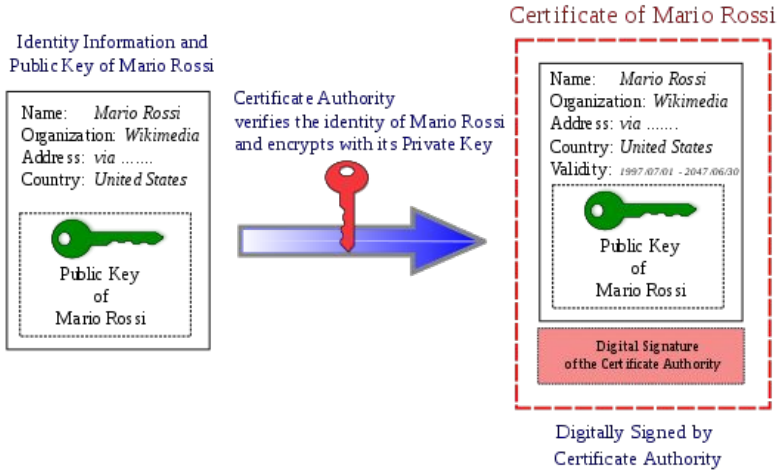


Autorité de certification numérique

Les autorités de certifications sont **pré-installées** dans nos systèmes et navigateurs donc on connait leurs clés publiques.

Comment font les autorités pour certifier (garantir) que la clé publique d'une entité donnée (exemple : b) est vraiment (PKb) ?

Solution : les autorités de certification signe un message qui contient les deux informations (b et PKb).



Autorité de certification numérique

En utilisant la notation, nous allons décrire un certificat de **b** délivré par l'autorité de certification **CA**.

- PK_{CA} est la clé publique de CA
- SK_{CA} est la clé privée de CA
- b est l'identité de b
- PKb est la clé publique de b

Pour établir un certificat, CA **signe** la structure suivante {b.PKb} donc :

- **b.PKb.{H(b.PKb)}SK_{AC}** est le **certificat** de **b délivré** par l'**autorité CA**

Autorité de certification et confiance

L'autorité de certification certifie la correspondance **(clé publique, identité)** pour l'ensemble d'une population. Ceci mène à faire **régner la confiance par transitivité** :

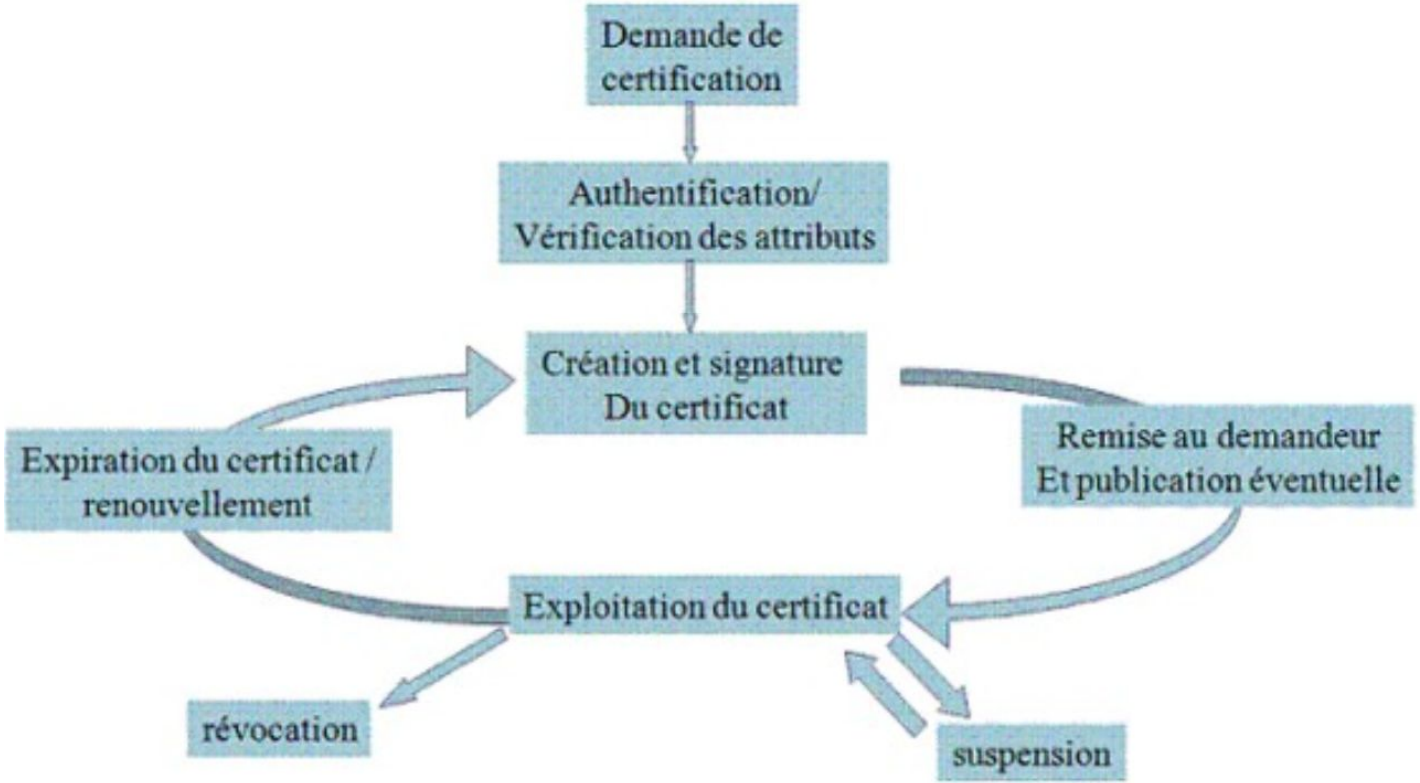
- **a** fait confiance à l'autorité de certification
- l'autorité de certification délivre un certificat à **b**
- **a** est assuré de l'identité de **b**

Structure d’un certificat numérique

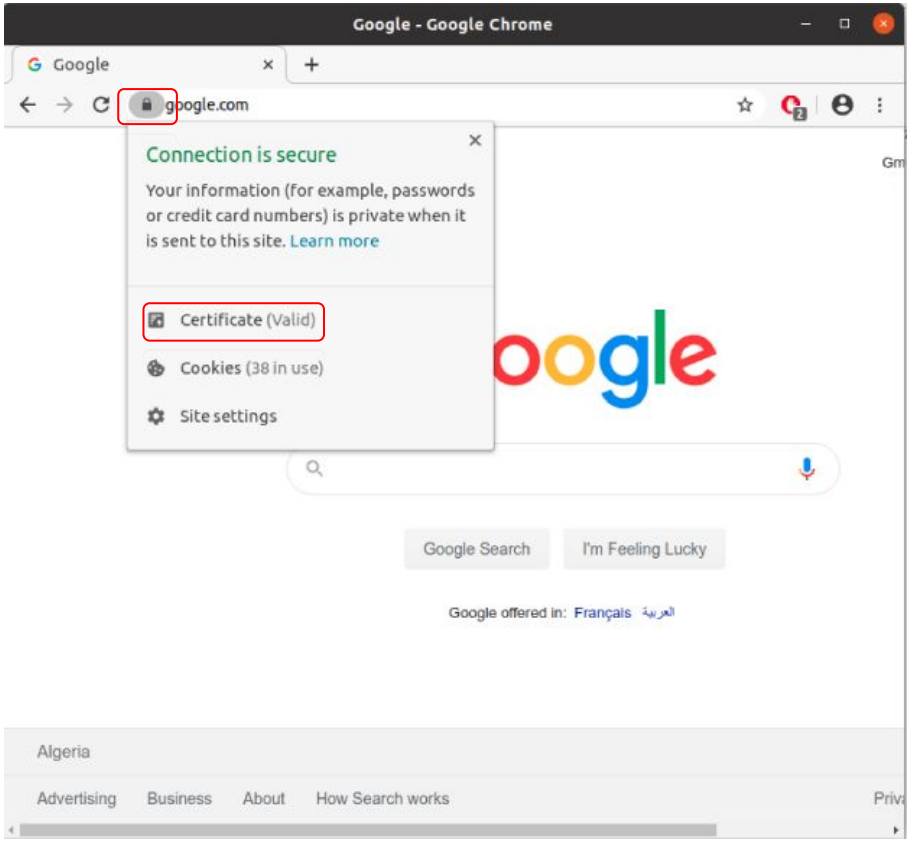
Un certificat numérique est une structure de données qui lie une identité à une clé publique. Le certificat est signé par une autorité de certification pour qu’il soit valide. En général, un certificat contient plusieurs autres informations :

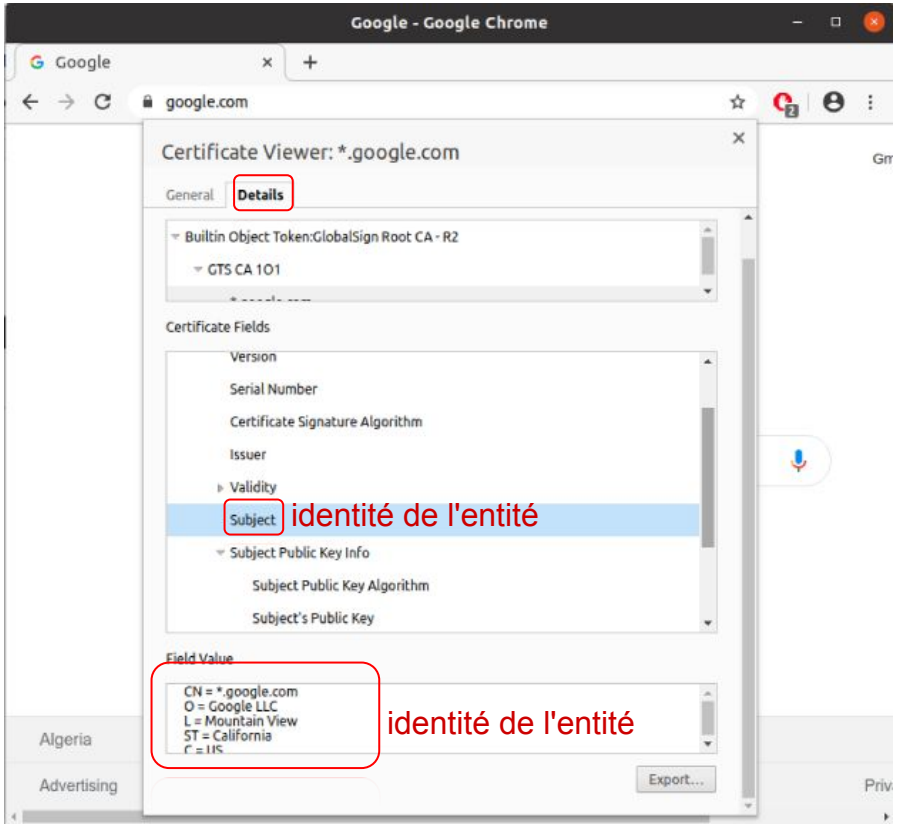
- identité
- clé publique correspondant à l'identité
- signature du certificat par l'autorité
- version du certificat
- numero de sequence du certificat
- validité du certificat
- information sur la clé publique de l'entité (algorithme utilisee, …)
- ...

Cycle de vie d'un certificat

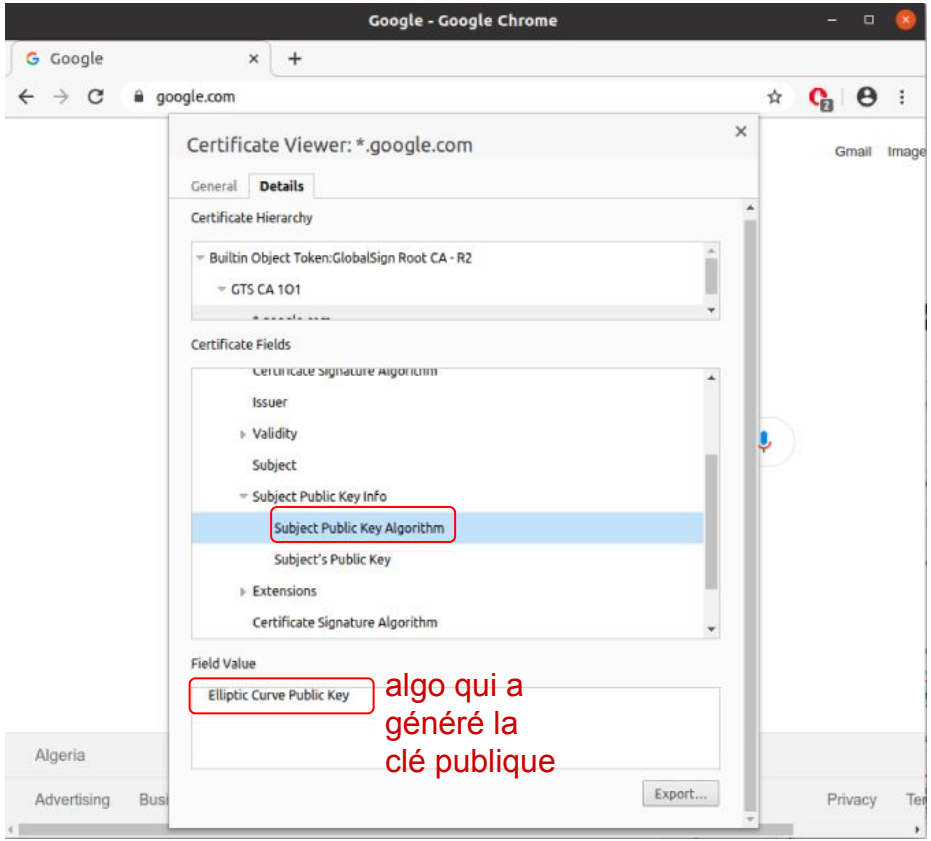


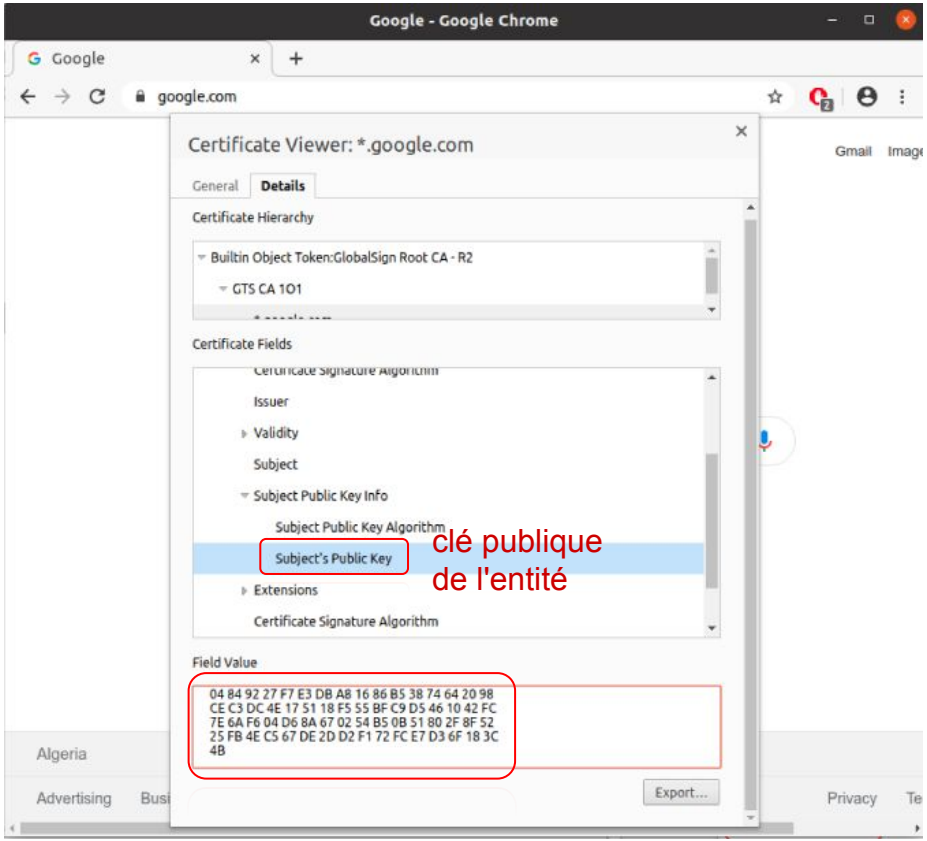
Structure d’un certificat numérique



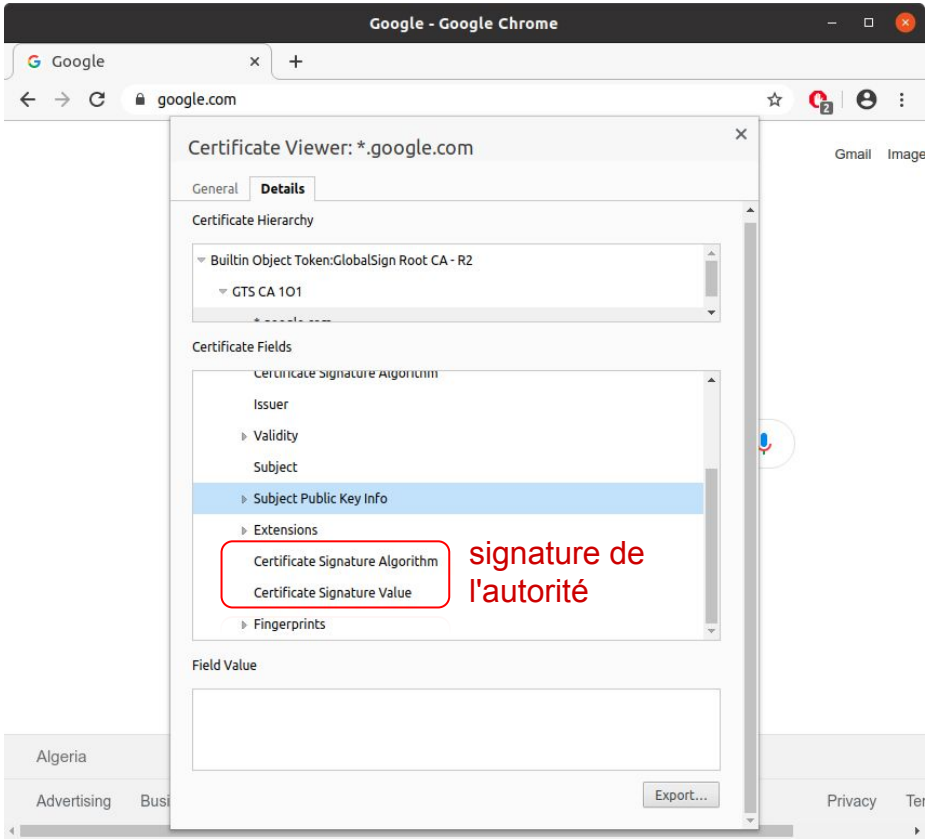


Structure d'un certificat numérique





Structure d’un certificat numérique



Certificats auto-signés des CA

Les clés publiques des autorités de certifications (Certification Authority) sont sauvegardés dans des certificats **auto-signés** par les autorités elles-mêmes. Ex :

- $CA1.PK_{CA1}.\{H(CA1.PK_{CA1})\}SK_{CA1}$: certificat auto-signé par CA1
- $CA2.PK_{CA2}.\{H(CA1.PK_{CA2})\}SK_{CA2}$: certificat auto-signé par CA2
- …

Les certificats des CA (Certification Authority) sont sauvegardés dans les navigateurs et dans les systèmes des entités.

Quand on achète un nouveau ordinateur, ou smartphone, … ces certificats viennent pré-installés dans les navigateurs/systèmes. Des nouveaux certificats peuvent être installés lors des mises à jours des navigateurs/systèmes ou manuellement par l'utilisateur

Infrastructure à clés publiques (PKI : Public Key Infrastructure)

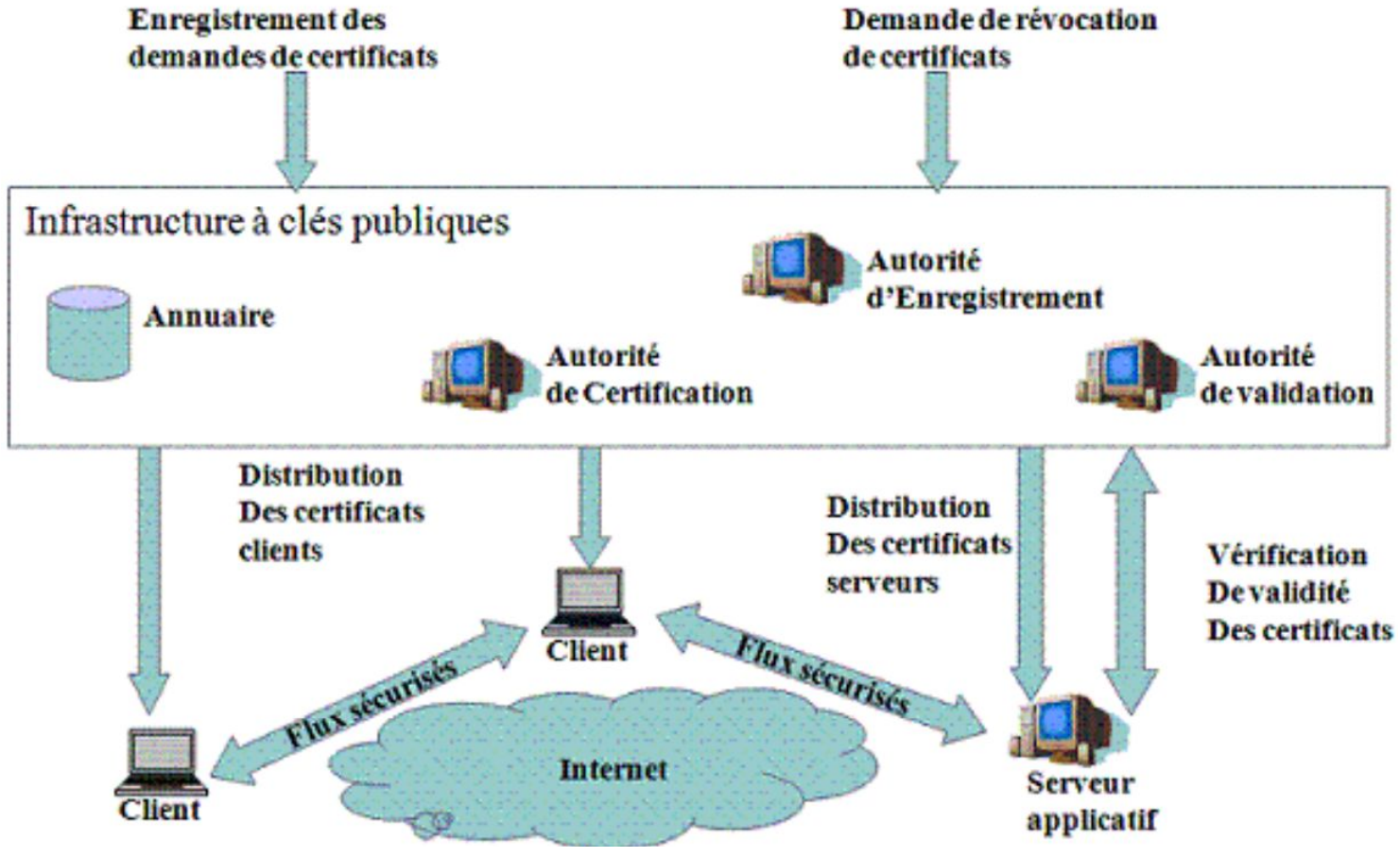
Définition : “ensemble de composants, fonctions et procédures dédié à la **gestion de clés et de certificats** utilisés par des services de sécurité **basés sur la cryptographie à clé publique**”[*]

[*] Politique de certification type: Ministère de l'Economie, des Finances et de l'Industrie, Fr

Fonction d’une PKI : Infrastructure à clés publiques

- Enregistrer et vérifier les demandes de certificats
  - autorité d'enregistrement
- Créer et distribuer des certificats
  - autorité de certification
- Vérification de validité de certificats
  - autorité de validation
- Gérer à tout moment l'état des certificats et prendre en compte leur révocation
  - Dépôt de listes de certificats révoqués – CRL (Certificate Revocation List)
- Publier les certificats dans un dépôt
  - Dépôt de certificats (Annuaire)

Schéma fonctionnel simplifié d'une PKI



Modèles de confiance dans les PKI

- Modèle monopolistique
  - Une CA pour tout le monde
- Modèle monopolistique avec autorités d'enregistrement
  - Une CA avec plusieurs RAs pour la vérification des identités, …
- Délégation de pouvoir de certification
  - Une CA délègue le pouvoir de certification à d'autres entités qui deviennent CA à leur tour, en leur fournissant un certificat qui certifie leur capacité d'être CA.
- Modèle oligarchique
  - Déploiement des produits (comme les navigateur web) avec plusieurs entités de confiance qui sont des CA. Le navigateur fera confiance à tout certificat signé par l'une de ces CA dans sa liste
- Modèle anarchique
  - Chaque utilisateur établit la liste des entités à qui il fait confiance

Validation de certificat

Pour faire confiance à un certificats, il faut réaliser les vérifications suivantes :

Vérification	Commentaire
Signature de l'AC	L'application doit vérifier que le certificat est intègre et authentique
Chemin de certification	L'application doit vérifier qu'il existe une chaîne de certificats valide permettant de remonter à une AC de confiance
Période de validité	L'application doit vérifier que le certificat présenté n'est pas expiré
Statut du certificat	L'application doit vérifier que le certificat n'est pas révoqué (ni suspendu)

Validation de certificat

Il existe par ailleurs différents moyens et techniques standards pour offrir ce service

- Vérification du statut du certificat
  - par récupération régulière de **CRL (Certificate Revocation List)**
- Vérification du statut du certificat en ligne :
  - **OCSP** (On-line Certificate Status Protocol)
- Vérification complète du certificat en ligne :
  - **SCVP** (Simple Certificate Validation Protocol)

Demande de certificat

Le demandeur doit être habilité et authentifié

- Le propriétaire du certificat
- Son supérieur hiérarchique
- Le service de gestion du personnel
- ...

Révocation de certificat

Un certificat peut être révoqué. La révocation intervient quand la fin de validité réelle précède la fin de validité prévue.

La révocation peut avoir plusieurs motifs :

- compromission réelle ou suspectée de la clé privée
- modification d'un au moins des attributs certifiés
- perte de la clé privée (effacement d'un disque dur, perte ou détérioration d'une carte à puce, oubli du code PIN, ...)
- évolution de l'état de l'art cryptographique (la cryptanalyse de la clé privée entre dans le domaine du possible, machines quantiques, ...)
- perte de confiance vis-à-vis d'un acteur ou d'un composant de la PKI

Application : Protocoles SSL/TLS

SSL est un protocole de sécurisation des échanges sur Internet.

- Il fonction sur la couche de transport (sur le protocole TCP).
- Il a été développé en 1994
- Il est devenu officiellement obsolète en 2015 et a été remplacé par TLS.

TLS est le protocole utilisé en ce moment pour la sécurisation des échanges sur Internet.

- Le protocole HTTPS se base sur l'utilisation de TLS avec HTTP.

SSL / TLS principe de fonctionnement

Compose de trois phases :

- Handshake
- Dérivation des clés
- Transfert des données

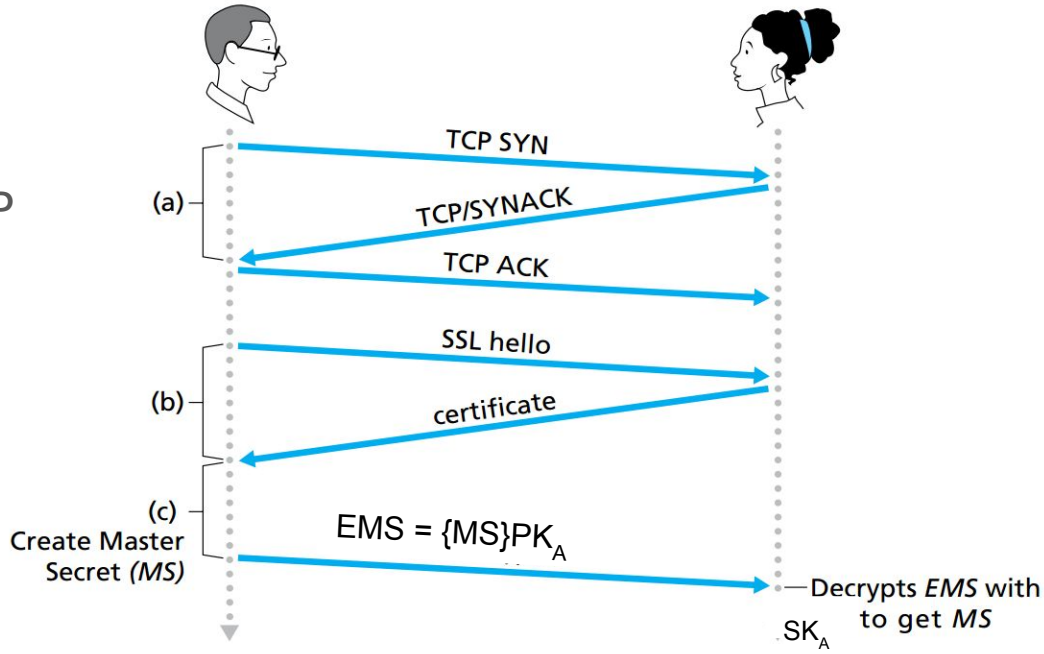
Kurose & Ross : Computer Networking: A Top-Down Approach. 6th edition.

Phase 1 : Handshake

(a) Établissement d'une connexion TCP

(b) vérifier que Alice est vraiment Alice (certificat)

(c) Bob génère une clé partagée Master Key (MS)
- Bob crypte MS avec PK_A EMS =$\{MS\}PK_A$
- Alice décrypte EMS avec SK_A et obtient MS
- MS va être utilisée pour la génération des clés de session.



Phase II : dérivation des clés

- SSL aurait pu utiliser la clé MS pour échanger des messages cryptés entre Alice et Bob

- En réalité : SSL utilise 4 clés dérivées de la cle MS
  - E_B la clé utilisée pour crypter les message de Bob vers Alice
  - M_B la clé utilisée pour vérifier l'intégrité HMAC des messages de Bob vers Alice
  - E_A la clé utilisée pour crypter les message de Alice vers Bob
  - $M_B$la clé utilisée pour vérifier l'intégrité HMAC des messages de Alice vers Bob

Phase III : envoi des données

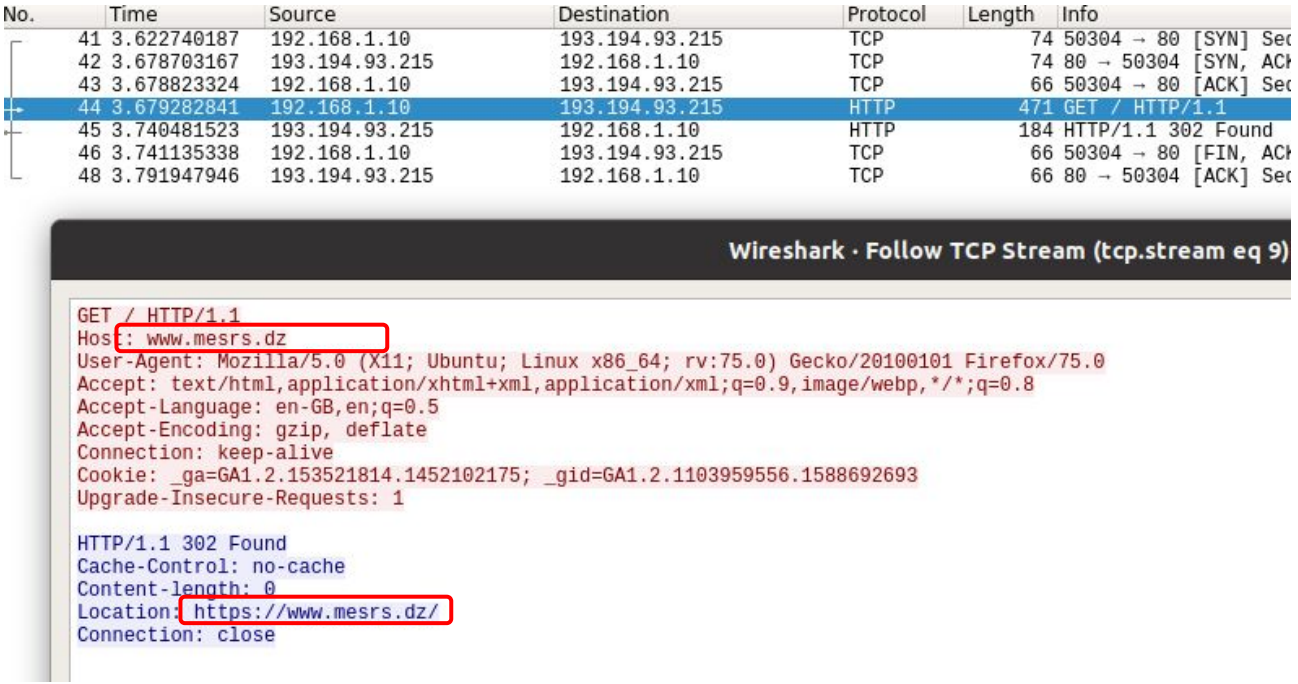
- SSL met les données dans des records

- Type : signifie si le message est
  - Handshake
  - Données (App Data)
  - terminaison de session

Example connexion a www.mesrs.dz

- Lancement de navigateur et taper www.mesrs.dz
- Envoi de requête HTTP GET
- Réception de réponse HTTP/1.1 302 Found (Redirection vers le site certifié https)

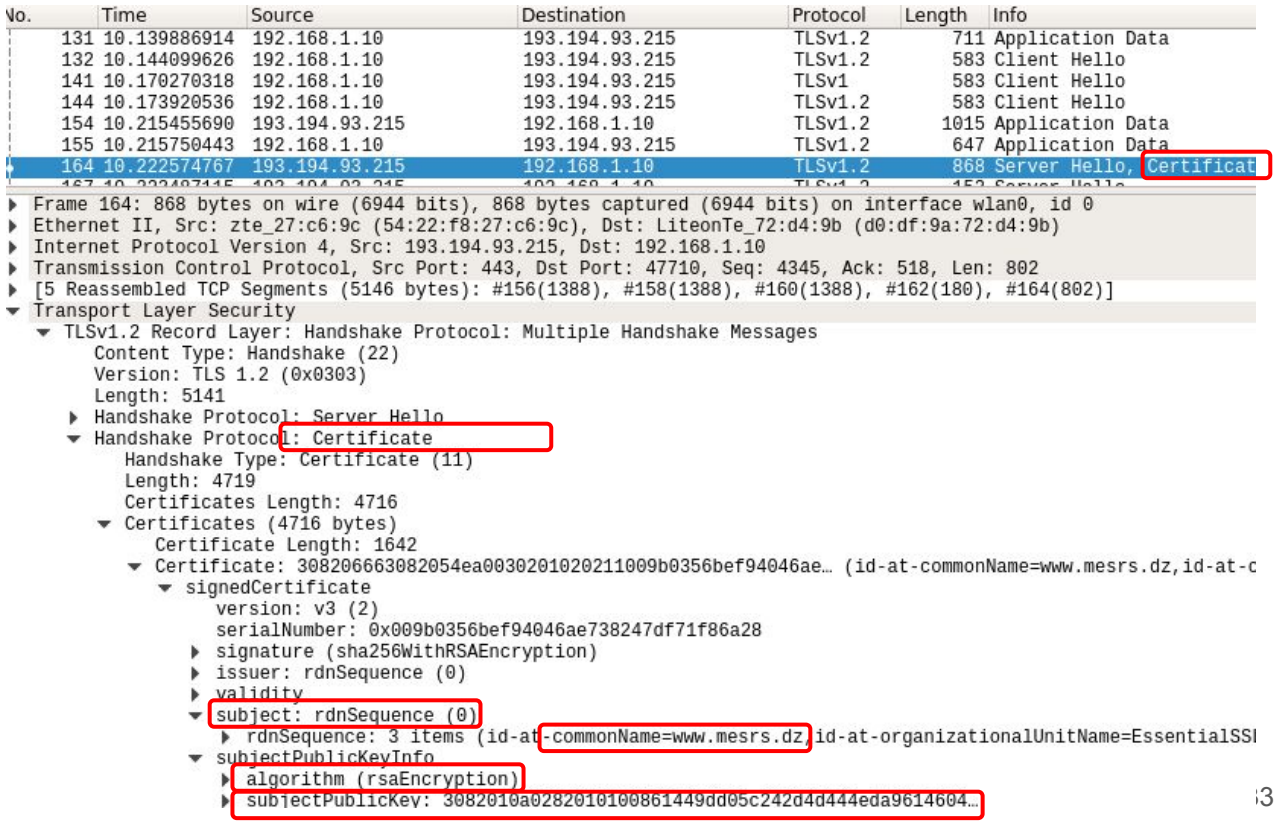
- Le client doit donc utiliser le protocole TLS (ou SSL) dans la suite des échanges



Capture wireshark

Obtention du certificat de mesrs.dz

- Passage au protocole TLS
- Le serveur envoie son certificat contenant son identité et sa clé publique)



Références bibliographiques

Yacine Challal, *Ingénierie des protocoles et logiciels sécurisés*
http://y_challal.esi.dz/index.php/2016/04/30/ingenierie-des-protocoles-et-logiciels-securises-ipls/

James F. Kurose, Keith W. Ross Computer Networking: A Top-Down Approach, 6th Edition (2013)

https://www.pearson.com/us/higher-education/product/Kurose-Computer-Networking-A-Top-Down-Approach-6th-Edition/9780132856201.html