Chiffrement, certificats

Denis Pugnère – CNRS / IN2P3 / IP2I

ANF « IoT perfectionnement »

11 au 15 septembre 2023, Centre Jean-Bosco, Lyon

Plan

- Introduction au chiffrement
- Vocabulaire
- Les primitives cryptographiques
- Les nombres aléatoires
- Fonctions de hachage, utilisations
- Stockage de mots de passe « hashés », « salés »
- Le chiffrement à clé secrète (symétrique)
- Le chiffrement à clé publique (asymétrique)
- La négociation de clés

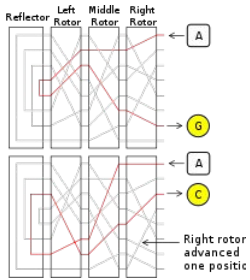
Introduction au chiffrement

- Les **objectifs** du chiffrement :
  - Conserver un **secret**
  - Assurer du secret des communications entre **plusieurs parties**
- N'est pas nouveau, histoire du chiffrement :
  - Sparte (Vème siècle avant notre ère): les premiers codes militaires, la **Scytale**: le premier dispositif matériel de chiffrement (par transposition). Le message est inscrit sur une lanière en cuir enroulée sur un bâton, le destinataire doit posséder un bâton de même diamètre pour être capable de lire le message en clair.
  - Le premier carré magique (« **carré de Polybius** », chiffrement par substitution), 150 ans avant notre ère
  - **Chiffre de César** : Jules César (Ier siècle avant notre ère) utilise une méthode de chiffrement auquel il lègue son nom et qui consiste à décaler les lettres de l'alphabet d'un nombre n de cases (si n=3, A devient D, etc.). C'est la plus ancienne **méthode de substitution** mono-alphabétique.
  - La machine **Enigma**
    - La plus célèbre des machines crypto : Une conception allemande, des rotors, chiffrement par substitution
    - Chiffrement « cassé » par Alan Turing en 1942

Scytale

carré dePolybius

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z



Vocabulaire

- **Cryptographie** : La création et l'utilisation d'écritures secrètes
- **Cryptanalyse** : Le déchiffrement des écritures secrètes
- **Stéganographie** : Les écritures dissimulées
- **Chiffrement** : Passer d'un texte clair à un texte chiffré
- **Déchiffrement** : Passer du texte chiffré au texte clair
- **Décryptage** : Passer du texte chiffré au texte clair sans en connaître la clé ou méthode
- **Cryptage** : ?
- **Texte clair** : Le message original, dans sa forme non dissimulée
- **Clé** : La partie permettant de passer du texte clair au texte chiffré, ou réciproquement
- **Substitution** : Les méthodes de chiffrement remplaçant un signe par un autre
- **Permutation** : Les méthodes de chiffrement modifiant l'ordre des signes
- **Codes** : Les listes arbitraires associant un « code » à un ensemble de signes : Chiffrer -> Coder, Déchiffrer -> Décoder

Primitives cryptographiques

- On a besoin de plusieurs éléments :
  - Des nombres aléatoires : **PRNG** : Les générateurs de nombres aléatoires
  - Des **fonctions** (qui traitent les données) :
    - **Hash(type,données)** : Les fonctions de hachage cryptographiques
    - **chiffrer(algorithme,clé,données)** : Les fonctions qui chiffrent (ou dé-chiffrement) les donnée
  - Des clés de chiffrement :
    - **Chiffrement Symétrique** : À base de « clés secrètes », partagées
    - **Chiffrement Asymétrique** : Bi-clés : 1 publique + 1 privée (secrète)
    - **Clés de session** : Négociation d'une clé de session éphémère
  - Des fonctions de **Signature numérique** :
    Utilisation du chiffrement asymétrique + une fonction de hachage

Les nombres aléatoires

- La qualité de base en cryptographie :
  - Ne pas pouvoir deviner ce qu'une tierce partie a choisi comme base => **Utilisation des nombres aléatoires**
- Les différentes méthodes de génération
  - Le « véritable » aléatoire : dispositif matériel
  - Le pseudo aléatoire déterministe : algorithmes à « source »
  - Le pseudo aléatoire à sources externes
- Le terme « **Entropie** »
  - L'entropie physique est la mesure du « désordre » d'un système
  - L'entropie informatique est appelée ainsi par association
  - X bits d'entropie pour générer Y bits aléatoires
- La solidité des clés ?
  - **Un bon générateur** de nombres aléatoire **génère des clés solides**
  - Un **mauvais** générateur aléatoire génère des **clés moins difficiles à trouver**

Fonctions de hachage, utilisations

- But :
  - **Transformer** de manière déterministe, **une suite de bits** de longueur quelconque, **en un condensat** (aussi appelé **empreinte** ou **haché**), **de taille fixée** (par l'algorithme).

    = Réduire une valeur de grande taille en une valeur plus petite
- Conséquence : La **comparaison de la valeur de hachage** est **plus pratique et plus rapide** que la comparaison octet par octet la source originale
- Caractéristique, elle doit être non inversible, c'est à dire :
  - Il n'est pas possible étant donné un condensat de trouver un message dont l'image par la fonction de hachage est égale à ce condensat
  - Il n'est pas possible de trouver deux messages distincts ayant même condensat (les fonctions de hachage doivent être **sans collision**)
- Un « **hash** » peut aussi être appelé « **empreinte** », à ne pas confondre avec « signature »
- Utilisations des fonctions de hashage :
  - **Le contrôle d'intégrité** = **vérification** si un document a été **modifié** (le changement d'une partie du document change son empreinte),
  - **codes d'authentification de message** (par exemple **HMAC**).
  - **Déduplication** : détecter des fichiers identiques sur un serveur, ou loes blocs de données identiques dans un système de stockage
  - **Stockage de mot de passe hashés**
  - **Signature numérique** : On chiffre un hash avec une clé privée, le résultat peut être utilisé comme signature

Stockage de mots de passe hashés, salés

- Stockage de mot de passe :
  - Ne jamais stocker des mots de passe en clair
  - Ne jamais stocker des mots de passe chiffrés non salés
  - Mais :
    - stockez les mots de passe **hachés** et **salés** : *salt = random ; h = hash(salt,password)*
    - Pour le stockage de mot de passe hashés, **il faut être résistant aux tables pré-calculées** (Rainbow tables) :
  - Illustration (avec des mots de passe **faibles**, **à ne pas utiliser** dans tous les cas :-):

```
$ for i in azerty qwerty 123456; do  echo -n "pw=$i "; echo -n "$i" | sha256sum; done
pw=azerty f2d81a260dea8a100dd517984e53c56a7523d96942a834b9cdc249bd4e8c7aa9  -
pw=qwerty 65e84be33532fb784c48129675f9eff3a682b27168c0ea744b2cf58ee02337c5  -
pw=123456 8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92  -
```

Mot de passe hashé

```
$ for i in {1..3};
   do salt=$(openssl rand -base64 12 |sed 's/=//g'); echo -n "password=azerty salt=${salt} hash=";
   echo -n "$i$salt" | sha256sum | base64 -w0; echo;
done
password=azerty salt=d6nyibaMxMj3vLWz hash=NGE2OTE3MjJlNTQ2ZTUzODY5NzAzMzM4MTgzNjNhZjMxOTViZTRiZDY5MzUyMmFlZDFkYzZiMzllOWE5YzU0OSAgLQo=
password=azerty salt=oyrmFINLJafuW+3d hash=OTQ3MTVmYzVjNGE3ZWUwOGI0YWYxNDk4MzE3NWJiNmU0NmI1MTY2YjI5Nzg5MTgxOTJjN2FjNjM2YzQwOTMyMyAgLQo=
password=azerty salt=EAy1CWSipN8zQaiM hash=MDk1NTE3NDM3MWFlZTBhY2M4ZjNjYjFmZWJkMWFkYWE4ZTk3ZGE1ZDJmYTJlZWM2MDNkMjdiODhiNDIxMDAyMCAgLQo=
```

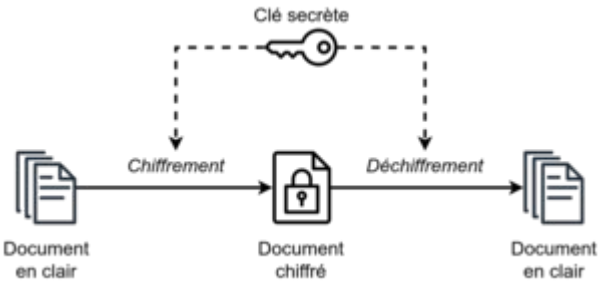
Mot de passe hashé et salé

```
$ echo "azerty" | openssl passwd -6 -stdin
$6$2nKfStgOCwCIcs5m$bS.XVACmW8LFHbAq5GPzwEr5T64i/eph3pa7AVobSElE9lqzD5Du3dHN0dyKf2hYYcRblUiiq2BagFopP7NEt1
```

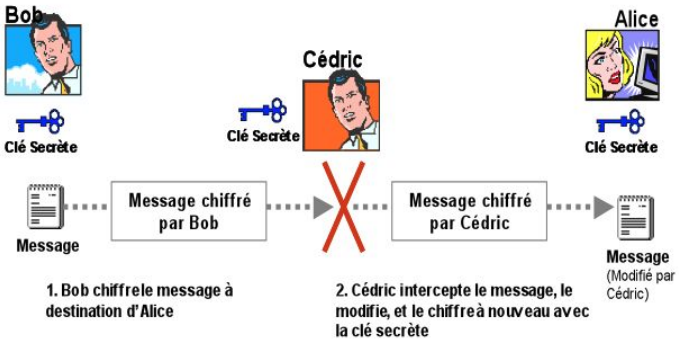
C’est celui-ci que l’on peut stocker dans une base de données

Le chiffrement à clés secrètes (symétrique)

- Les principes de base du **chiffrement à clé secrète**...mais partagée
  - La même clé est utilisée pour chiffrer et déchiffrer
  - La sécurité des données réside dans la solidité de la clé (et sa non divulgation)
  - On l'appelle aussi « **chiffrement à clé symétrique** »
- Les deux modes de chiffrement
  - Le chiffrement **par flot**
  - Le chiffrement **par bloc**

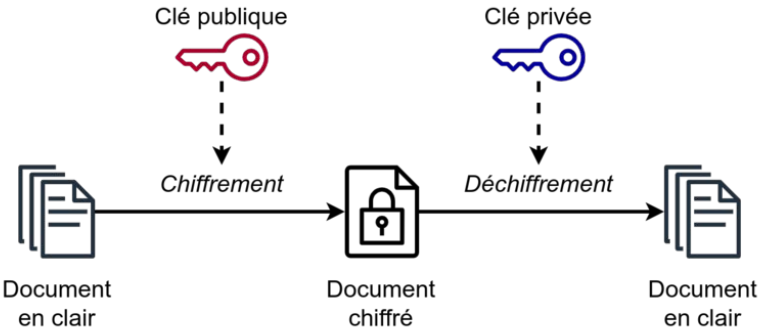


- Liste d'algorithmes symétriques communs : RC4 + RC5 (obsolètes), DES (obsolète), Triple-DES, AES, Blowfish, Serpent, Twofish
- Se référer aux guides de l'ANSSI pour **choisir la taille de la clé** (contre les attaques **exhaustives** ou par **dictionnaire**)
- Les **avantages** du chiffrement à clé secrète
  - **Rapidité**
  - Implémentation sur du matériel (primitives AES sur les processeurs ou dans l'embarqué…)
- Les **inconvénients** du chiffrement à clé secrète
  - Chiffrement **non authentifié**, Identité de l'expéditeur ?
  - Pas d'intégrité du message => rajouter un mécanisme d'authentification de message (MAC)



Chiffrement par clés publiques (asymétrique)

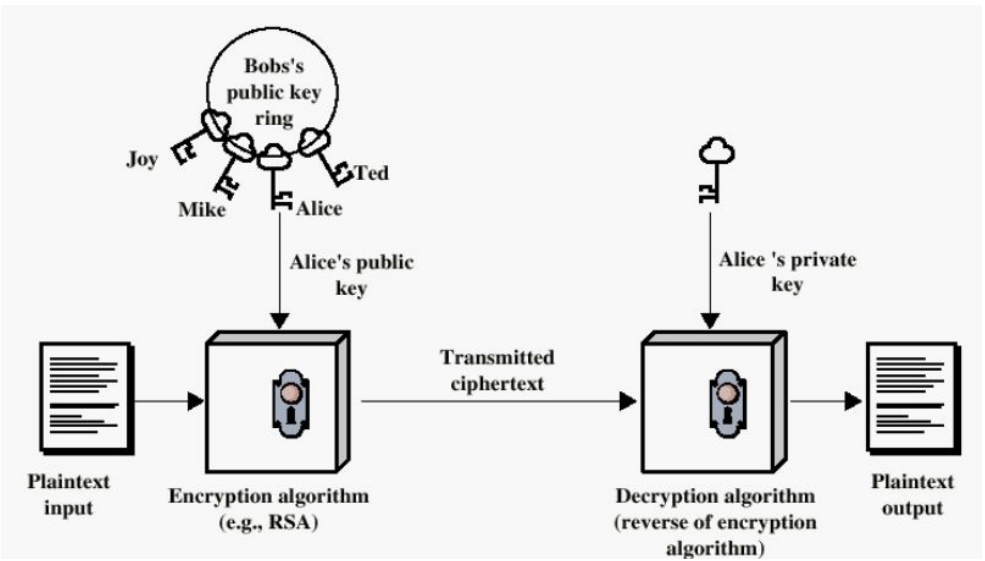
- Deux grandes familles mathématiques
  - Les facteurs premiers (RSA) : N = P x Q
  - Les courbes elliptiques (Curve25519) et logarithmes « discrets »
- Aussi appelé chiffrement à **clés asymétriques**
  - Une clé « maître » est choisie, on en déduit **une paire** de clés
  - Il est possible de divulguer une partie des clés
  - L'une des **clés (publique)** est la clé que l'on peut publier



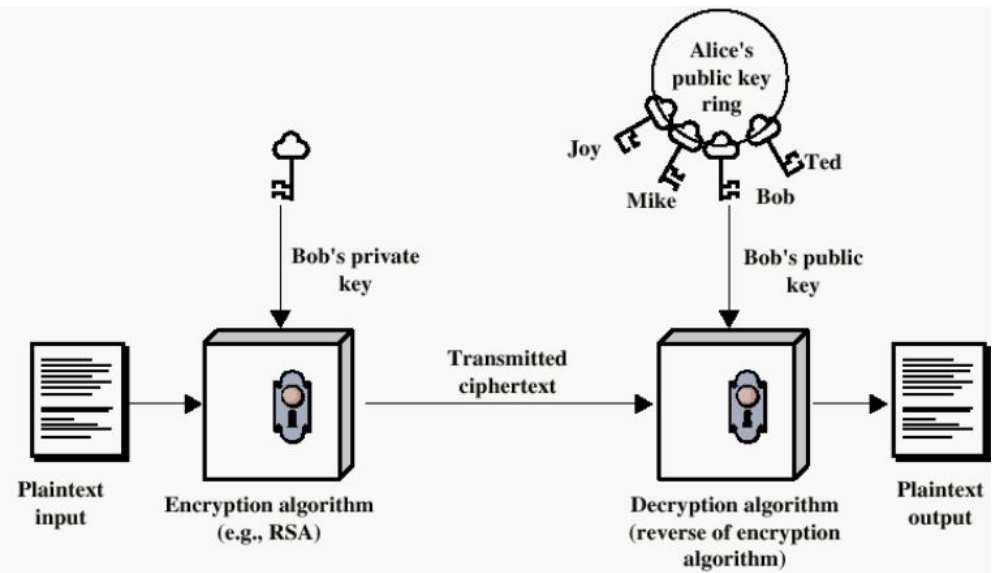
  - L'autre **clé (privée)**, qui doit impérativement **être protégée**, stockée elle même de manière **chiffrée**
  - Il n'existe pas de fonction « simple » permettant de déterminer une clé à partir de l'autre
  - Il n'existe pas de fonction permettant de retrouver la clé maître à partir d'une clé
  - Il n'est pas possible de chiffrer ou de déchiffrer uniquement à l'aide de la partie publique
- La grande difficulté de la cryptographie a clef publique
  - Les performances moindre (facteur 100 à 1000) que chiffrement à clés symétriques
  - Les tailles des clefs pour les clés RSA (> 2048 bits)
  - Problématique de la diffusion de la clé publique (canal de confiance, annuaire...)

Chiffrement versus Authentification (asymétrique)

Chiffrement : Bob utilise
la clé publique du destinataire Alice



Authentification : Bob utilise sa clé privée pour chiffrer et Alice la clé publique de Bob pour déchiffrer (et authentifier l'expéditeur)



Clés de session

- Le **chiffrement à clés asymétriques plus lent** (facteur 100 à 1000) **que le chiffrement à clés secrètes** (symétrique) :
- Introduction du concept de « **clé de session** » :
  - Déterminer une clé de session **éphémère, jetable** basée sur une clé secrète (**symétrique**)
  - Le principe est de calculer une clé sans qu'une tierce partie ne puisse la trouver, cette **clé est calculée par les 2 parties, dans un canal chiffré (par des clés asymétriques)**
  - Exemple : Échange de clés Diffie-Hellman (DH) [1]
  - **Utilisée qu'une seule fois**, pendant un **laps de temps donné**, pour le chiffrement et le déchiffrement de données entre deux parties durant cette seule conversation
  - les futures conversations entre les deux parties seront chiffrées avec des clés de session différentes. Une clé de session est comme un mot de passe que quelqu'un réinitialise à chaque nouvelle fois qu'il se connecte.

    [1] : https://fr.wikipedia.org/wiki/Échange_de_clés_Diffie-Hellman

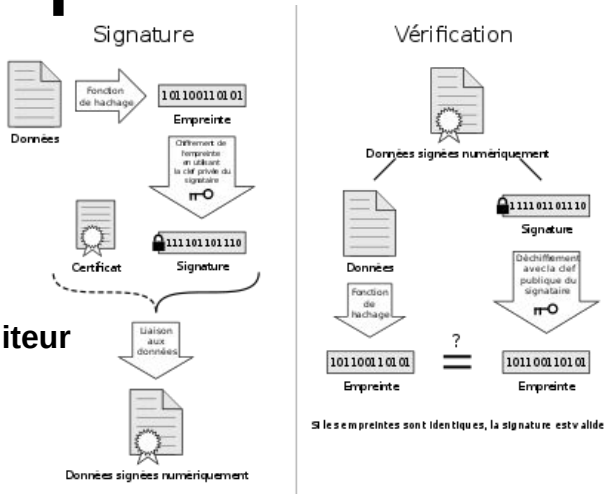
Vérification authenticité par clés publiques (asymétrique)

- Les clés publiques
  - **doivent** généralement **être distribuées de façon authentique**
    - Via un canal de communication de confiance
    - Via une **Infrastructures de Gestion de Clés (IGC)** / Public Key Infrastructure (PKI)
- **Comment vérifier l'authenticité** des parties ? un serveur (SSH, https…) ou un utilisateur (certificat X509, bi-clés ssh RSA ou autre) ?
  - SSH : vérification de l'authenticité du serveur et de l'utilisateur
    - Vérification de l'authenticité du serveur **via empreinte de sa clé privée** présentée lors de la connexion (et de la comparaison par rapport à l'empreinte précédente stockée dans ***.ssh/known_hosts*** *du client*)
      => Mécanisme de TOFU (Trust On First Use)
    - De l'utilisateur par la transmission d'un challenge chiffré par la clé publique de l'utilisateur (stockée dans ***.ssh/authrorized_keys*** du serveur) et déchiffré par la clé privée de l'utilisateur (stocké dans **.ssh/id_rsa** par exemple)
  - HTTPS : vérification via la confiance que l'on place dans le certificat présenté par le serveur, car le certificat est signé par une AC (Autorité de Certification)

La signature numérique

Permet de garantir d'identifier l'émetteur et l'intégrité d'un document

- **Signer un message**
  - **Chiffrer** un message **avec sa clé privée**
  - N'importe qui (connaissant la clé publique), peut déchiffrer le message
  - Comme seul l'émetteur connaît la clé privée, nous avons les propriétés suivantes :
    => on a la certitude que seul l'émetteur a pu générer le message => **authenticité de l'expéditeur**
    => opposable à l'émetteur (**irrévocable, non-répudiation**) : il ne peut pas nier être l'auteur
  - Ces propriétés ne sont pas possibles avec un mécanisme symétrique
- **Chiffrement asymétrique + hachage = La signature numérique [1]**
  - Message **haché + chiffré par une clef privée**
  - Si le **déchiffrement par la clef publique** donne le bon hachage, les données sont donc les bonnes (**intègres + authentiques**)
- **La certification de la clef publique**
  - La signature ne prouve que **quelqu'un dispose de la clef privée**, mais **pas l'identité de cette personne**
  - L'importance de disposer d'un mécanisme de distribution des clés (IGC) ou **d'authentification de clef publique de confiance**

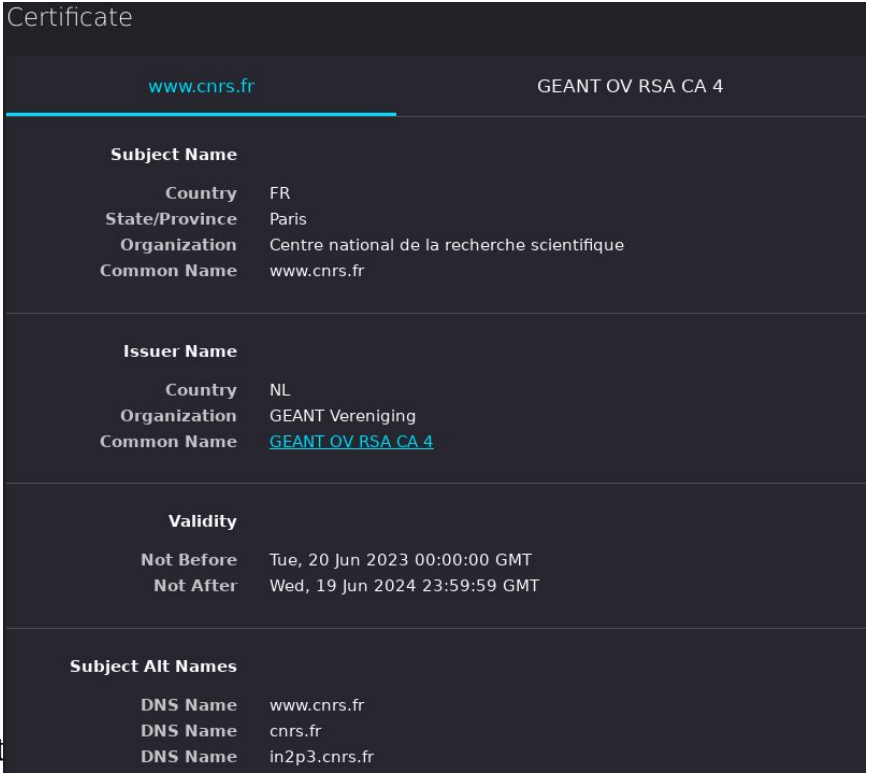


[1] : https://fr.wikipedia.org/wiki/Signature_numérique

Les certificats X509

- **Le certificat** atteste l'**association d'<u>une clef</u> avec une <u>entité</u>, <u>vérifiée</u>** par une **<u>organisation</u>**
- Utilisation du chiffrement asymétrique (clé publique + clé privée)
  - Un certificat X509 associe la clef (publique) avec un « nom »
  - Le nom est un nom X500 … ou un nom « alternatif » (nom de machine, adresse électronique), exemple ici Certificat de ***www.cnrs.fr*** valable aussi pour ***cnrs.fr*** , ***in2p3.cnrs.fr*** …
- Pour générer certificat SSL il est nécessaire de générer une CSR (Certificate Signing Request (Demande de Signature de Certificat) :
  - CSR : texte chiffré qui précise de manière unique qui vous êtes et quel nom de domaine
  - Clé privée (qui va prouver que vous avez le certificat)
- Le certificat dit que vous êtes « Untel » ici ***www.cnrs.fr***
- Un certificat doit être signé par une autorité de certification (ici par « GEANT Vereniging » pour prouver qu'il n'a pas été altéré

  Lors de la demande du certificat par le CNRS, **le signataire** (GEANT Vereniging), **<u>après contrôle</u>**, utilise sa clef privée pour **signer** le certificat ***www.cnrs.fr***
- Une fois signé par l'autorité de certification (GEANT Vereniging), des attributs sont rajoutés (numéro de série, DN, durée de validité...)
- Infrastructure de gestion de clefs « PKI » : Autorité de certifications enregistrées (ou pas) dans les navigateurs
- Vérification



Guides, référentiels

- https://www.ssi.gouv.fr/guide/mecanismes-cryptographiques/
  - Guide de sélection d'algorithmes cryptographiques
  - Règles et recommandations concernant le choix et le dimensionnement de mécanismes cryptographiques