INSA | INSTITUT NATIONAL DES SCIENCES APPLIQUÉES CENTRE VAL DE LOIRE

Module : Sécurité Réseaux

Outils d'authentification et de chiffrement

4A STI Année 2020-2021

M. SZPIEG Promo 2022

Les outils de la cryptographie et des signatures

Les fonctions de hachage

Elles prennent en entrée un message de taille variable, le compressent et produisent une empreinte (digest) de taille fixe.

Le résultat d'une fonction de hachage est identique pour deux entrées identiques.

Les outils de la cryptographie et des signatures

Les fonctions de hachage

Les fonctions de hachage permettent difficilement (ou ne permettent pas) de retrouver le texte qui a donné une empreinte donnée.

De plus une fonction de ce type doit éviter les collisions. C'est-à-dire que deux textes différents doivent avoir une probabilité très faible de donner la même empreinte.

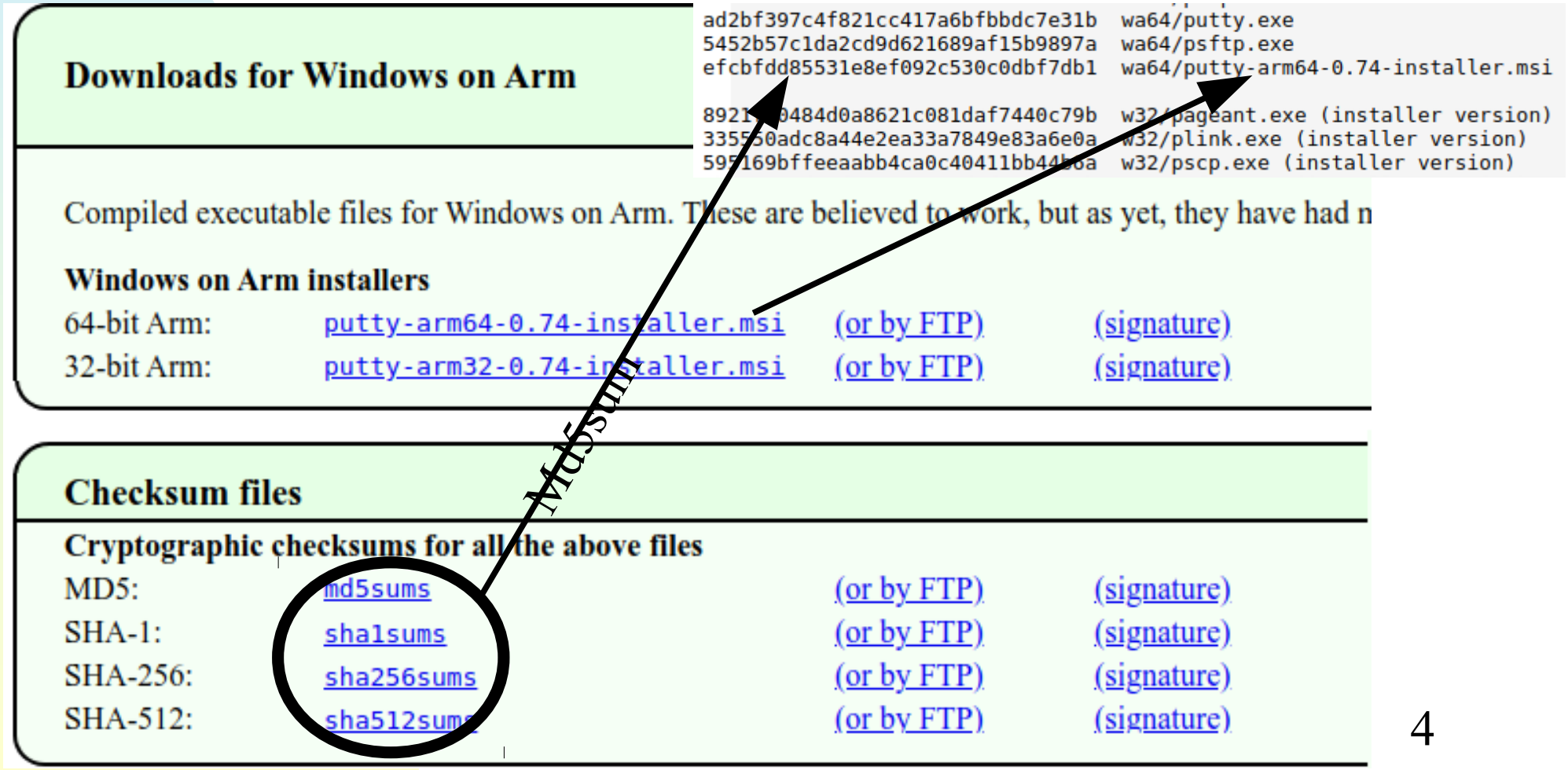
Les fonctions les plus courantes :

- MD5(Message Digest 5)
  génère une empreinte de 128 bits à partir d'un texte de longueur quelconque
- SHA (Secure Hash Algorithm)
  génère une empreinte de 160 bits.
- RIPE-MD (Race Integrity Primitives Evaluation)
  émise par la communauté européenne, génère une empreinte de 128 bits ou 160 bits.

Les outils de la cryptographie et des signatures

Les fonctions de hachage : utilisation pratique, intégrité des fichiers.

Site : https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html



Les outils de la cryptographie et des signatures

Le chiffrement

Ces algorithmes assurent la transformation d'un message en clair ("plaintext") en un message brouillé (ciphertext)

Il existe deux grandes familles d'algorithmes :

- Ceux qui imposent au système qui chiffre de savoir déchiffrer (algorithmes symétriques).
- Ceux qui ne permettent pas au système qui chiffre de déchiffrer (algorithmes asymétriques).

Pour les deux cas les algorithmes de chiffrement sont commutatifs.

Deux familles d'algoritmes symétriques

Les modes de chiffrement par blocs

On distingue dans les d'algorithmes symétriques deux modes :

- Le mode ECB (Electronic Code Book)
  traitement d'un bloc en code brouillé indépendamment des autres blocs.
  Inconvénient : un même bloc produit le même code (entête IP)

- Le mode CBC (Cipher Block Chaining)
  utilisation des blocs précédemment envoyés pour coder le bloc courant.

Algoritmes symétriques courants :

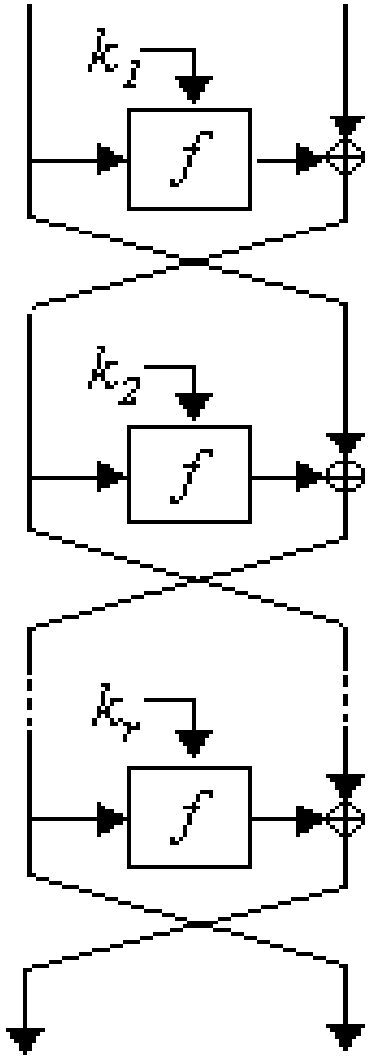
Chiffrement par blocs (block cipher)

- Blowfish, longueur variable (32 à 448 bits) free
- CAST(Carlisle Adams Statford Tavares)
- DES (Data Encryption Standard) free
  1977, 64 bits de clé dont 56 utiles (controle de parité), chiffre de FEISTEL à 16 rondes. Variante le triple DES 112 bits.
- IDEA (International Data Encryption Algorithm)
  année 1991, itération 128 bits et 8 rondes.
- AES(Advance Encryption Standard) remplace le triple DES 128, 192, 256 bits plus rapide que IDEA et triple DES free.

Chiffrement par flux

RC4 (Rivest Cypher) (pb de synchronisation).

Le chiffre de FEISTEL, blocs avec itérations



On chiffre les blocs par un processus comportant plusieurs rondes. Dans chaque ronde, la même transformation est appliquée au bloc, en utilisant une sous-clef dérivée de la clef de chiffrement.

Exemple la famille des chiffres de Feistel. Un bloc de texte en clair est découpé en deux ; la transformation de ronde est appliquée à une des deux moitiés, et le résultat est combiné avec l'autre moitié par ou exclusif. Les deux moitiés sont alors inversées pour l'application de la ronde suivante.

Algorithmes symétriques comparaisons

Source : http://www.schneier.com/blowfish-speed.html

Bruce Schneier

Speed Comparisons of Block Ciphers on a Pentium

Algorithm	Clock cycles per round	# of rounds	# of clock cycles per byte encrypted	Notes
Blowfish	9	16	18	Free, unpatented
Khufu/Khafre	5	32	20	Patented by Xerox
RC5	12	16	23	Patented by RSA Data Security
DES	18	16	45	56-bit key
IDEA	50	8	50	patented by Ascom-Systec
Triple-DES	18	48	108	

Les algorithmes asymétriques courants :

Algorithmes à clé publique (asymétrique)

On utilise une paire de clés

- l'une publique qui chiffre le message, cette clé est inefficace pour le déchiffrer.
- L'autre privée qui sert à décoder le texte.

Le symétrique fonctionne aussi : un message chiffré avec la clé privée peut être déchiffré avec la clé publique.

Le plus connu RSA (Ron Rivest,Adi Shamir et Leonard Adleman).Particularité si l'une des deux clefs chiffre les données seule l'autre pourra déchiffrer le message. D'où un intérêt certain pour la non-répudiation.

Basé sur la factorisation des grands nombres.

https://www.apprendre-en-ligne.net/crypto/rsa/251_088_096.pdf

Les algorithmes asymétriques courants :

Algorithme asymétrique pour l'authentification El-Gamal, du nom de son inventeur.

Inconvénient : Le texte brouillé représente deux fois la longueur du texte clair.

Ce principe est utilisé par DSA **(Digital Signature Algorithm) :**
En fait DSA consiste à générer deux valeurs de 160 bits avec la clé privée, puis on démontre du coté du récepteur en utilisant la clé publique que seule cette clé privée pouvait générer ces valeurs. Il n'y a donc pas de chiffrement proprement dit mais une authentification.

DSA utilise une fonction de hachage pour la signature
SHA **(Secure Hash Algorithme)**

Code d’authentification des messages

Pour authentifier un message, on peut le signer en chiffrant la totalité du message avec une clef secrète. Ceci donne des méthodes lentes lors de l'exécution. Pour vérifier, il faut traiter tout le message.

Aussi, il existe des méthodes plus rapides pour garantir l’intégrité et authenticité, ce sont les « codes d’authentification de message » MAC (Message Authentification Code).

Pour générer un MAC, il suffit de hacher le message en le combinant (OU exclusif) à un « secret partagé » appelé « clé secrète ». Puisqu’on utilise les fonctions de hachage, on parle de HMAC.

URL :

https://fr.wikipedia.org/wiki/Keyed-hash_message_authentication_code#:~:text=Un%20HMAC%2C%20de%20l'anglais,fonction%20de%20hachage%20cryptographique%20en

Les plus connus : HMAC-SHA ou du HMAC-MD5

Code d'authentification des messages

$$\mathrm{HMAC}_K(m) = h\Big((K \oplus opad) \mid\mid h\big((K \oplus ipad) \mid\mid m\big)\Big)$$

avec :

- *h* : une fonction de hachage itérative,
- *K* : la clé secrète, hachée par la fonction h si plus longue que sa taille de bloc, puis complétée avec des zéros pour qu'elle atteigne la taille de bloc de la fonction h
- *m* : le message à authentifier,
- "||" désigne une concaténation et "⊕" un « ou » exclusif,
- *ipad* et *opad*, chacune de la taille d'un bloc, sont définies par : *ipad* = 0x363636...3636 et *opad* = 0x5c5c5c...5c5c. Donc, si la taille de bloc de la fonction de hachage est 512 bits, *ipad* et *opad* sont 64 répétitions des octets, respectivement, 0x36 et 0x5c.

Les protocoles cryptographiques

Les protocoles cryptographiques sont une série d'étapes prédéfinies, basées sur un langage commun, qui permettent à plusieurs participants (généralement deux) d'accomplir une tâche : ce peut être une authentification, un échange de clef,...

Une particularité des protocoles cryptographiques est que les tiers en présence ne se font généralement pas confiance et que le protocole a donc pour but d'empêcher l'espionnage et la tricherie.

Exemples de protocoles :

- SSL (Secure Sockets Layer)
- TLS (Transport Layer Security)

Les protocoles cryptographiques

Les machines communicant entre-elles ne se faisant pas confiance, elle vont faire appel à un « tiers de confiance ».

Ce tiers de confiance, organisme extérieur aux deux hôtes qui communiquent, va participer à l'échange de clefs et à l'authentification des deux éléments qui échangent des données pour augmenter la confiance de la transaction.

Exemples :

- Kerberos
- Sésame
- infrastructure à clefs publiques (PKI) :
  - X.509
  - Openpgp
- ...

Les échanges de clés

La première méthode d'échange de clés à avoir été décrite fut celle de Diffie-Hellman (1976). Cette méthode repose sur une fonction à sens unique du logarithme secret.

Les échanges de clés publiques vont se faire, en générant un secret partagé sur un réseau non sécurisé.

Les deux hôtes qui communiquent vont se transmettre en clair quatre nombres entiers qui vont leur permettre de générer un secret partagé sans qu'un tiers qui aurait récupéré ces quatre valeurs puissent calculer le secret partagé.

URL : https://fr.vikidia.org/wiki/Algorithme_Diffie-Hellman#:~:text=Diffie%2DHellman%20est%20un%20algorithme,article%20New%20Directions%20in%20Cryptography.

Cette méthode est tout de même sensible à l'attaque par interposition (man in the middle), néanmoins cet inconvénient peut être résolu en signant les échanges.

Notion de PFS (Perfect Forward Secrecy)

PFS peut se traduire par sécurité persistante.

Le but est de garder le secret des informations transmises bien après leur transmission.

Sachant que le calculateur A communique avec le serveur B en chiffrant les données avec la clé publique de B. Une fois les données chiffrées seul B est en capacité de retrouver le message original.

Scénario : un calculateur C « man in the middle » stocke les données chiffrées qui transitent sur un disque dur. Deux mois après l'échange le propriétaire de l'ordinateur C arrive à récupérer la clé privée du serveur. Il peut alors déchiffrer les données chiffrées qu'il avait stockées.

Le PFS définit le chiffrement d'une connexion qui rend le scénario ci-dessus inopérant. Ceci signifie que, même avec la clé privée du serveur, C ne pourra pas déchiffrer les données stockées sur le disque dur.
On utilise, en général, Diffie-Hellmann pour assurer la qualité de PFS.

Les infrastructures à clés publiques (PKI)

PKI Public Key Infrastructure, définition :

Ensemble de matériels et de logiciels permettant de mettre en œuvre une confiance numérique.

En pratique, le principe consiste à employer des techniques de chiffrement asymétrique dont les clés publiques sont certifiées par une autorité de certification.

Dans ce but une PKI doit fournir des certificats numériques contenant des clefs publiques et un service de stockage de diffusion et de révocation des clés.

Autorités de certification CA :

Une PKI (Public Key Infrastructure) doit remplir les rôles suivants :

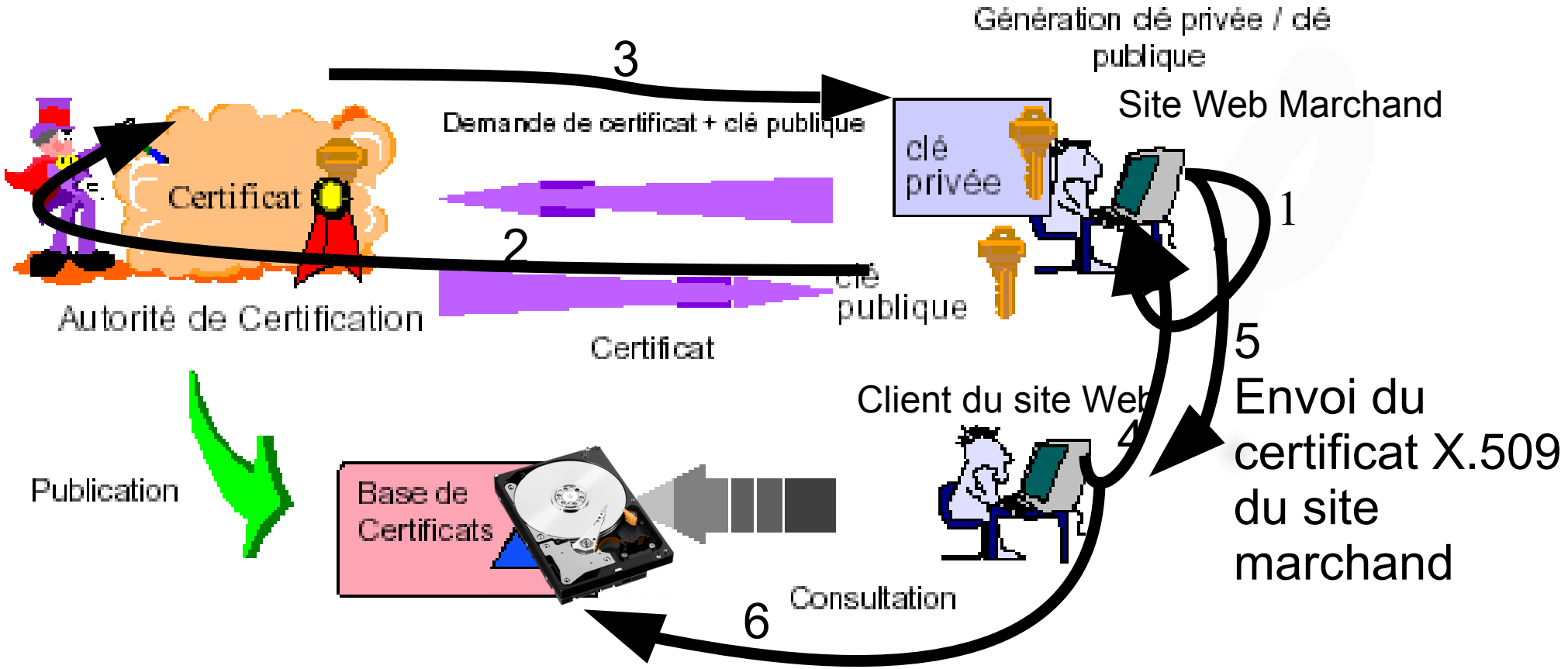
- Signer des clés publiques pour l'authentification d'un serveur, d'un utilisateur, ou d'un programme exécutable par l'intermédiaire de centres certificateurs (aussi appelés autorités de certification « Certificate Authority » CA) qui utilisent leurs propres clés privées pour faire cette opération.
  Le résultat est mis sous la forme d'un fichier appelé certificat X.509 qui est transmis au demandeur souvent après paiement.
- Gérer des serveurs de données dont le rôle consiste à stoker et diffuser les certificats des centres certificateurs agréés sur la planète et à stocker les certificats des clients X.509.
- Gérer la diffusion des certificats des centres certificateurs lors de la mise à jour des systèmes d'exploitation (Windows Update, Linux apt-get Aptitude … , MAC OSX Appstore, etc.)
- Permettre à un client de résilier son certificat X.509.

Autorités de certification CA :

Paquets de mises à jour d'un système d'exploitations Linux :

```
martial@acerfix ~ $ dpkg -l
||/ Nom                  Version                  Architecture Description
ii  ca-certificates      20201027ubuntu0.16.04.1  all          Common CA certificates
ii  dirmngr               2.1.11-6ubuntu2.1        amd64        server for managing certificate revocation lists
```

1 Génération d'une paire une privée (gardée jalousement) et une publique.
2 Demande d'un certificat contenant la clé publique du site web signée.
3 Retour du certificat avec la clé signée.
4 Consultation du site par le client avec demande d'authentification (https)
5 Réception du certificat du serveur Web par la machine du client.
6 Vérification du certificat du site Web par le poste du client en trouvant sur son disque dur la clé publique de l'autorité de certification.



Ciphers suite :

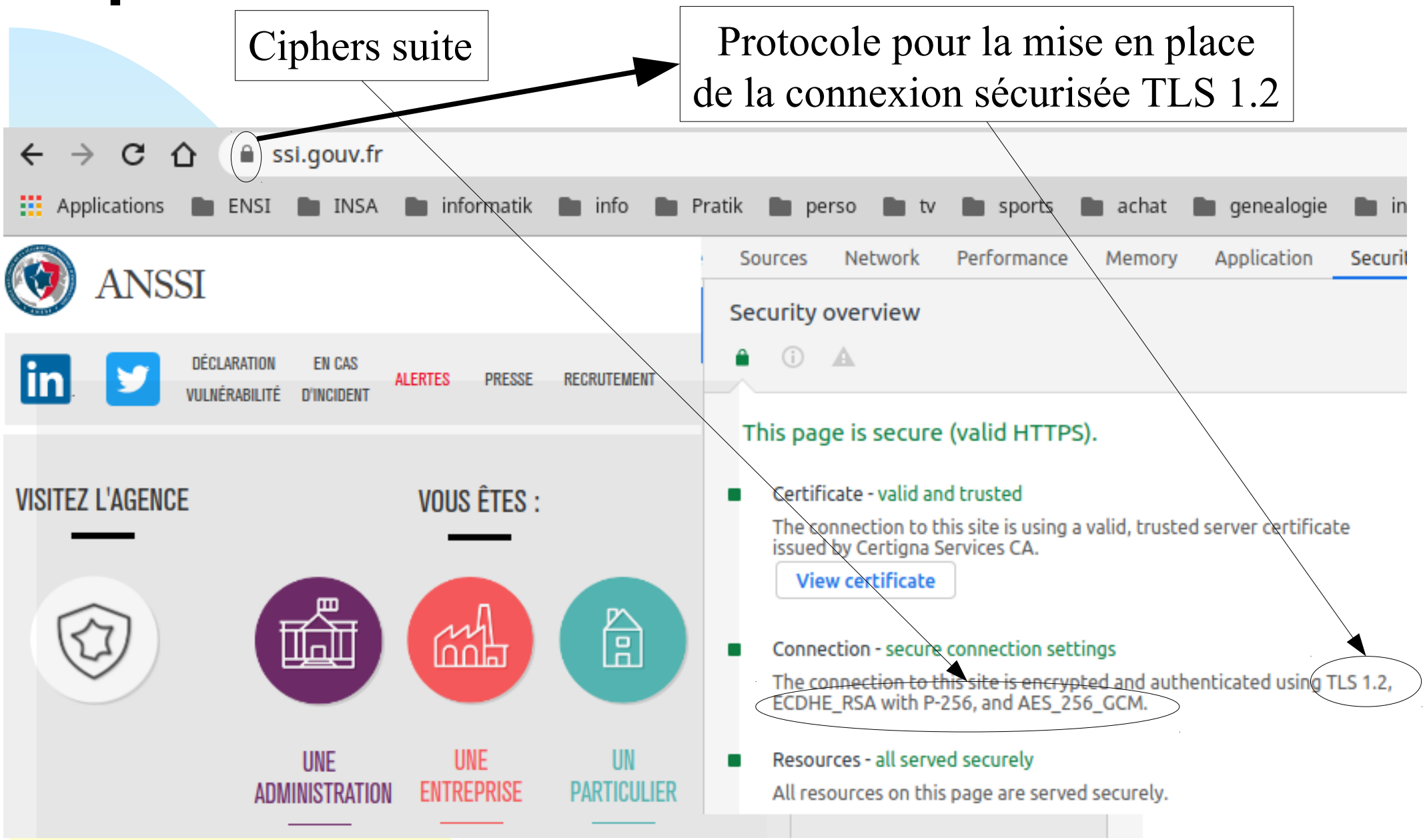
Certaines connexions réseaux utilisent plusieurs algorithmes afin de sécuriser et authentifier les éléments transmis.

Cette suite algorithmes est appelée « ciphers suite » .

Listes des ciphers suites sous une machine Linux :

```
martial@acerfix ~ $ openssl ciphers
ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECD
HE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:SRP-DSS-AES-256-CB
C-SHA:SRP-RSA-AES-256-CBC-SHA:SRP-AES-256-CBC-SHA:DHE-DSS-AES256-GCM-SHA384:DHE-RSA-A
ES256-GCM-SHA384:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA256:DHE-RSA-AES256-SHA:DHE-D
SS-AES256-SHA:DHE-RSA-CAMELLIA256-SHA:DHE-DSS-CAMELLIA256-SHA:ECDH-RSA-AES256-GCM-SHA
384:ECDH-ECDSA-AES256-GCM-SHA384:ECDH-RSA-AES256-SHA384:ECDH-ECDSA-AES256-SHA384:ECDH
-RSA-AES256-SHA:ECDH-ECDSA-AES256-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:CAME
LLIA256-SHA:PSK-AES256-CBC-SHA:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA
256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDS
A-AES128-SHA:SRP-DSS-AES-128-CBC-SHA:SRP-RSA-AES-128-CBC-SHA:SRP-AES-128-CBC-SHA:DHE-
DSS-AES128-GCM-SHA256:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES128-SHA256:DHE-DSS-AES128-
SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA:DHE-RSA-SEED-SHA:DHE-DSS-SEED-SHA:DHE-RS
A-CAMELLIA128-SHA:DHE-DSS-CAMELLIA128-SHA:ECDH-RSA-AES128-GCM-SHA256:ECDH-ECDSA-AES12
8-GCM-SHA256:ECDH-RSA-AES128-SHA256:ECDH-ECDSA-AES128-SHA256:ECDH-RSA-AES128-SHA:ECDH
-ECDSA-AES128-SHA:AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:SEED-SHA:CAMELLIA128-SHA
:PSK-AES128-CBC-SHA:ECDHE-RSA-RC4-SHA:ECDHE-ECDSA-RC4-SHA:ECDH-RSA-RC4-SHA:ECDH-ECDSA
-RC4-SHA:RC4-SHA:RC4-MD5:PSK-RC4-SHA:ECDHE-RSA-DES-CBC3-SHA:ECDHE-ECDSA-DES-CBC3-SHA:
SRP-DSS-3DES-EDE-CBC-SHA:SRP-RSA-3DES-EDE-CBC-SHA:SRP-3DES-EDE-CBC-SHA:EDH-RSA-DES-CB
C3-SHA:EDH-DSS-DES-CBC3-SHA:ECDH-RSA-DES-CBC3-SHA:ECDH-ECDSA-DES-CBC3-SHA:DES-CBC3-SH
A:PSK-3DES-EDE-CBC-SHA
```

Ciphers suite sous Chrome 1/2 :



Ciphers suite sous Chrome 2/2 :