Fondamentaux de la sécurité réseau

La Revue 3EI
Culture Sciences de l'Ingénieur

Édité le 16/02/2024

école normale supérieure paris-saclay

Maxime SECHEHAYE[1] - Anthony JUTON[2] - Magali SAUVERGEAT[3]

[1] *ENS Paris-Saclay – DER Nikola Tesla*
[2] *Professeur agrégé à l'ENS Paris-Saclay – DER Nikola Tesla*
[3] *Enseignante BTS CIEL, Arpajon*

Cette ressource fait partie du N°111 de La Revue 3EI de janvier 2024.

Si les attaques les plus nombreuses viennent de l'intérieur après une arrivée par mail (virus ou cheval de Troie téléchargé par un membre du personnel), nombre d'attaques parmi les plus spectaculaires exploitent les vulnérabilités des communications réseau. Avant d'aborder les différents champs d'application de la cybersécurité des systèmes industriels, cette ressource rappelle les principes fondamentaux de la sécurité réseau : confidentialité, intégrité, disponibilité, authentification et non-répudiation. Elle détaille ensuite le protocole TLS, très populaire, qui illustre trois de ces principes. La ressource se prolonge par une liste de vidéos d'illustration ou d'approfondissement, en français et en anglais, pouvant servir de support pour des séquences de classe inversée ou un co-enseignement anglais-GEII.

1 - Principes fondamentaux

Cette première partie présente les principes fondamentaux (confidentialité, intégrité, disponibilité, authentification et non-répudiation) de la cybersécurité qui, s'ils sont correctement implémentés, garantissent la protection via le réseau, excepté en cas d'exploitation de failles non documentées, ce qui est hors du champ de ce dossier.

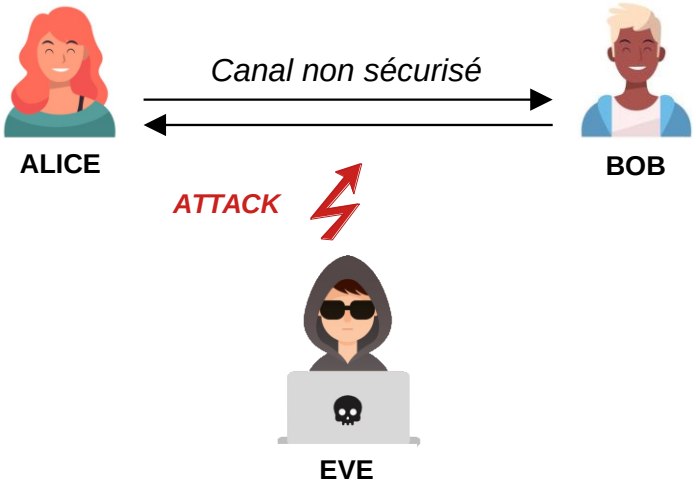


Figure 1 : Schéma simplifié d'une attaque

On imagine qu'Alice et Bob veulent échanger et que Eve souhaite nuire à leur relation. Alice et Bob peuvent être par exemple :

- Des personnes,
- Un terminal de paiement et un serveur de banque,
- Un automate industriel et sa supervision,
- Des calculateurs embarqués automobiles,
- Un smartphone et une serrure connectée.

Les communications sans fil peuvent être interceptées facilement et les communications sur Internet utilisent des chemins non connus et passent par des routeurs potentiellement écoutés. Le canal est donc souvent non sécurisé.

La **confidentialité** est l'impossibilité pour Eve de comprendre les messages circulant sur le canal, l'**intégrité** est la non-modification par Eve des messages (ou la détection d'une modification), l'**authentification** assure Bob que le message vient bien d'Alice et inversement, la **non répudiation** empêche Alice de nier avoir envoyé tel ou tel message et la **disponibilité** est la possibilité pour Bob et Alice de continuer à échanger.

1.1 - Confidentialité

La confidentialité permet de s'assurer que seules les personnes autorisées peuvent avoir accès à l'information transmise. Elle permet au concepteur d'un système d'information d'empêcher tout attaquant de récolter des informations sensibles en faisant de l'écoute clandestine (on parle d'*eavesdropping* en anglais).

La technique la plus courante pour garantir la confidentialité des échanges est l'utilisation de clés pour le **chiffrement** des données. Le chiffrement de données consiste en leur transformation en d'autres données, pas forcément de même longueur, grâce à un algorithme de chiffrement qui va utiliser une **clé** comme paramètre. Le terme clé est assez explicite : L'émetteur doit avoir une clé pour enfermer le message dans le coffre qui permet son transport et le destinataire doit aussi avoir une clé (la même ou une clé « associée ») pour ouvrir le coffre et lire le message.

Les algorithmes de chiffrement sont longs à développer et à vérifier. Les fuites survenant inévitablement, on ne peut envisager de redévelopper un algorithme de chiffrement à chaque fuite. C'est pourquoi le **Principe de Kerckhoffs** explique que l'algorithme doit être public et seules les clés doivent être secrètes. Celles-ci peuvent alors être renouvelées régulièrement.

On imagine ainsi qu'Alice souhaite envoyer un message à Bob. Ce message circulant sur un canal de communication accessible (ondes ou Internet par exemple), il peut être lu mais ne doit pas pouvoir être compris par Eve.

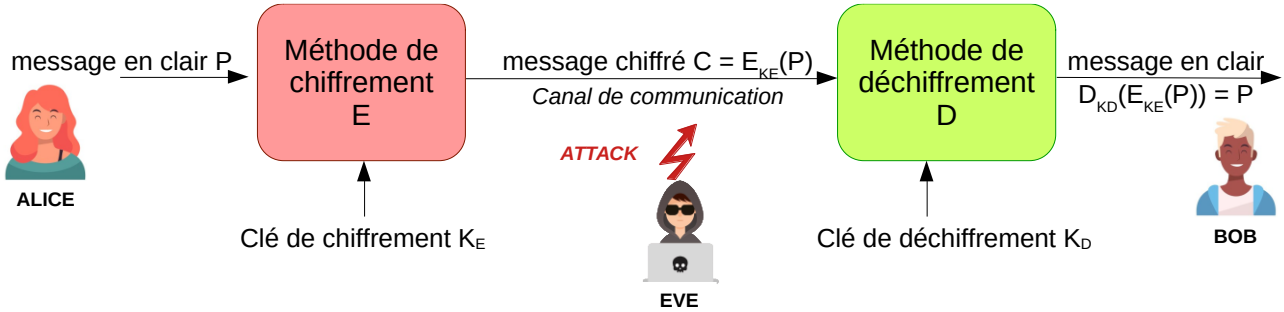


Figure 2 : Schéma de principe du chiffrement

Le développement des méthodes de chiffrement est la cryptographie, la recherche de méthode de déchiffrement sans connaître la clé est la cryptanalyse qui peut exploiter :

→ le texte chiffré seul,

→ un texte en clair connu (voir comme illustration le film *Imitation Game*),

→ un texte en clair choisi.

Chiffrement symétrique

Le chiffrement symétrique est le plus simple : l'émetteur et le récepteur **partagent le même secret**, la clé de chiffrement.

On peut imaginer par exemple deux individus qui discutent dans une langue qu'ils ont créée. La clé de chiffrement dans ce cas est le dictionnaire de cette langue. A priori, toute personne écoutant leur conversation et ne connaissant pas le dictionnaire de cette langue ne peut rien comprendre.

Un exemple classique de chiffrement symétrique est le **chiffrement par décalage** où l'on décale les lettres de l'alphabet d'un même nombre qui sera la clé de chiffrement. Par exemple, si l'on choisit un décalage de 5, `BONJOUR` sera chiffré en `GTSOTZW`. Pour déchiffrer le message, le récepteur doit seulement connaître la valeur du décalage et décaler les lettres dans le sens inverse. D'autres méthodes simples (OU exclusif avec un motif par exemple) consistent également à remplacer un caractère par un autre. On parle de **chiffrement par substitution**.

Il est aussi possible de modifier l'ordre des lettres, on parle alors de **chiffrement par permutation**.

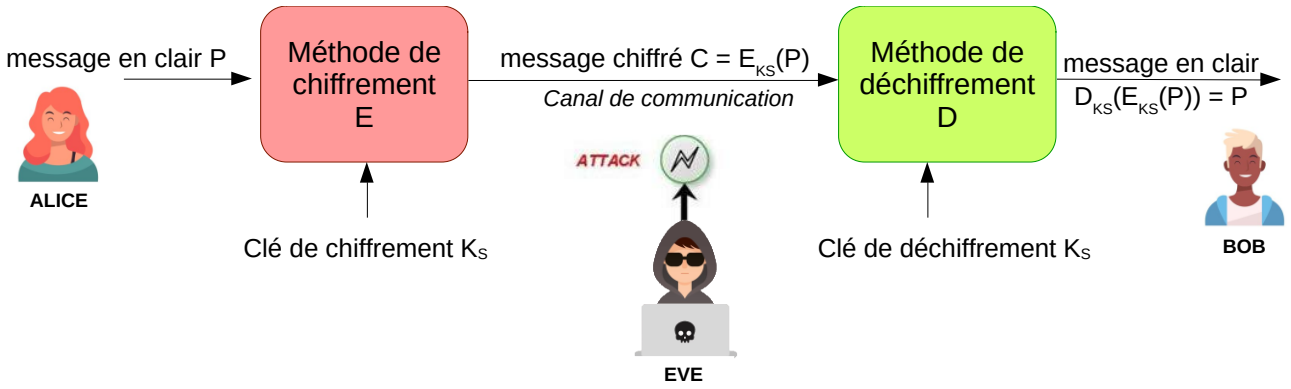


Figure 3 : Schéma de principe du chiffrement par clé symétrique

Un algorithme performant utilise à la fois permutation et substitution, pour éviter les méthodes de cryptanalyse statistique (pour du texte, on cherche le motif le plus courant et on lui associe la lettre la plus fréquente). Il utilise des clés suffisamment longues pour éviter les recherches de clés par force brute (tests successifs de toutes les clés possibles). L'algorithme de chiffrement symétrique le plus couramment utilisé (pour le wifi WPA2 ou pour HTTPS notamment) est l'algorithme **AES (Advanced Encryption Standard), avec des clés symétriques de 256 bits (soit 10^{77} clés possibles).**

> Les algorithmes de chiffrement symétriques sont plus rapides que les algorithmes de chiffrement asymétriques. Leur sécurité repose cependant sur le transfert de la clé secrète entre les deux appareils.

Chiffrement asymétrique

Le chiffrement asymétrique est plus complexe mais évite d'avoir à distribuer à l'avance une clé secrète aux participants : deux clés sont utilisées dans le processus de chiffrement : une **clé**

publique, connue de tous, et une **clé privée**, gardée secrète le plus souvent par le récepteur des données.

> La confidentialité des échanges est assurée lorsqu'on chiffre les données avec la clé publique : dans ce cas, seul le possesseur de la clé privée pourra déchiffrer le message.

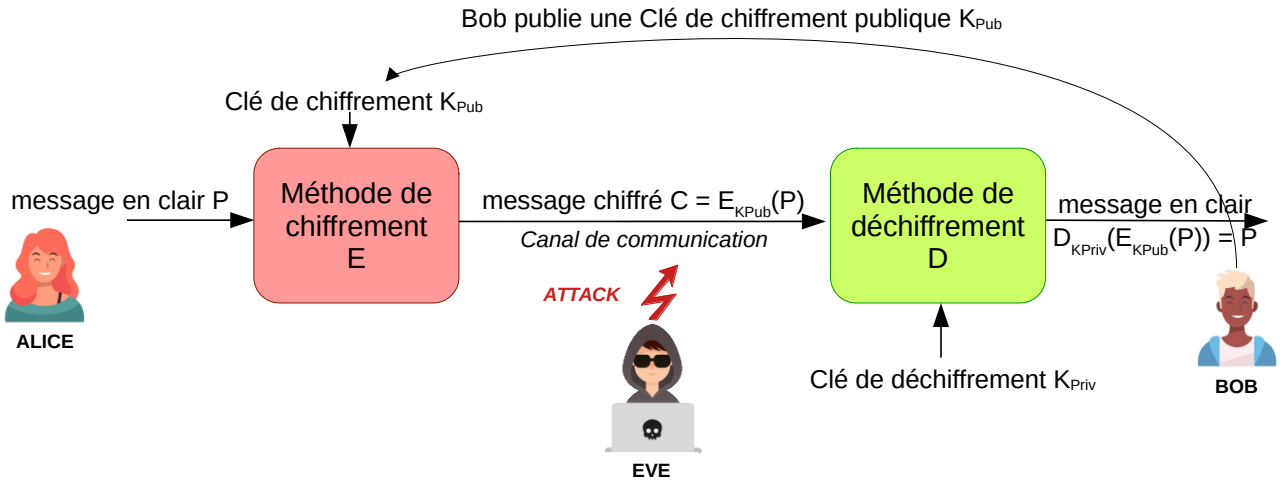


Figure 4 : Schéma de principe du chiffrement par clés asymétriques

Une analogie courante pour décrire le chiffrement asymétrique est celle de l'envoi d'un message de Bob vers Alice, en utilisant un coffre :

1° Alice dispose de cadenas identiques, s'ouvrant avec une clé et se refermant sans la clé (comme un cadenas classique). Elle envoie les cadenas ouverts à toute personne souhaitant lui envoyer un message secret, dont Bob. Elle garde la clé avec elle

2° Bob dispose d'un coffre et met son message à l'intérieur. Il prend le cadenas reçu d'Alice et le referme pour verrouiller le coffre.

3° Alice reçoie le coffre fermé avec son cadenas. Elle est la seule à pouvoir ouvrir ce cadenas.

L'algorithme de chiffrement asymétrique le plus souvent utilisé est l'algorithme **RSA**. Celui-ci est détaillé dans la ressource « Déchiffrez c'est gagné » [8], du même dossier.

Le protocole d'échange de clés **Diffie-Hellman** utilise aussi un algorithme asymétrique pour générer et partager de manière sécurisée une clé symétrique commune aux deux participants.

> Les algorithmes asymétriques sont les plus coûteux en ressources pour les calculs et en temps. Ils sont toutefois plus sécurisés que les algorithmes symétriques car aucune donnée secrète ne doit être partagée entre les deux protagonistes. On les utilise le plus souvent comme un moyen d'établir et de communiquer une clé symétrique à chaque nouvelle session (avec la méthode Diffie-Hellman par exemple).

On note que les clés publiques et privées de RSA sont réversibles, ce qui servira dans la partie suivante sur l'intégrité :

$D_{KPriv}(E_{KPub}(P)) = P$ mais aussi $D_{KPub}(E_{KPriv}(P)) = P$

1.2 - Intégrité

L'intégrité des données échangées désigne le fait qu'elles n'ont pas été modifiées durant le cycle de vie du message. Lors du téléchargement d'un logiciel par exemple, celui-ci n'est pas secret, il serait lourd de chiffrer l'ensemble du logiciel. Pour garantir l'intégrité du logiciel, l'éditeur met à disposition un code issu de l'ensemble des données de ce logiciel passé par une fonction de hachage cryptographique (**HMAC *Hash Message Authentication Code***).

Fonctions de hachage

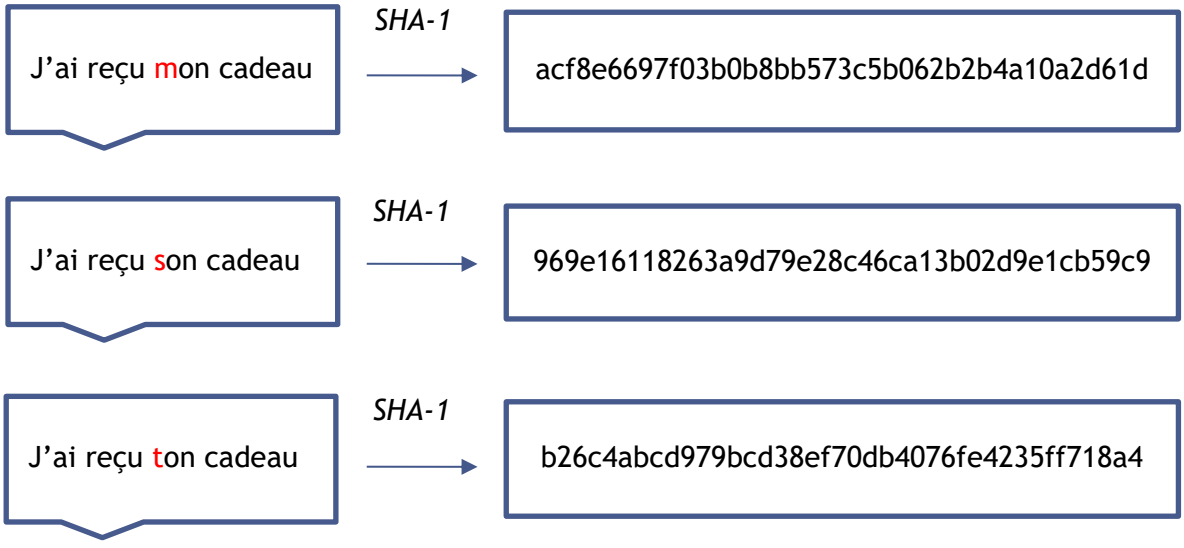
Les fonctions de hachage sont très souvent rencontrées en cybersécurité. Une fonction de hachage est une fonction qui, pour une donnée en entrée de longueur variable (qui peut être très longue, comme une vidéo ou un logiciel), retourne une valeur de longueur fixe (quelques octets), nommée condensat (ou en anglais *digest*).

Les codes de détection d'erreur CRC (Cyclic Redundancy Check) utilisés pour vérifier l'intégrité aux perturbations électromagnétiques des trames Ethernet ou CAN sont des fonctions de hachage, mais pas de cybersécurité.

Les fonctions de hachage classiques ne sont pas conçues pour répondre à toutes les problématiques de cybersécurité. C'est pourquoi on utilise des fonctions particulières, appelées **fonctions de hachages cryptographiques**. Les plus utilisées sont SHA-1 (*Secure Hash Algorithm*) et SHA-2, qui comprend notamment SHA-256.

Ces fonctions répondent aux exigences suivantes :

- Déterminisme : la même valeur de hachage (condensat) sera systématiquement générée pour un même message ;
- Rapidité de calcul ;
- **Non-réversibilité** : on ne peut pas reconstruire un message en ne connaissant que son condensat ;
- Résistance aux **collisions** : on ne peut pas trouver facilement un second message produisant le même condensat ;
- **Effet d'avalanche** : une légère modification du message entraîne une importante modification du condensat (voir exemple ci-dessous avec la fonction *SHA-1*).



Outre pour la signature numérique présentée ci-dessous, les fonctions de hachage sont utilisées pour le stockage des mots de passe. Les mots de passe ne sont normalement pas stockés en clair dans une application, pour éviter leur divulgation en cas de fuite. L'application se contente de stocker le condensat du mot de passe (mot de passe souvent concaténé avec une chaîne de caractère nommée « sel » pour améliorer la robustesse du stockage du condensat).

Signature numérique

On utilise ici la réversibilité des clés asymétriques. Si Alice chiffre un document avec sa clé privée K_{PRIVA}, qu'elle seule connaît, toute personne ayant la clé publique d'Alice peut déchiffrer le message et être sûr qu'il provient d'Alice.

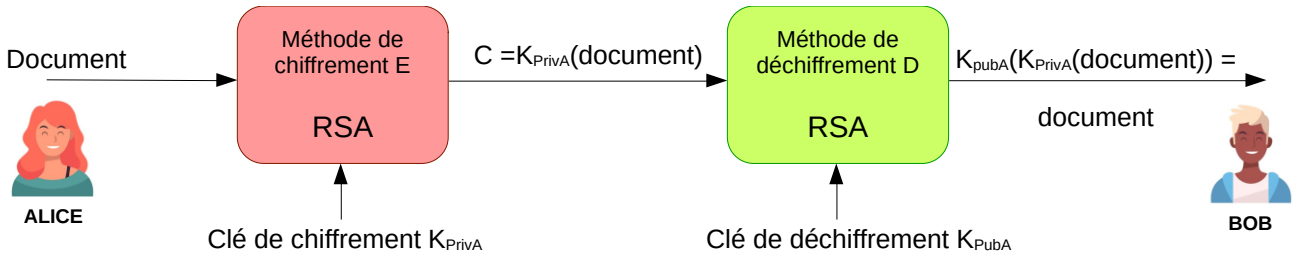


Figure 5 : Schéma de principe de la signature numérique

Cependant, chiffrer un gros fichier avec une clé asymétrique est gourmand en temps et en énergie. Il est plus économe de le hacher et de ne chiffrer avec la clé privée que le condensat.

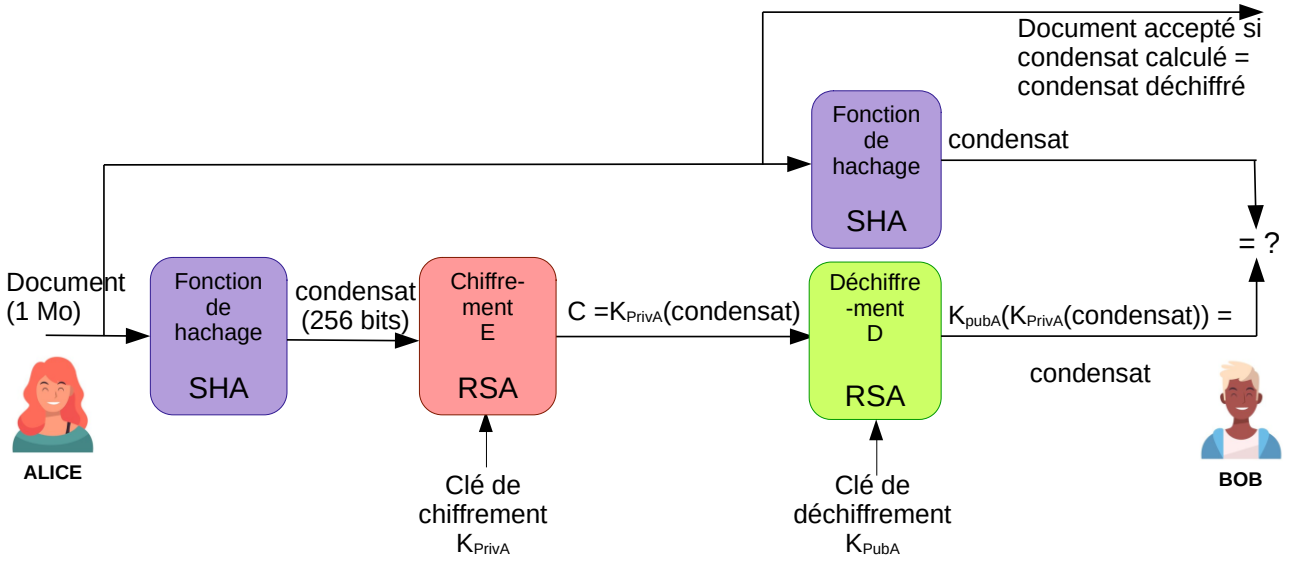


Figure 6 : Schéma de principe de la signature numérique avec fonction de hachage

Bob a ainsi l'assurance que le document reçu n'a pas été modifié entre son émission par Alice et sa réception.

1.3 - Non-répudiation

La non-répudiation assure qu'une action réalisée par une entité ne peut être niée ou ignorée. Elle apporte donc des preuves irréfutables qui pourront être utilisées a posteriori.

L'utilisation de **signatures numériques** peut assurer la non-répudiation. En chiffrant les messages avec sa clé privée (voir Figure 6), l'entité ne peut pas nier avoir signé le message car elle est la seule à posséder cette clé privée. De plus, certains mécanismes de signature numérique utilisent l'horodatage dans les données générant la signature numérique. Ainsi, même si la clé privée est volée par la suite, on peut s'assurer que la signature est bien valide car générée avant le vol.

1.4 - Authentification

L'authentification est le processus de vérification de l'identité d'une entité (utilisateur, système, appareil, etc.). Elle permet de s'assurer que, dans la communication, chacun est réellement qui il prétend être.

Les mécanismes d'authentification sont nombreux. On peut séparer d'une part les authentifications d'utilisateurs pré-enregistrés et les authentifications par **certificats**.

La première famille regroupe l'authentification par nom d'utilisateur et mot de passe ou l'authentification biométrique (empreinte digitale, reconnaissance faciale, etc.). Elle demande à l'un des partenaires de l'échange de connaître les identifiants de l'autre partenaire. Ce ne peut être une solution pour des échanges entre deux protagonistes ne se connaissant pas.

> Pour renforcer l'authentification, il est fréquent de mettre en place une authentification à plusieurs facteurs (deux voire trois). L'authentification à deux facteurs peut par exemple demander en plus d'un mot de passe la saisie d'une valeur envoyée par SMS.

Pour la seconde famille, on s'intéresse maintenant à deux acteurs ne se connaissant pas à l'avance, pour illustrer l'intérêt des certificats dans l'authentification.

La simple déclaration n'est évidemment pas suffisante, Eve pouvant déclarer être Alice.

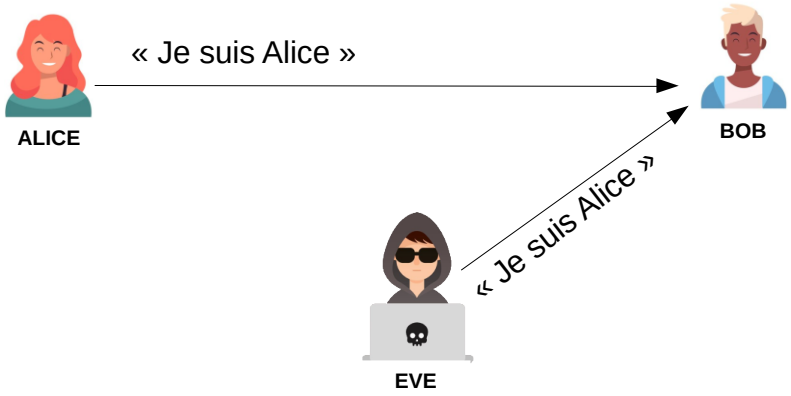


Figure 7 : Usurpation d'identité

Une déclaration chiffrée (ce qui demanderait d'avoir une clé symétrique) peut être rejouée par Eve pour se faire passer pour Alice.

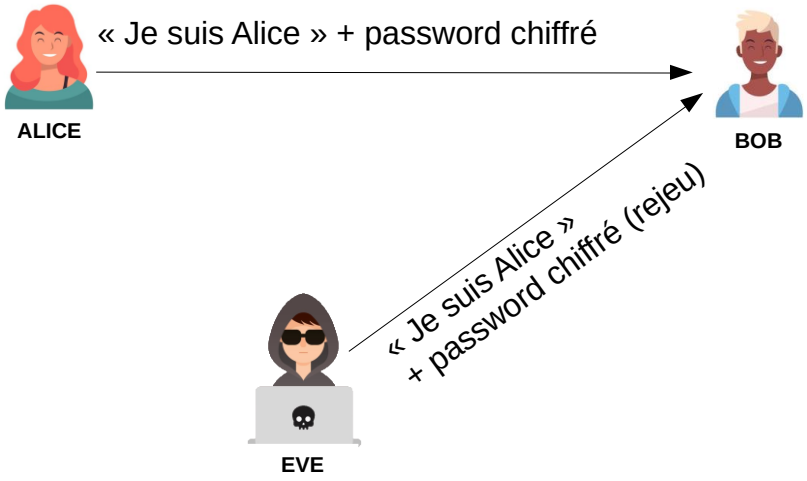


Figure 8 : Usurpation d'identité chiffrée par rejeu

Pour éviter le rejeu, Bob envoie un nombre R à usage unique. Comme les acteurs n'ont pas partagé au préalable une clé secrète, un mécanisme à clés asymétriques est utilisé. K_{PUBA} et K_{PRIVA} sont les clés publiques et privées d'Alice. Alice renvoie R chiffré avec sa clé privée et diffuse sa clé publique. On joue sur la réversibilité des clés publiques et privées. En connaissant la clé publique d'Alice, Bob (et toute autre personne voyant passer le message) peut déchiffrer le message $K_{PrivA}(R)$ et donc être sûr qu'il provient d'Alice.

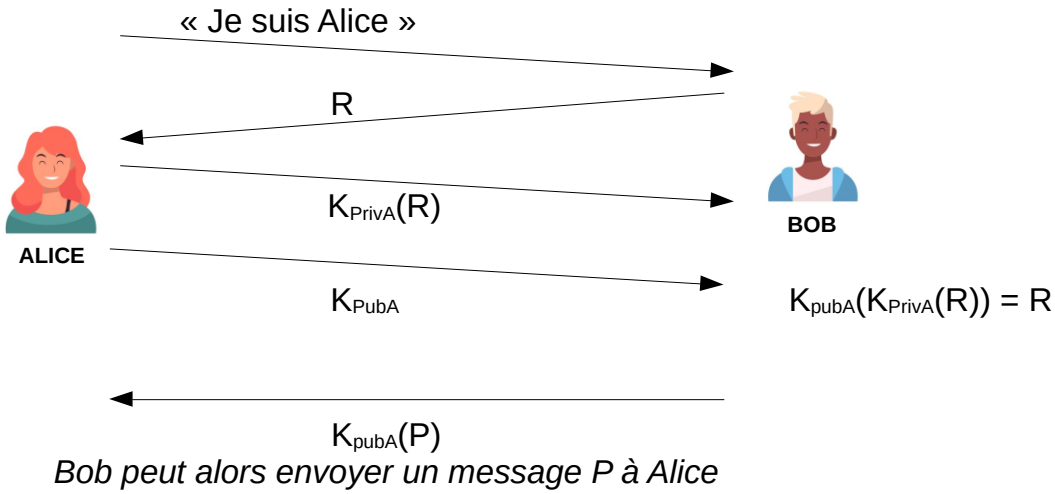


Figure 9 : Authentification par signature d'un nombre R

Reste à être sûr que K_{pubA} est bien la clé publique d'Alice. En effet, il est possible d'imaginer qu'Eve s'interpose entre Bob et Alice et intercepte les communications de l'un comme de l'autre. On appelle cette attaque « Man in the Middle ». Eve se fait passer pour Alice auprès de Bob et pour Bob auprès d'Alice. Elle peut alors lire et/ou modifier le message P envoyé par Bob à Alice.

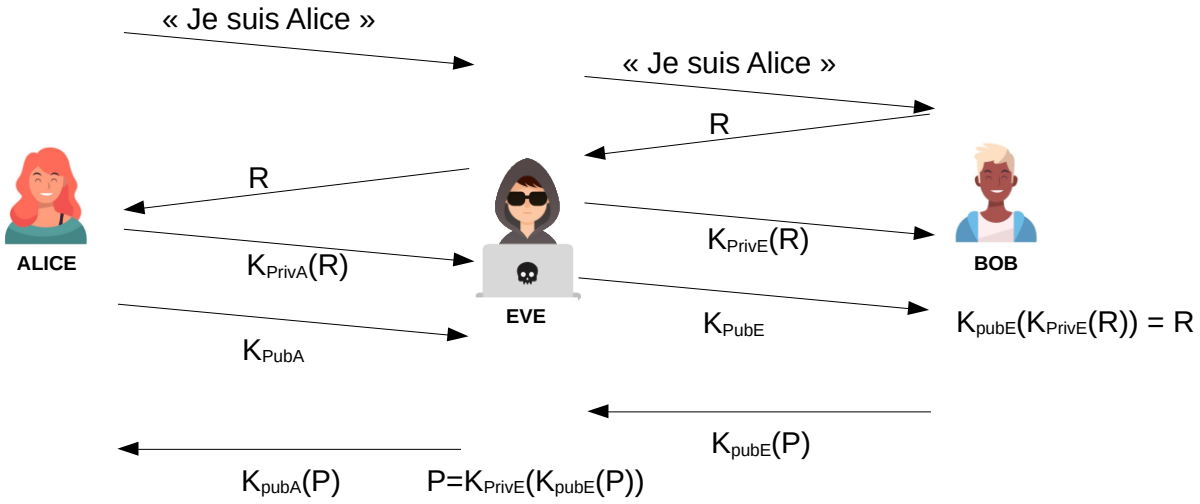


Figure 10 : Principe de l'attaque Woman in the middle

Il est donc nécessaire de certifier les clés publiques, ce que permettent les certificats. X509 est le protocole de gestion de certificats le plus populaire. Un tiers de confiance nommé autorité de certification signe numériquement (c'est-à-dire chiffre avec sa clé privée un condensat) le certificat contenant notamment la clé publique et le nom du protagoniste. La clé publique de cette autorité de certification est connue (les navigateurs web ont par exemple les clés publiques des principales autorités de certification) ou alors elle est diffusée elle-même avec un certificat d'une autorité de certification connue.

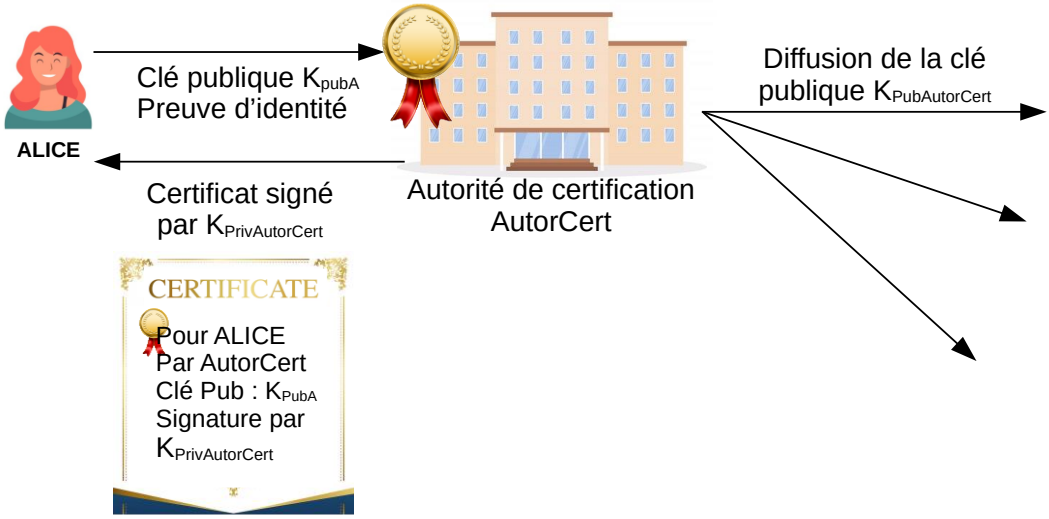


Figure 11 : Obtention d'un certificat auprès d'une autorité de certification

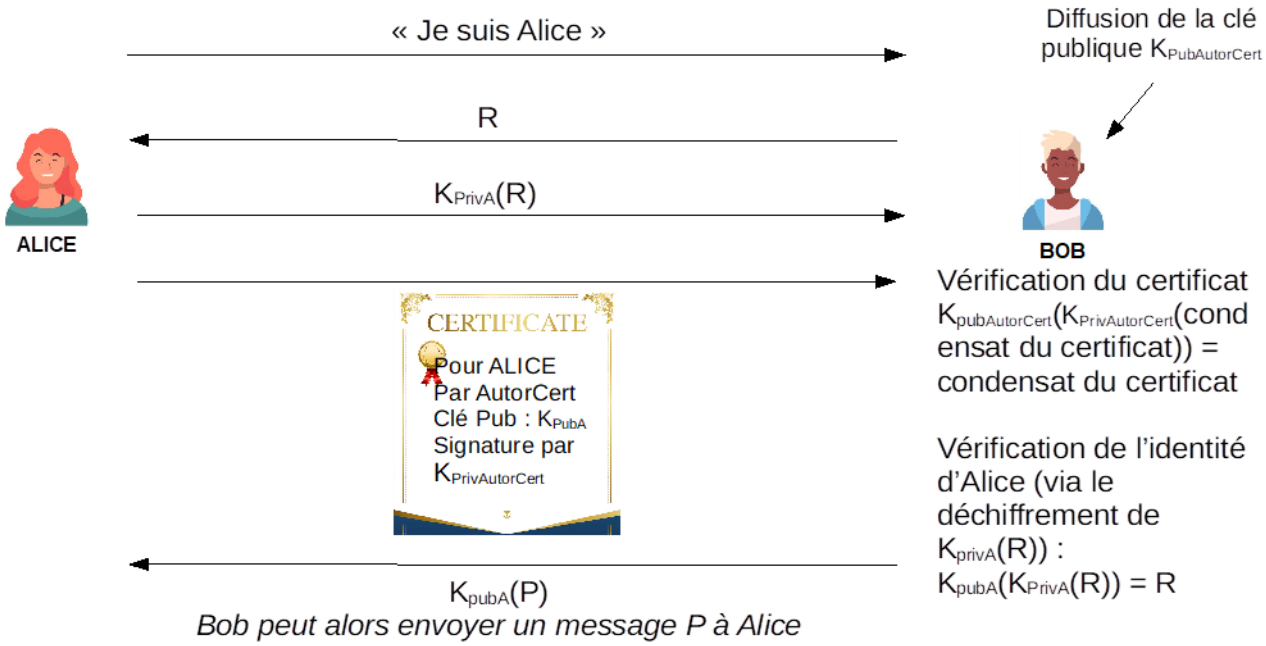


Figure 12 : Authentification avec utilisation d'un certificat

On résout ainsi le risque d'usurpation d'identité par Eve. On retrouve ces concepts en vidéo sur la chaîne d'Hervé Discours [7].

Figure 13 : Copie d'écran de la vidéo d'Hervé Discours sur les clés et certificats

1.5 - Disponibilité

La disponibilité désigne la capacité d'un service, d'un système à être accessible par un utilisateur autorisé quand il le souhaite.

> La disponibilité d'un système d'information est l'un des piliers de son bon fonctionnement. En effet, une attaque rendant impossible l'accès à un service en ligne fourni par une société nuit grandement à l'image de cette dernière.

Les attaques par déni de service distribué (DDoS, Distributed Denial of Service) sont particulièrement difficiles à bloquer car elles n'utilisent que des requêtes légales, issues d'appareils détournés, pour saturer un serveur. Les dernières attaques DDoS atteignent des débits de plusieurs $Tb.s^{-1}$ (2,3 $Tb.s^{-1}$ pour l'attaque d'AWS en février 2020)

Les requêtes étant légales, les mécanismes permettant de garantir la disponibilité sont complexes mais très étudiées au vu des enjeux financiers ou organisationnels liés à la coupure de service réseau :

- **Redondance** des systèmes ;
- Répartition de la charge de travail sur plusieurs serveurs pour éviter la surcharge ;
- Filtrage en amont des requêtes avec des pare-feux avancés.

2 - Exemple du protocole TLS

Le protocole TLS (*Transport Layer Security*, sécurité de la couche de transport) est un protocole cryptographique permettant de sécuriser les communications sur un réseau informatique (ici confidentialité via une clé symétrique AES-128, intégrité via une fonction de hachage SHA-256 et authentification via un certificat X509 et des échanges par clés asymétriques Diffie Hellman), au-dessus de la couche transport TCP et en-dessous de la couche application. Son utilisation la plus connue est le protocole HTTPS qui consiste en une version sécurisée du protocole HTTP. Il est aussi utilisé pour des protocoles industriels (OPC UA, IEC61850) et dans l'automobile (IEC15118).

> On utilise parfois le terme « SSL/TLS ». SSL désigne l'ancêtre du protocole TLS et toutes ses versions sont obsolètes depuis 2015. Il convient donc d'utiliser uniquement le terme TLS dans la conception d'un système d'information et de ne pas utiliser abusivement le terme « SSL/TLS ».

Cette partie explique les principes de fonctionnement de ce protocole afin d'illustrer l'implémentation des principes fondamentaux exposés précédemment. La version 1.3 étant un peu plus simple que la 1.2, elle sert de support à cette explication. Les principes sont les mêmes, avec un peu plus d'échanges pour la version 1.2, encore utilisée.

2.1 - Négociation de la connexion sécurisée

Dans un premier temps, le client demande au serveur de s'authentifier, les deux se mettent d'accord sur les paramètres de la connexion sécurisée et, via des clés asymétriques, échangent les clés symétriques de session.

Premier contact par le client - *Client Hello*

Dans ce premier message, le client communique entre autres au serveur les versions de TLS qu'il prend en charge, l'ensemble des algorithmes cryptographiques qu'il peut utiliser par ordre de préférence et un nombre aléatoire qui servira pour générer les clés de cette session. Pour reprendre une session interrompue, il est possible de fournir un identifiant de session, *Session ID*.

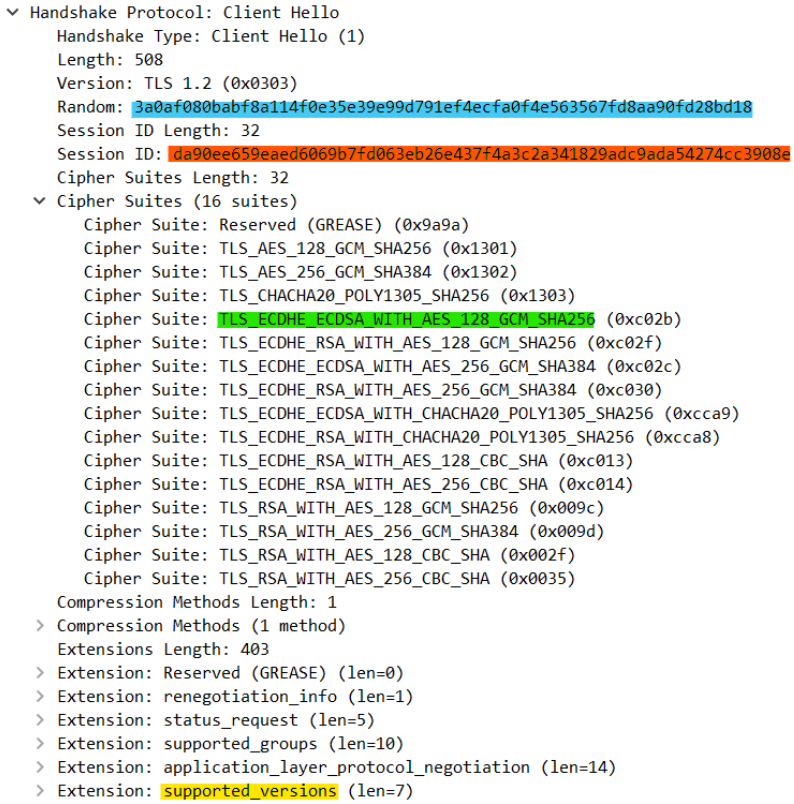


Figure 14 : Extrait d'un ClientHello sur Wireshark

Il est intéressant d'analyser les types d'algorithmes cryptographiques utilisés :

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

Elliptic Curve Diffie-Hellman Ephemeral

Algorithme asymétrique qui permet la génération et l'échange d'une clé symétrique utilisée pour l'établissement des clés de session

Elliptic Curve Digital Signature Algorithm

Algorithme asymétrique qui permet l'authentification, la non-répudiation et l'intégrité.

AES 128 bits Galois/Counter Mode

Algorithme de chiffrement symétrique qui permet la confidentialité des communications

Secure Hash Algorithm 256 bits

Fonction de hachage qui permet de générer un condensat pour vérifier l'intégrité des données échangées

Réponse du serveur - *Server Hello*

Le serveur répond alors au client en sélectionnant pour chaque paramètre une valeur parmi celles proposées par le client et qui convient à ses capacités.

Le serveur envoie également un nombre aléatoire qui servira pour générer les clés de cette session

Envoi et vérification du certificat du serveur

Le serveur fournit ensuite au client un certificat électronique, aussi appelé certificat numérique. Il va permettre au client de faire confiance à l'entité avec laquelle il communique. Il s'agit donc d'une étape essentielle pour garantir la sécurité de la connexion qui est sur le point d'être établie.

On peut considérer ce certificat comme la carte d'identité du serveur. On observe sur la Figure 15 le certificat numérique du site de l'ENS Paris-Saclay (accessible en cliquant sur le cadenas dans la barre d'adresse).

Figure 15 : lecture du certificat numérique du site de l'ENS Paris-Saclay sur un navigateur Google Chrome

Le certificat numérique est émis par une autorité de certification. Sur la Figure 15, le certificat a été fourni par l'organisation GEANT. Cette organisation fournit, moyennant rétribution, le certificat pour une période donnée, cette période est d'un an pour celui de la Figure 15.

> Il existe de nombreuses autorités de certifications qui opèrent à différentes échelles. Cela permet de ne compromettre que les serveurs situés dans une zone précise si une autorité de certification n'est plus sûre. Ainsi, les autorités de certification intermédiaires doivent aussi présenter un certificat numérique et le client doit vérifier tous les certificats numériques jusqu'à remonter à une autorité de certification de confiance.

Le client va donc devoir :

- Vérifier l'intégrité du message avec le condensat (condensant nommé « Empreinte SHA » sur la Figure 15) ;
- Vérifier l'authentification grâce à la signature du certificat par la clé privée de l'autorité de certification ;
- S'assurer qu'il peut faire confiance à l'autorité de certification en remontant la chaîne des certificats numériques, c'est-à-dire en vérifiant les certificats des autorités de certification jusqu'à arriver sur une autorité de certification de référence, connue du logiciel (le navigateur pour HTTPS par exemple) (voir Figure 16).

Figure 16 : Chaîne de vérification des certificats pour le site de l'ENS Paris-Saclay

Calcul du secret partagé

L'algorithme de Diffie Hellman, un peu différent de RSA, permet d'établir une clé commune à partir d'éléments de clés publiques envoyés par les 2 acteurs. Chacun peut alors déchiffrer à l'aide de sa clé privé les échanges.

Le client connaît désormais via le certificat les éléments de clés publiques du serveur et a transmis les siens lors du premier échange. Indépendamment, le client et le serveur vont donc pouvoir calculer à partir de ces éléments de clés publiques un nouveau secret commun, le *master secret*.

Calcul des clés de session

Le client peut alors communiquer de manière chiffrée avec le serveur, par le *master secret*. Il peut ainsi envoyer des clés symétriques AES au serveur, de manière chiffrée. 4 clés sont partagées : 1 pour les données client → serveur, 1 pour la vérification de l'intégrité client → serveur et la même chose dans le sens serveur → client.

Les différentes données utilisées pour établir la sécurité de l'application (valeurs aléatoires, *pre-master secret*, *master secret*, etc.) ne seront pas réutilisées lors de futures sessions entre les mêmes client et serveur. Cela participe au respect d'un principe appelé « confidentialité persistante » (*Forward Secrecy* en anglais) qui garantit que la découverte d'un secret privé ne compromet pas la confidentialité des échanges passées.

2.2 - Établissement de la connexion sécurisée

Une fois les clés de session établies, le client et le serveur vont chacun envoyer un message permettant de vérifier qu'ils possèdent bien les mêmes clés et signalant que les échanges futurs seront chiffrés.

Les échanges sont à présent sécurisés.

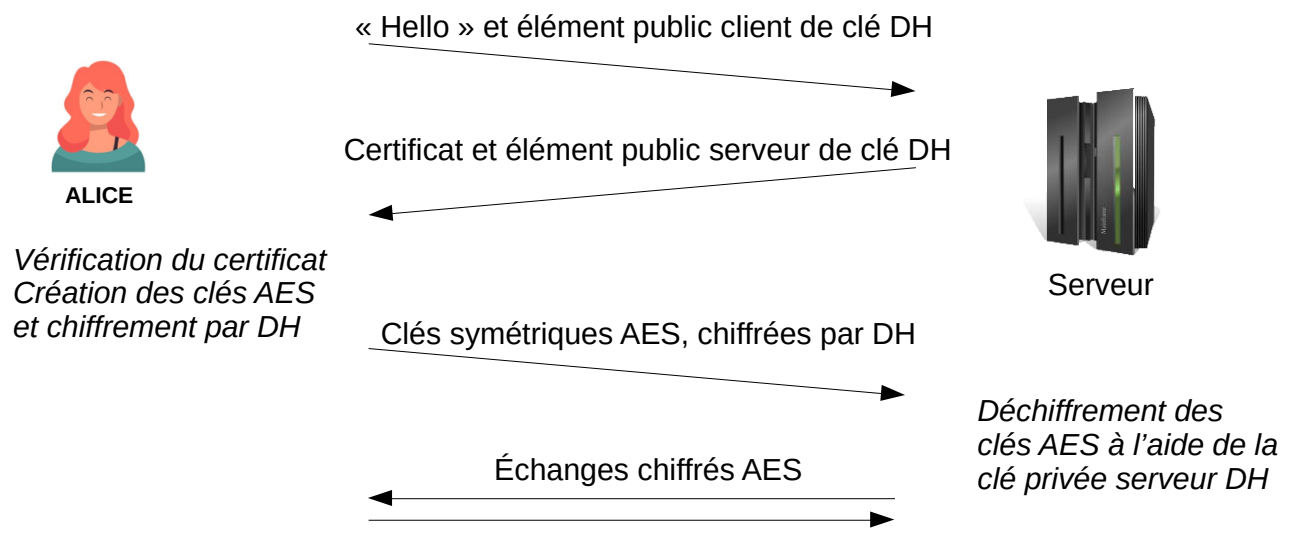


Figure 17 : Echanges TLS 1.3 menant à la mise en place d'une communication sécurisée

Les acquisitions ci-dessous présentent une acquisition wireshark de la mise en place d'un échange TLS 1.2 et TLS 1.3. Seuls les messages TLS sont affichés, les acquittements TCP notamment n'apparaissent pas, pour plus de lisibilité. Application Data signifie que les échanges chiffrés sont commencés.

Source	Destination	Protocol	Length	Info
192.168.1.41	129.175.212.146	TLSv1.2	744	Client Hello
129.175.212.146	192.168.1.41	TLSv1.2	1434	Server Hello
129.175.212.146	192.168.1.41	TLSv1.2	1430	Certificate
129.175.212.146	192.168.1.41	TLSv1.2	354	Server Key Exchange, Server Hello Done
192.168.1.41	129.175.212.146	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
129.175.212.146	192.168.1.41	TLSv1.2	117	Change Cipher Spec, Encrypted Handshake Message
192.168.1.41	129.175.212.146	TLSv1.2	1930	Application Data
129.175.212.146	192.168.1.41	TLSv1.2	685	Application Data

192.168.1.41	138.231.176.51	TLSv1.3	721	Client Hello
138.231.176.51	192.168.1.41	TLSv1.3	4150	Server Hello, Change Cipher Spec, Application Data
138.231.176.51	192.168.1.41	TLSv1.3	2739	Application Data, Application Data, Application Data
192.168.1.41	138.231.176.51	TLSv1.3	134	Change Cipher Spec, Application Data
192.168.1.41	138.231.176.51	TLSv1.3	679	Application Data
138.231.176.51	192.168.1.41	TLSv1.3	325	Application Data

Figure 18: Comparaison entre l'établissement d'une connexion TLS 1.2 et celui d'une connexion TLS 1.3

3 - Conclusion

Cette ressource a expliqué les principes fondamentaux de la sécurité réseau que l'on retrouvera dans les autres ressources de ce dossier avant de voir comment ces principes sont mis en œuvre par le protocole TLS.

La ressource associée « Informatique débranchée : Déchiffrez c'est gagné » [8] donne un exemple de mise en pratique des algorithmes de chiffrement par les étudiants.

Les principes présentés ici sont issus de l'informatique et appliqués en automatisme industriel (OPC UA, IEC 61850, …) ou dans l'automobile (IEC 15118). Dans les objets connectés, aux capacités de calcul et à la consommation plus réduites, il est important d'appliquer les mêmes principes, avec des algorithmes peu gourmands. Cela passe notamment par l'intégration hardware de blocs de chiffrement, ce qui fige les algorithmes utilisés.

Par ailleurs, la puissance des ordinateurs augmentant, la taille des clés croît également pour résister aux attaques par force brute, passant par exemple de 1024 à 2048 bits pour RSA. Cela pose la question de la mise à jour logicielle, voire matérielle, des systèmes industriels à la longue durée de vie.

Enfin, l'arrivée des premiers ordinateurs quantiques, avec des capacités estimées très importantes pour craquer les clés asymétriques, amène dès aujourd'hui à réfléchir à de nouveaux algorithmes de chiffrement, plus robustes face à ces nouvelles machines. Que restera-t-il alors de la sécurité réseau des systèmes industriels installés en 2024 ?...

4 - Annexe : Quelques références supplémentaires

Cette annexe propose une sélection de vidéos YouTube, en français ou en anglais, permettant d'illustrer ou d'approfondir les différentes notions expliquées dans cette ressource.

Cette sélection restreinte parmi les nombreuses vidéos sur le sujet permet aussi de découvrir quelques chaînes YouTube dédiées à la sécurité réseau. Le lecteur intéressé par la pédagogie ou le niveau technique d'une des vidéos pourra explorer les autres vidéos de la chaîne.

Ces vidéos peuvent être utilisées notamment pour réaliser des séquences pédagogiques sous forme de classe inversée ou comme supports pour le co-enseignement en anglais en BTS CIEL.

4.1 - Exemples d'attaques liées à des problématiques de cybersécurité

La chaine **Cyber Vox** regroupe des vidéos, en français et en anglais, sur des attaques historiques telles que Wannacry, notPetya, stuxnet solarWinds : https://www.youtube.com/@CyberVox

4.2 - Principes fondamentaux : Vidéos de vulgarisation

4.2.1 - Confidentialité

Chiffrement symétrique / Chiffrement asymétrique

Cette vidéo de vulgarisation sur le chiffrement symétrique et asymétrique, issue de la chaîne en anglais Code.org, s'appuie sur des animations très pédagogiques :

The Internet : Encryption & Public Keys : https://www.youtube.com/watch?v=ZghMPWGXexs

La chaine Exo7Math apporte les notions mathématiques pour comprendre le protocole RSA du point de vue algorithmique :

Cryptographie - partie 5 : arithmétique pour RSA : https://youtu.be/M7vOxKVLsVY

Cryptographie - partie 6 : chiffrement RSA : https://www.youtube.com/watch?v=Xlal_d4zyfo

4.2.2 - Intégrité

Fonctions de hachage

Cette vidéo, issue de la chaîne **Bande de Codeurs**, détaille les fonctions de hachage et leurs différentes utilisations :

Comprendre les fonctions de HACHAGE : https://www.youtube.com/watch?v=OHXfKCH0b6s

4.2.3. - Disponibilité

Cette vidéo issue de la chaîne **@IBM technology** aborde la résistance aux ransomware :

Protecting Yourself from Ransomware : https://www.youtube.com/watch?v=eizn9TC68E8

4.2.4 - Authentification

Cette vidéo de la chaîne **kubucation,** en anglais, approfondit les notions de certificat et d'autorité de certification appliquées à HTTPS :

How does HTTPS work? What's a CA? What's a self-signed Certificate? :
https://www.youtube.com/watch?v=T4Df5_cojAs

4.2.5 - Non-répudiation

La chaîne **LeDroitpourMoi** propose une vidéo s'intéressant au côté juridique de la signature électronique :

Signature électronique : comment ça marche ? https://www.youtube.com/watch?v=GhTZUbp9M-8

4.3 - Sujets divers

La chaîne L'informateur aborde en français tous les sujets de la sécurité réseau, notamment les réseaux VPN.

Sécurité 13 : IPSec et comment fonctionne un VPN :
https://www.youtube.com/watch?v=V9bTy0gbXIQ&list=PLOapGKeH_KhFBC39ltMDhkEx1aI3hlwSK

La chaîne **@IBM technology** présente, en anglais, le concept Zero Trust :
Why Implement Zero Trust : https://www.youtube.com/watch?v=lT11tGaEC3s

Références

[1]: *Fiches incidents cyber SI industriels – Fiche 22*, Clusif, 2022
https://clusif.fr/publications/fiches-incidents-cyber-si-industriels/

[2]: *Hackers Remotely Kill a Jeep on the Highway – With Me in It*, Andy Greenberg, WIRED, 2015
https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

[3]: *The Fresh Smell of ransomed coffee*, Martin Hron, Avast, 2020
https://decoded.avast.io/martinhron/the-fresh-smell-of-ransomed-coffee/

[4]: *Essonne : un centre hospitalier visé par une cyberattaque, une rançon de 10 millions de dollars exigée*, Le Figaro, 2022
https://www.lefigaro.fr/secteur/high-tech/essonne-un-centre-hospitalier-vise-par-une-cyberattaque-une-rancon-de-10-millions-de-dollars-exigee-20220822

[5]: Images issues de Freepik : https://fr.freepik.com/vecteurs-libre/avatars-gens-heureux_7085154.htm et https://fr.freepik.com/vecteurs-libre/jeune-pirate-anonyme-au-design-plat_2753360.htm et https://fr.freepik.com/vecteurs-libre/ensemble-batiments-ville_8270967.htm

[6]: Computer Networks, Andrew Tanenbaum, Nick Feamster, David Wetherall, Pearson Education Limited

[7]: Cybersécurité, Clés publique/privée, Certificat X509, PKI, Chiffrement RSA, fichier.pem OPC-UA, Hervé Discours, https://www.youtube.com/watch?v=58FUQzWxs3Y

[8]: Informatique débranchée : Déchiffrez c'est gagné, A. Juton, février 2024, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/informatique-debranchee-dechiffrez-cest-gagne

[9]: Computer Networking, a top-down approach, Jim Kurose, Keith Ross, Pearson; 8th edition (September 13, 2020), http://gaia.cs.umass.edu/kurose_ross

Ressource publiée sur Culture Sciences de l'Ingénieur : https://eduscol.education.fr/sti/si-ens-paris-saclay