Sécurisation des communications

I./ Nécessité de la sécurisation des communications réseaux :

Les premiers protocoles ayant été utilisés sur Internet, dont TCP/IP, FTP, HTTP, SMTP, etc… ne sont pas sécurisés, dans le sens où les données circulent en clair (donc non chiffrées) et qu'expéditeur et destinataire ne sont pas authentifiés. Cela pose deux problèmes de sécurité :

- Un pirate éventuel qui intercepterait les signaux émis d'un ordinateur A à un ordinateur B peut directement lire en clair les informations transmises.
- Un pirate peut se faire passer pour l'expéditeur d'un message, ou pour son destinataire, pour récupérer des données qui ne lui sont pas destinées.

Plusieurs techniques permettent de réduire ces risques, notamment l'utilisation de protocoles sécurisés (comme FTPS, HTTPS, SMTPS, etc…). Afin d'obtenir la version sécurisée de ces protocoles, on leur ajoute une couche de sécurité (d'où le S final dans l'acronyme), comme SSL (Secure Sockets Layer) ou TLS (Transport Layer Security). On parle parfois de couche SSL/TLS.

Cette couche de sécurité assure la confidentialité des données échangées (grâce à des techniques de chiffrement) ainsi que l'authentification du serveur ou du client. Nous nous intéresserons seulement au chiffrement des données. Pour l'authentification, vous trouverez quelques informations ici : https://fr.wikipedia.org/wiki/Certificat_%C3%A9lectronique

II./ Techniques de chiffrement :

1./ Chiffrement symétrique :

Pour chiffrer un message M (qui peut être considérée comme étant une suite de caractères), on utilise une suite de caractère appelée clef de chiffrement K. Le message et la clef peuvent donc être vus comme des chaînes de caractères. Mais on peut aussi les représenter par des grands nombres binaires (cf. Cours de NSI Première, représentation des données).

Dans un algorithme de chiffrement symétrique, c'est la même clef K qui permet de chiffrer et de déchiffrer le message.

Soit un individu A qui veut envoyer un message M à un individu B à l'aide d'un algorithme de chiffrement symétrique. A et B doivent tout d'abord se mettre d'accord sur une clef de chiffrement K. Une fois le choix de la clef fait, A chiffre le message M avec la clef K et obtient le message chiffré M'. On a : M' = K(M). Il transmet ensuite le message chiffré M' à B. B déchiffre le message chiffré M' à l'aide de la clef K. On a : M = K(M').

Il existe plusieurs algorithmes de chiffrements symétriques : nous en avons vu plusieurs lors des révisions sur le langage Python, comme le chiffre de César ou le chiffre de Vigenère. Un autre exemple est la fonction XOR (ou OU EXCLUSIF), vue l'année dernière (cf. Cours de NSI Première, variables booléennes). On rappelle la table de vérité de la fonction XOR :

E1	E2	S
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Soit le message M suivant : « **`Bonjour !`** ». Pour des raisons de simplicités, on se limitera au codage ASCII. La représentation binaire du message M est :

`01000010 01101111 01101110 01101010 01101111 01110101 01110010 00100000 00100001`

Soit la clef de chiffrement K suivante : « **`Clef`** ». La représentation binaire de la clef K est :

`01000011 01101100 01100101 01100110`

La clef K étant plus courte que le message M, on la répète autant de fois qu'il faut pour avoir le même nombre de caractère que le message M.

Message M en clair								
B	**o**	**n**	**j**	**o**	**u**	**r**		**!**
01000010	01101111	01101110	01101010	01101111	01110101	01110010	00100000	00100001
Clef de chiffrement K								
C	**l**	**e**	**f**	**C**	**l**	**e**	**f**	**C**
01000011	01101100	01100101	01100110	01000011	01101100	01100101	01100110	01000011
Message chiffré M' : résultat de l'opération M XOR K bit à bit								
00000001	00000011	00001011	00001100	00101100	00011001	00010111	01000110	01100010
				,			**F**	**b**

La représentation binaire du message chiffré M' est :

00000001 00000011 00001011 00001100 00101100 00011001 00010111 01000110 01100010

Ce qui donne la chaîne de caractère suivante : « **, Fb** ». (Les caractères correspondent à des caractères de contrôle non affichables, correspondant aux codes ASCII inférieurs à 32).

Vérifiez que l'opération M' XOR K bit à bit redonne bien le message M :

……

……

……

……

……

……

……

……

Pour vous aider à traduire une chaîne de caractère ASCII en binaire, et pour faire l'opération inverse, vous pouvez vous aider des traducteurs en ligne suivants :

Traduction ASCII vers binaire : https://www.rapidtables.com/convert/number/ascii-to-binary.html

Traduction binaire vers ASCII : https://www.rapidtables.com/convert/number/binary-to-ascii.html

Programmation Python : implémenter en Python l'algorithme de chiffrement / déchiffrement par l'opération XOR. Pour réaliser l'opération XOR bit à bit entre deux nombres **a** et **b**, il faut utiliser l'opérateur : **a ^ b**.

En l'absence de la clef K, un espion éventuel n'obtiendra que le message chiffré M', soit une suite de caractères n'ayant aucun sens. Si on connaît la technique utilisé pour le chiffrement, ainsi que certaines informations sur la clef, on peut éventuellement retrouver le message originel M sans le passer par le déchiffrement. On appelle cela « décrypter » le message. Cela peut être très difficile selon la méthode de chiffrement, et la longueur de la clef.

Le principal défaut des algorithmes de chiffrement symétriques est qu'il faut que A transmette la clef K à B d'une façon ou d'une autre, que ce soit par courrier, par téléphone, ou par voie électronique (mais non chiffrée, car A et B n'ont justement pas encore partagé la clef !). Si un pirate intercepte la clef lors de son envoi, il pourra déchiffrer le message. Pour éviter cela, on peut utiliser des algorithmes de chiffrement asymétriques.

L'avantage des algorithmes de chiffrement symétriques sont qu'ils sont généralement plus rapides à utiliser que leurs homologues asymétriques.

2./ Chiffrement asymétrique :

Dans un algorithme de chiffrement asymétrique, chaque utilisateur possède deux clefs : une clef publique Kpu (qu'il peut diffuser à tout le monde), et une clef privée Kpr (qu'il faut qu'il garde secrète). L'individu A aura donc une clef publique KpuA et une clef privée KprA. De même, l'individu B aura une clef publique KpuB et une clef privée KprB. Avec un système de chiffrement asymétrique, ce n'est pas la même clef qui sert à chiffrer et à déchiffrer le message.

Si A veut envoyer un message M à B, il chiffre le message M à l'aide de la clef publique KpuB de B. Il obtient le message chiffré M'. On a : M' = KpuB(M). Le message chiffré M' est ensuite transmis à B qui le déchiffre grâce à sa clef privée. On a : M = KprB(M'). B étant normalement le seul individu à connaître sa clef privée KprB, il est le seul à pouvoir déchiffrer le message M'.

Lorsque B voudra envoyer sa réponse à A, il la chiffrera avec la clef publique KpuA de A. Après réception de cette réponse chiffrée, A pourra la déchiffrer à l'aide de sa clef privée KprA, qu'il est normalement le seul à connaître.

A aucun moment les clefs privées ne sont échangées, ce qui garantit un très haut niveau de sécurité.

Le principe du chiffrement asymétriques repose sur le fait que certaines opérations mathématiques sont très faciles et rapides à faire dans un sens, mais peuvent devenir très complexes et lentes dans l'autre sens. Par exemple, l'algorithme de chiffrement asymétrique RSA, notamment utilisé pour les transactions bancaires, repose sur le fait qu'il est extrêmement facile et rapide de calculer le produit de deux nombres premiers même très grands. Si A et B sont premiers, il suffit d'une simple multiplication pour trouver C = A × B. Par contre si on ne connait que le nombre C, et qu'on veut retrouver les deux nombres premiers A et B tels que A × B = C, il n'existe pas à ce jour d'algorithme permettant de le faire dans un temps raisonnable (il faut en effet tester toutes les possibilités). Il y a donc un lien entre la clef publique Kpu et la clef privée Kpr d'un individu, mais il est impossible de retrouver Kpr à partir de Kpu dans un temps raisonnable.

Pour plus d'informations sur l'algorithme de chiffrement RSA : https://fr.wikipedia.org/wiki/Chiffrement_RSA

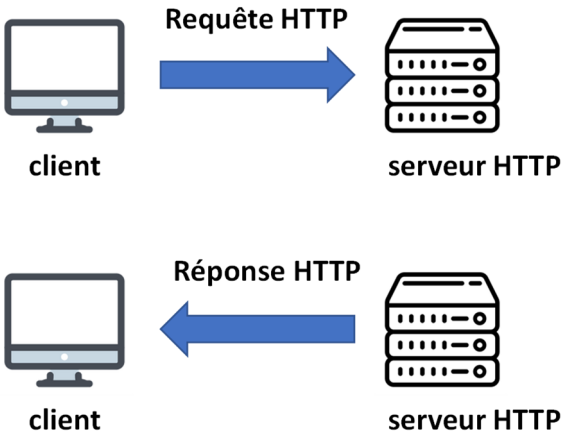
Même si les algorithmes de chiffrement asymétriques assurent une plus grande sécurité que les algorithmes symétriques, ils ne sont pas incassables dans l'absolu :

- Les ordinateurs étant de plus en plus rapides, les attaques de type brute force peuvent être de plus en plus rapides. Ce premier point peut être facilement réglé, il suffit d'augmenter la taille des clefs de chiffrement (actuellement, pour RSA par exemple, on utilise des clefs de 1024 à 2048 bits de longueur).
- De plus des progrès théoriques en mathématiques peuvent aboutir à des algorithmes de recherche performants permettant de résoudre des problèmes jusqu'ici difficiles. Si on trouve un algorithme efficace pour factoriser un grand nombre en produit de nombres premiers, RSA deviendra obsolète.
- Enfin, l'avènement de nouvelles technologies (comme l'ordinateur quantique) peut accélérer grandement certains types de calculs. Des algorithmes « quantiques » permettant de casser RSA sont déjà au point, il ne manque plus que la machine pour les faire tourner ! Mais de nouvelles techniques de chiffrement utilisant les propriétés des ordinateurs quantiques sont déjà à l'étude.

III./ Application pratique des chiffrements symétriques et asymétriques : le protocole HTTPS :

Revenons tout d'abord sur le protocole HTTP : un client HTTP (généralement un navigateur web) va envoyer une requête vers un serveur web, situé sur une machine distante : par exemple, le client peut demander qu'on lui envoie une page web d'URL donnée (c'est-à-dire les fichiers HTML et CSS décrivant la page web, ainsi que tous les autres fichiers liés : images, vidéos, etc…). Le serveur répond alors à la requête du client, en lui envoyant les fichiers demandés.

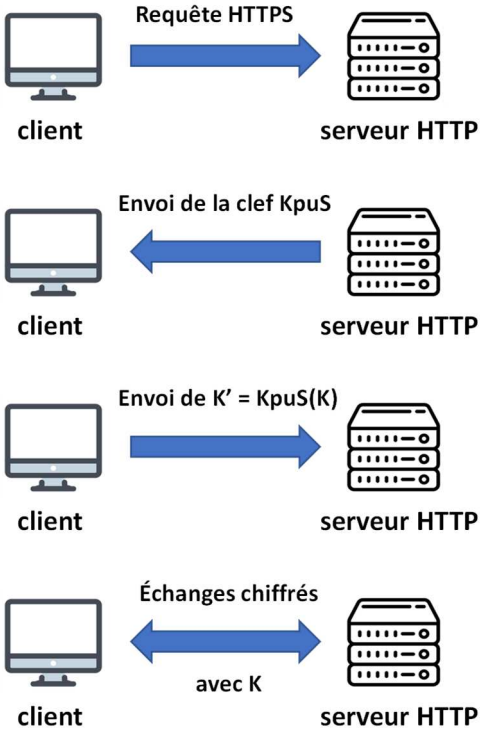
Cet échange, illustré sur le schéma suivant, se fait en clair, c'est-à-dire que les données transmises ne sont à aucun moment chiffrées, et peuvent être facilement espionnées.



Pour sécuriser cet échange de données, il suffit de chiffrer les données, en utilisant un algorithme de chiffrement symétrique pour assurer simplicité et rapidité des échanges. Pour cela, il faut que le client et le serveur connaissent la même clef de chiffrement symétrique K. Comment assurer la communication de la clef de chiffrement symétrique K du client vers le serveur de façon sécurisée ? En chiffrant la clef K à l'aide d'un algorithme de chiffrement asymétrique :

1. Le client envoie une première requête HTTPS au serveur en lui demandant sa clef publique KpuS.
2. Le serveur lui donne sa clef KpuS. Il garde par contre précieusement et de façon secrète sa clef privée KprS.
3. Le client utilise alors un algorithme de chiffrement asymétrique pour chiffrer la clef symétrique K à l'aide de la clef publique KpuS. Il obtient alors une clef symétrique chiffrée K', telle que : K' = KpuS(K). Cette clef chiffrée K' est alors envoyée au serveur.
4. Le serveur va déchiffrer la clef symétrique chiffrée K' à l'aide de l'algorithme de chiffrement asymétrique et de sa clef privée KprS. Il retrouve la clef symétrique K, car K = KprS(K').
5. Le client et le serveur étant tous les deux en possession de la clef symétrique K, ils peuvent l'utiliser pour échanger des données chiffrées par un algorithme symétrique.

Ces étapes sont illustrées par le schéma suivant :



Si un pirate intercepte les communications entre le client et le serveur, il obtiendra des données chiffrées avec la clef symétrique K qu'il ne connait pas. Si le pirate a réussi à intercepter le tout premier échange, il peut obtenir la clef publique KpuS du serveur, ainsi que K', la version chiffrée de la clef symétrique K. Or, comme nous l'avons vu précédemment, il est quasiment impossible, dans un temps raisonnable en tout cas, de retrouver la clef privée KprS du serveur à partir de sa clef publique KpuS. On ne peut donc pas déchiffrer K' pour retrouver K, et on ne peut donc pas déchiffrer le reste des échanges entre les deux machines.

Les connexions sécurisées HTTPS se remarquent grâce à deux détails :

- La présence des lettres « https » en début d'URL, dans la barre d'adresse du navigateur web.
- Le petit cadenas, juste devant l'URL.

Attention : la présence du cadenas veut simplement dire que la connexion est sécurisée par le protocole HTTPS, et que les données seront donc chiffrées entre votre navigateur et la machine distante. Cela n'est en aucun cas l'assurance que vos données ne peuvent pas être récupérées : en effet rien n'empêche un pirate de créer un site en HTTPS et de lui donner l'apparence d'un faux site bancaire ou commercial tout à fait convaincant.

Il y a de moins en moins de sites utilisant le protocole HTTP, pour assurer une plus grande sécurité des échanges de données. Pour inciter la migration des sites existant vers HTTPS, Google utilise ce critère dans son algorithme PageRank, ce qui veut dire qu'un site en HTTP sera moins bien classé par le moteur de recherche de Google, ce qui peut être pénalisant pour générer du trafic.

Image par TheDigitalWay de Pixabay