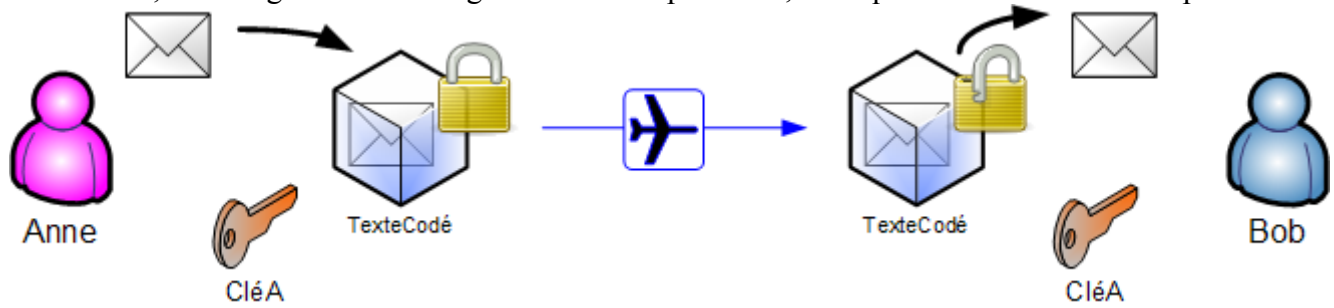
Sécurité des communications

1. Introduction

Les communications, qu'elles soient entre humains, entre humains et machines, ou entre machines, peuvent être interceptées par différents moyens, parfois surprenants[1]. Il est donc important de les protéger, que ce soit pour protéger sa vie privée ou pour éviter un piratage. Pour protéger ces communications, on chiffre[2] les messages à l'aide d'une méthode de cryptographie, comportant une ou plusieurs clés secrètes. Les deux principes de chiffrement sont le chiffrement symétrique, où la même clé est utilisée pour le chiffrement et le déchiffrement du message, et le chiffrement asymétrique, où deux clés différentes sont utilisées. Reste à assurer que les clés ne soient pas piratées, ce qui est le problème de l'authenticité des utilisateurs. La sécurité des communications dépend à la fois des méthodes de chiffrement et des protocoles d'échange des clés.

2. Chiffrement symétrique

Dans le chiffrement symétrique, la même clé permet à la fois de chiffrer et de déchiffrer le message. La clé secrète doit donc être connue préalablement par les deux entités communiquant. On peut remarquer que si la clé fait la taille du message (ou plus), alors le message ne peut pas être décrypté. Comme dit en introduction, il faut également échanger cette clé au préalable, sans qu'elles ne soient interceptées !



a. Des exemples historiques

Le chiffre de César est peut-être le plus connu. César chiffrait ces messages en décalant toutes les lettres de 3 (car C est la troisième lettre de l'alphabet). Ce chiffrement a été utilisé par les égyptiens en -2000.

Plus sophistiqué, le chiffre de Vigénère utilise un décalage alphabétique également, mais avec une clé de plusieurs lettres répétée. Si la clé est NSI, la première et la quatrième lettre du message seront décalées de 14, la deuxième et la cinquième de 19, la troisième et la sixième de 9 etc. Ce chiffre a résisté longtemps au décryptage, et a été utilisé jusque pendant la guerre de Sécession.

Enfin, le fameux code Enigma utilisé par l'Allemagne nazie pendant la deuxième guerre mondiale a été cassé par Alan Turing. Comme les alliés ne voulaient pas que les allemands changent de système, ils ont masqué leur découverte, en rajoutant du « bruit » à l'information recueillie. Par exemple, ils faisaient passer des avions de reconnaissance « par hasard » en vue d'un convoi de ravitaillement des forces de l'axe dont le trajet était connu grâce à Enigma. Et très probablement, les alliés ne ciblaient pas tous les convois repérés.

b. Méthodes actuelles

L'algorithme AES est un standard de chiffrement symétrique. Il est peu coûteux en mémoire, facile à mettre en forme, et il n'existe pas actuellement de méthode plus efficace que la force brute pour le casser.

---

[1] A l'époque des écrans cathodiques, il était possible « d'écouter » au travers d'un mur les variations de fréquence de balayage de l'écran pour trouver les caractères tapés au clavier… Seule solution, écrire dans une fenêtre cachée par une autre (cette histoire est probablement apocryphe, mais elle est très jolie ☺).

[2] Vocabulaire : chiffrer/déchiffrer s'utilise pour transformer un message clair en message codé à l'aide d'une clé. Décrypter, c'est casser le code sans connaître la clé. Crypter n'est pas du français mais du globbish.

3. Chiffrement asymétrique

Lors du chiffrement asymétrique, deux clés sont utilisées, l'une privée (secrète) et l'autre publique. La clé publique permet de chiffrer le message, et la clé privée permet de le déchiffrer. Comme leur nom l'indique, la clé privée n'est pas transmise, contrairement à la clé publique.

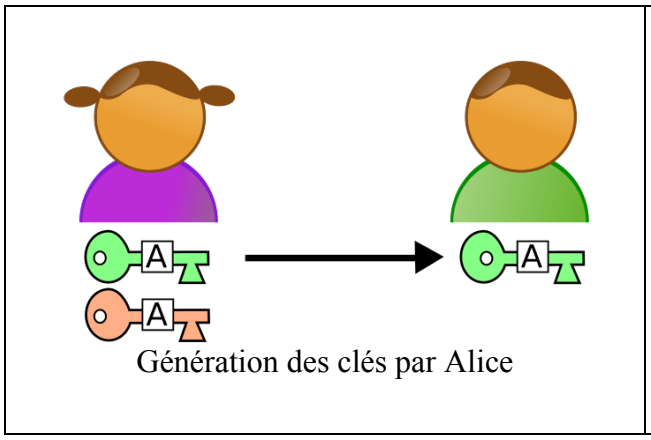
Le principe a été imaginé en 1976 par Diffie et Hellman, mais sans algorithme correspondant. En 1978 Rivest, Shamir et Adleman inventent l'algorithme RSA. Les services secrets anglais avaient déjà imaginé des concepts semblables quelques années plus tôt, mais ont gardé ces recherches secrètes jusqu'en 1997.

Ces protocoles reposent sur l'asymétrie des fonctions utilisées (fonctions dites à sens unique). Par exemple, dans RSA on utilise le produit de deux nombres premiers. Autant il est facile d'effectuer une multiplication de deux nombres premiers, autant il est difficile d'obtenir la décomposition de ce produit en les deux nombres d'origine[3].

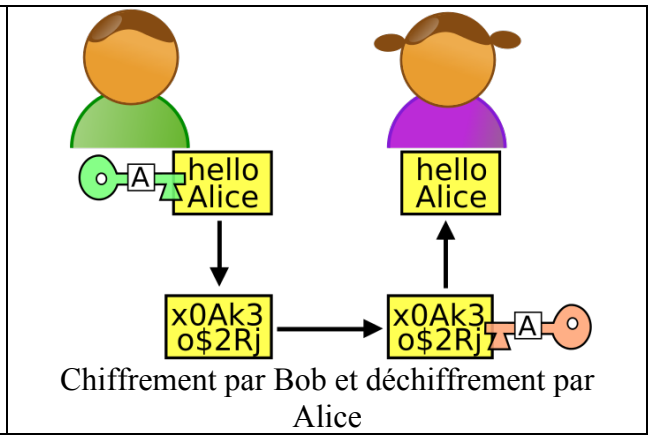
Principe :

Alice souhaite recevoir un message secret de Bob, sur un canal susceptible d'être écouté par Ève.

- Alice génère deux clés (sa clé publique et sa clé privée).
- Elle transmet à Bob la clé publique et conserve précieusement la clé privée.
- Bob chiffre son message avec la clé publique et l'envoie à Alice.
- Alice déchiffre le message avec sa clé privée.
- Si Ève intercepte le message, elle ne peut pas le déchiffrer, ne disposant pas de la clé privée.



Génération des clés par Alice



Chiffrement par Bob et déchiffrement par Alice

On peut compléter ce protocole avec un mécanisme d'authentification (signature des messages) ou des certificats de sécurité. En effet, Alice ne peut pas savoir un message reçu provient de Bob ou d'Ève, vu qu'Ève dispose également de la clé publique (cf. ci-après)

4. Compléments

a. Échange de clé de Diffie-Hellman

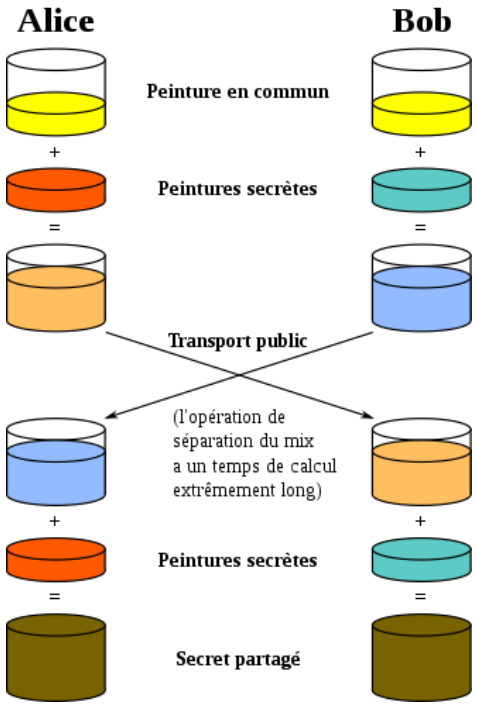
Ce protocole permet à Alice et Bob de convenir d'une clé symétrique, par un moyen de communication non sûr. Pour cela, il faut disposer d'une « fonction à sens unique » possédant certaines propriétés. Le mélange de couleurs en est une bonne métaphore : il est facile de mélanger deux couleurs, par contre retrouver les teintes d'origine est complexe. Par ailleurs cette fonction a la propriété fondamentale, qui est que le mélange de trois couleurs donne le même résultat, quelque soit l'ordre dans lequel on le fait.

Le schéma de Wikipédia (et votre livre p 412) sont très clairs :

- Alice et Bob décident d'une couleur en commun (jaune), connue de tous, y compris Eve.
- Chacun choisit une couleur secrète (Alice le rouge, Bob le vert).
- Puis mélange sa couleur secrète avec la couleur commune.

[3] Autre exemple de problème à sens unique : le problème du sac à dos. On a vu en 1ère que ce problème était de complexité exponentielle. Par contre, vérifier une solution est presque immédiat (complexité linéaire)

- Alice et Bob s'échangent leur mélange. Ève peut intercepter les mélanges, mais ne pourra que très difficilement extraire les couleurs d'origine.
- Alice et Bob mixent le mélange avec leur couleur secrète. Ils obtiennent la même teinte (kaki), inconnue d'Ève.



b. Attaque de l'homme du milieu

Cette attaque consiste à intercepter les communications entre deux entités, sans que ni l'une ni l'autre ne le réalise. Comme son nom l'indique, il suffit pour un attaquant de se placer entre les deux interlocuteurs et d'en usurper l'identité. Par exemple, dans le protocole d'échange de clés de Diffie-Hellman, Eve peut choisir aussi une peinture secrète et intercepter l'échange des mélanges. Elle créera alors deux mélanges « secrets partagés » : un pour Alice et un pour Bob. Elle pourra ainsi intercepter les messages, et par exemple dans le cas d'un message d'Alice à Bob, elle le déchiffrera avec le mélange partagé avec Alice pour le lire, puis le chiffrera avec celui de Bob, afin qu'il le reçoive bien et ne se doute de rien. Dans ce cas la puissance du chiffrement est inutile !

Pour pallier à ce problème, des procédures d'authentification sont mises en place. La plus connue sur le web est celle des certificats de sécurités (X.509 ou OpenPGP principalement). Ces certificats sont gérés par des tiers de confiance.

Remarque : sous Firefox, vous pouvez afficher le certificat de sécurité d'un site accessible en https. Cliquez sur le cadenas, puis sur « > », « plus d'information », « afficher le certificat ».

c. Protocoles hybrides

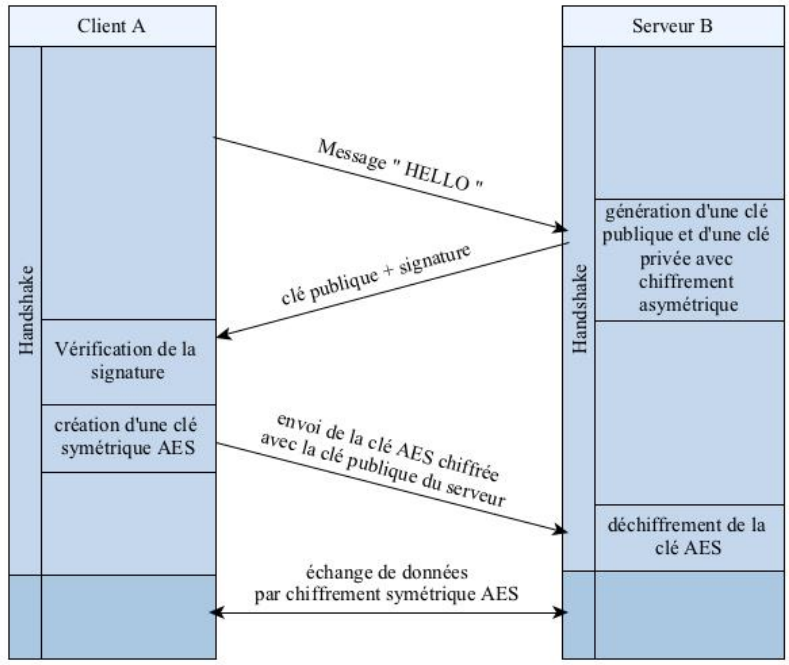
Les protocoles asymétriques sont difficiles à casser mais lents. Les protocoles symétriques sont rapides, mais nécessitent l'échange préalable des clés. Les protocoles hybrides utilisent les deux méthodes. RSA peut par exemple être utilisé pour échanger une clé de protocole symétrique.

Par exemple, TLS/SSL (le « https ») utilise en partie un chiffrement asymétrique, pour initialiser la connexion client/serveur, et créer une clé symétrique utilisée pour la suite de la session. Ce protocole mélange donc les deux types de chiffrement. Cette méthode est couplée avec un mécanisme d'authentification : l'utilisation des certificats de sécurité, confiés à un tiers de confiance.

d. https plus en détail

Le protocole http permet d'échanger des données entre client et serveur. En première, on voit deux possibilités de http avec les méthodes GET et POST. Ce protocole n'est pas sécurisé, il n'est donc pas adapté à tous les échanges de données sur le web. https rajoute à http une couche de sécurité, la couche TLS (plus sûre que SSL).

Les protocoles de communication commencent par une mise en relation des entités. On appelle souvent cette mise en relation la « poignée de mains » (handshake).



- La vérification de la signature se fait auprès d'autorités extérieures (les tiers de confiance)
- La clé symétrique est une clé AES, qui est chiffrée avec la clé publique du serveur. Elle ne sert que pour une seule session : une autre clé est générée s'il y a déconnexion et reconnexion.
- La clé symétrique est déchiffrée par le serveur avec sa clé privée : on utilise deux protocoles successivement. Protocole asymétrique pour la poignée de mains, puis protocole symétrique pour l'échange des données.

Plus de détails, tout en restant dans la vulgarisation, sur cette bande dessinée (en anglais) : https://howhttps.works/
Test des vulnérabilités de votre navigateur :
https://clienttest.ssllabs.com:8443/ssltest/viewMyClient.html

e. Le futur

Les protocoles actuels sont très solides, les coûts de décryptage étant prohibitifs. Par contre, avec un ordinateur quantique ils seront cassés très rapidement. Il faudra alors passer à des méthodes de cryptographie quantique. Notamment, lors de transmissions de paquets quantiques, une éventuelle interception est toujours détectable (elle modifie la nature du paquet transmis). Cela permet au moins de savoir que la communication a été interceptée.

EXERCICES

1. UN PROTOCOLE PAS SI SÉCURISÉ QUE CELA

Alice et Bob souhaitent communiquer, en s'envoyant des messages dans une boîte fermée à clef. Bien sûr, Eve veut les espionner. Ils utilisent le protocole suivant :

- Alice envoie à Bob la boîte ouverte, ne contenant que la clé publique d'Alice.
- Bob place le message dans la boîte puis ferme celle-ci à l'aide de la clé publique d'Alice.
- Bob envoie la boîte fermée à Alice.
- Alice utilise sa clé privée pour ouvrir la boîte.
- Alice récupère le message de Bob.

Alice et Bob sont fiers de leur protocole, mais Ève rit sous cape et lit facilement les messages de Bob. Comment fait-elle ?

2. UN EXEMPLE DE CHIFFREMENT SYMÉTRIQUE

Le XOR est très utilisé dans les protocoles de chiffrement symétrique. En effet, c'est une opération qui est sa propre réciproque, ce qui n'est pas le cas du ET ni du OU. C'est à dire que : $A \oplus B = C \Leftrightarrow A \oplus C = B \Leftrightarrow C \oplus B = A$ (1)

a. Démontrer la propriété (1) à l'aide de tables de vérité

b. On va programmer un protocole de chiffrement symétrique utilisant le XOR.
On souhaite chiffrer un message (par exemple une chaîne de caractères) à l'aide d'une clé, qui peut aussi être une chaîne de caractères. Pour simplifier la programmation, on supposera que seuls des caractères ASCII sont utilisés (pas d'utf-8 ou plus généralement d'Unicode).
   - Recopier la clé à la suite d'elle-même autant de fois que nécessaire, pour que la longueur totale soit égale à celle du message (on tronque si nécessaire).
   - Transcrire le message en binaire
   - Transcrire la suite de clés en binaire
   - Le message chiffré est obtenu avec un XOR des deux nombres binaires précédents.
   - On peut alors retranscrire le message en caractères.
   - Le déchiffrement se fait avec un XOR entre le message chiffré en binaire et le nombre binaire correspondant à la suite des clés.

   Programmer ce chiffre en Python (deux fonctions au moins, un pour chiffrer et une pour déchiffrer). On peut se contenter de chiffrer un nombre, voire un tableau de 0/1, et ainsi de ne pas écrire les fonctions de codage des chaînes en binaire.

3. Un exemple de chiffrement asymétrique

Dans cet exercice, on va programmer le « RSA pour les enfants », qui est un algorithme public.
Création des clés par Alice. Elle :

- choisit quatre nombres a, b, $a1$ et $b1$;
- Calcule

$$M = a \times b - 1 \qquad e = a1 \times M + a \qquad d = b1 \times M + b$$

$$n = \frac{e \times d - 1}{M} = a1 \times b1 \times M + a \times b1 + a1 \times b + 1$$

- La clé publique d'Alice est (n, e) et sa clé privée est (n, d)

Échange des messages :

- Pour envoyer un message à Alice, Bob utilise la fonction $M_{\text{chiffré}} \equiv e \times M_{\text{clair}} [n]$, c'est-à-dire en Python : `message_chiffré = (e * message_clair) % n`
- Pour déchiffrer le message, Alice utilise sa clé privée : $M_{\text{clair}} \equiv d \times M_{\text{chiffré}} [n]$, c'est-à-dire en Python : `message_clair = (d * message_chiffré) % n`

a. Exemple
Vérifier qu'avec les nombres $a = 9, b = 11, a1 = 5, b1 = 8$ et le message $M_{\text{clair}} = 538$, alors le message chiffré est $M_{\text{chiffré}} = 1294$. Vérifier que la méthode de déchiffrement est correcte.

b. Programmer ce chiffre en Python (deux fonctions au moins). L'objectif est de chiffrer un nombre (et non une chaîne).

c. Écrire la fonction d'Ève qui permet de décrypter ce chiffre par force brute. Est-elle compliquée ? Lente ?

d. Pour ceux qui font mathématiques expertes la méthode de décryptage « intelligente » est là : http://math.uttyler.edu/sjgraves/aam/sec-PublicKey-KidRSA.html
Conclusion : un protocole asymétrique ne suffit pas, il faut aussi un algorithme de chiffrement puissant !

4. Exercices du livre 185 150 p 135

Merci aux contributeurs de la liste NSI et à leurs idées toujours très riches.