MINISTÈRE
DE L'ÉDUCATION
NATIONALE,
DE LA JEUNESSE
ET DES SPORTS

Liberté
Égalité
Fraternité

VOIE GÉNÉRALE

2DE | 1RE | TLE

Numérique et sciences informatiques

ENSEIGNEMENT
SPÉCIALITÉ

SÉCURISATION DES COMMUNICATIONS

Référence au programme

« La protection des données sensibles échangées est au cœur d'internet. Les notions de chiffrement et de déchiffrement de paquets pour les communications sécurisées sont explicitées. »

Sécurisation des communications.	Décrire les principes de chiffrement symétrique (clef partagée) et asymétrique (avec clef privée/clef publique). Décrire l'échange d'une clef symétrique en utilisant un protocole asymétrique pour sécuriser une communication HTTPS.	Les protocoles symétriques et asymétriques peuvent être illustrés en mode débranché, éventuellement avec description d'un chiffrement particulier. La négociation de la méthode chiffrement du protocole SSL (*Secure Sockets Layer*) n'est pas abordée.

Avec l'essor d'Internet, certains échanges comme les achats en ligne, les démarches administratives ou encore les transactions bancaires nécessitent de pouvoir garantir l'authenticité et la confidentialité des données échangées.

Dans cette ressource, la première partie permet d'**illustrer le principe de chiffrement symétrique** avec des exemples classiques comme le chiffre de César ou bien le chiffrement de Vernam. Puis de faire le **lien avec des chiffrements symétriques plus récents** comme DES et AES.

La deuxième partie aborde le **principe de chiffrement asymétrique** en prenant comme exemple le chiffrement RSA.

La troisième partie se concentre sur **l'échange d'une clef symétrique en utilisant un protocole symétrique** pour sécuriser une communication HTTPS.

Le principe de chiffrement

Le souci de confidentialité des échanges entre des personnes n'est pas récent. En effet, dès l'Égypte ancienne (vers 2700-2195 av. J.-C.), on retrouve des procédés cryptographiques[1] rendant des messages aux contenus religieux illisibles pour les profanes.

Bien évidemment, dès l'apparition de messages chiffrés, des personnes ont essayé de les déchiffrer. Et plus les spécialistes du décodage (appelés **briseurs de chiffrements**) avancent dans leurs recherches, plus les chiffrements doivent évoluer. Une fois qu'une faiblesse d'un chiffrement est connue, celui-ci devient inutilisable et il faut trouver une nouvelle méthode de chiffrement. Ce combat acharné entre les concepteurs de chiffrements et les spécialistes pour déchiffrer ceux-ci a permis de faire avancer la recherche, surtout en informatique.

Aujourd'hui, la cryptographie est utilisée quotidiennement, notamment par les navigateurs Web des internautes, dans le cadre des communications « HTTPS » et du protocole sous-jacent SSL/TLS qui sécurisent les échanges.

Définir quelques bases de la **cryptographie** en abordant les principes de **chiffrement symétrique** et **asymétrique** est nécessaire avant d'aborder la sécurisation des communications.

La **cryptographie** (du grec ancien *kruptos*, « caché », et *graphein*, « écrire ») a pour objectif de protéger des messages.

Le **chiffrement** consiste à transformer, à l'aide d'une clef, un message compréhensible (dit **texte clair**) en un message incompréhensible (dit **texte chiffré**). Il est extrêmement difficile, voire impossible, de retrouver le texte clair à partir du texte chiffré pour toute personne qui ne dispose pas de la clef de chiffrement. Le texte chiffré peut alors être envoyé sur un média qui n'est pas sécurisé, puisqu'il est incompréhensible sans la clef.

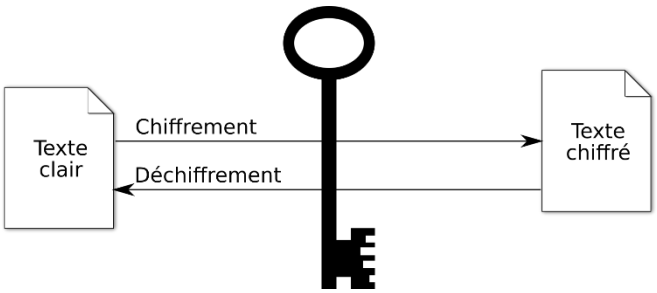
Pour aller plus loin

La suite de cette ressource s'intéresse au chiffrement et au déchiffrement. Même si un message est indéchiffrable, la communication chiffrée entre deux personnes donne tout de même des informations sur leurs liens, puisqu'elles échangent des informations qui se veulent secrètes. Par exemple, un internaute qui se connecte en HTTPS (HTTP sécurisé) à un site illégal va laisser une trace de cette connexion, même si le contenu est chiffré. On peut aussi noter que certains algorithmes peuvent être vulnérables s'ils ne sont pas employés correctement. Par exemple, divulguer en clair certains messages chiffrés, ou utiliser un en-tête identique dans tous les messages en clair peut donner des informations supplémentaires à un attaquant. C'est par exemple le cas d'Enigma lors de la Seconde Guerre mondiale, dont les Alliés ont réussi à casser le code[2].

1. Musée Josèphe Jacquiot : http://baladeaumusee.e-monsite.com/pages/fenetres-sur/la-cryptographie.html
2. TPE réalisé par Guillaume Munch et Julien Milli : *ENIGMA et la Seconde Guerre mondiale*
   Émission « La méthode scientifique » du 10/10/19 à partir de 6'36'': Enigma, les secrets du code nazi

Les principes de chiffrement symétrique

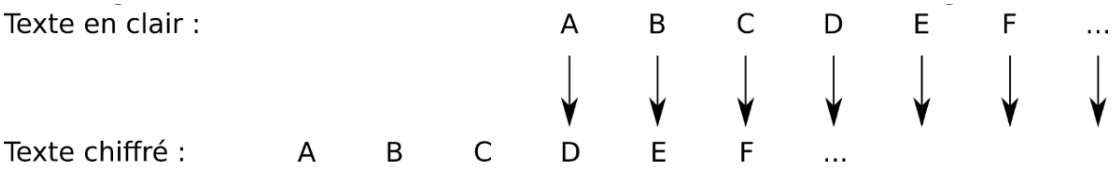
Le chiffrement symétrique, ou à **clef partagée**, permet de chiffrer et de déchiffrer des messages à l'aide d'une même clef. Cette clef doit donc être partagée par l'expéditeur et son destinataire tout en n'étant connue que par eux.



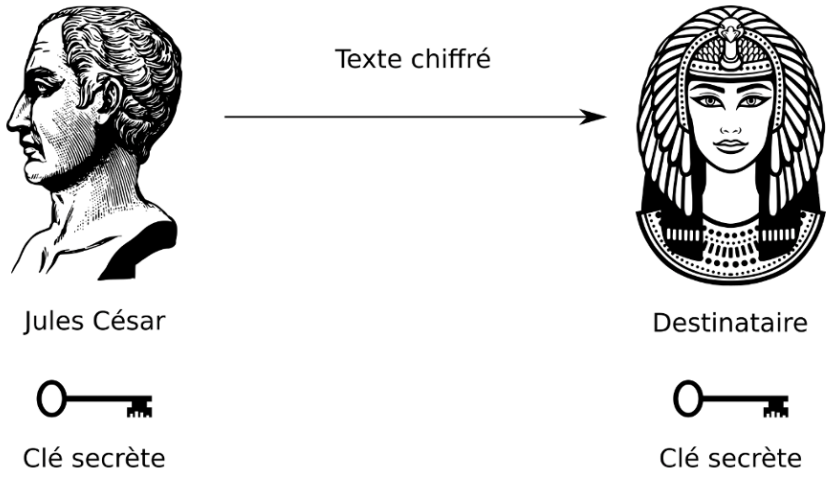
Le chiffre de César

L'exemple le plus connu de chiffrement symétrique, et un des plus simples, est le **chiffre de César**. Cette méthode de communication chiffrée aurait été inventée par Jules César lui-même[3]. Pour obtenir le texte chiffré, le chiffre de César consiste à remplacer chaque lettre du texte en clair par la lettre obtenue après un décalage d'un nombre fixe de lettres dans l'alphabet.

Pour un décalage de 3 lettres, le A devient D, le B devient E… Comme le montre l'image ci-dessous :



Le nombre 3, qui correspond au nombre de lettres à décaler, est appelé la **clef secrète de chiffrement**. Pour déchiffrer le message, il suffit de donner la clef de chiffrement au destinataire du message qui réalise l'opération inverse pour déchiffrer le message.



3. Extrait de *La vie des douze Césars* de Suétone : « On possède enfin de César des lettres à Cicéron, et sa correspondance avec ses amis sur ses affaires domestiques. Il y employait, pour les choses tout à fait secrètes, une espèce de chiffre qui en rendait le sens inintelligible (les lettres étant disposées de manière à ne pouvoir jamais former un mot), et qui consistait, je le dis pour ceux qui voudront les déchiffrer, à changer le rang des lettres dans l'alphabet, en écrivant la quatrième pour la première, c'est-à-dire le D pour le A, et ainsi de suite. »

Jules César et son destinataire utilisent la même clef de chiffrement et elle n'est connue que par eux. Ce chiffrement est donc symétrique et cette clef est la **clef partagée**.

Déchiffrer le message chiffré « qf hwduytlwfumnj jxy zynqnxjj ifsx qjx sfanlfyjzwx bjg ! » avec la clef partagée 5 en débranché permet de mieux appréhender la notion de chiffrement symétrique. Le texte déchiffré est alors « la cryptographie est utilisée dans les navigateurs web ! ».

Fonction Python qui déchiffre un message encodé par le chiffrement de César avec un décalage de *n* caractères :

```
def dechiffrement_cesar(texte_a_dechiffrer, n):
    a = "abcdefghijklmnopqrstuvwxyz"
    return ''.join([l if not(l in a) else a[(a.index(l) - n) % len(a)]\
                    for l in texte_a_dechiffrer])
```

Seconde version plus adaptée aux élèves :

```
def dechiffrement_cesar_eleve(texte_a_dechiffrer, n):
    a = "abcdefghijklmnopqrstuvwxyz"
    texte_chiffre = ""
    for lettre in texte_a_dechiffrer:
        if lettre in a:
            indice = (a.index(lettre) - n) % len(a)
            lettre_chiffre = a[indice]
        else:
            lettre_chiffre = lettre
        texte_chiffre = texte_chiffre + lettre_chiffre
    return texte_chiffre
```

Fonction Python qui chiffre un message en utilisant le chiffrement de César avec un décalage de *n* caractères. Le texte à chiffrer doit être écrit en minuscule et sans accent. En plus de l'ensemble du code et en complément de ce document, le fichier **Chiffrement_Cesar.py** (téléchargeable ici) propose une fonction qui calibre le texte à chiffrer sous ce standard. Les espaces et signes de ponctuation sont alors ignorés :

```
def chiffrement_cesar(texte_a_chiffrer, n):
    a = "abcdefghijklmnopqrstuvwxyz"
    return ''.join([l if not(l in a) else a[(a.index(l) + n) % len(a)] \
                    for l in texte_a_chiffrer])
```

Seconde version plus adaptée aux élèves :

```
def dechiffrement_cesar_eleve(texte_a_chiffrer, n):
    a = "abcdefghijklmnopqrstuvwxyz"
    texte_chiffre = ""
    for lettre in texte_a_chiffrer:
        if lettre in a:
            indice = (a.index(lettre) + n) % len(a)
            lettre_chiffre = a[indice]
        else:
            lettre_chiffre = lettre
        texte_chiffre = texte_chiffre + lettre_chiffre
    return texte_chiffre
```

Ce chiffrement fait apparaître quelques limites intéressantes, dont le principe peut s'appliquer à d'autres algorithmes plus modernes :

- la clef 0 n'est pas utilisable, car elle ne chiffre pas le message (notion de **clef faible**);
- si on se limite à l'alphabet sans ajouter de signe de ponctuation, le nombre de clefs possibles est limité à 25. Il est donc possible de toutes les tester et il s'agit alors d'une attaque dite par « **force brute** »;
- l'informatique permet également de réaliser une analyse de la fréquence des occurrences de chaque lettre dans une langue et de la comparer aux fréquences trouvées dans le message chiffré. On sait par exemple qu'en français, la lettre la plus fréquente est le « e ». Si la lettre la plus fréquente du message chiffré est le « f », la clef vaut alors certainement 1.

Fonction Python qui renvoie la lettre la plus fréquente du texte passé en argument. Si des lettres ex æquo apparaissent, on choisit parmi les plus fréquentes la première qui apparait dans le message chiffré.

Version utilisant un dictionnaire :

```
def analyse_frequentielle_avec_dico(texte_chiffre):
    dico = {}
    for caractere in texte_chiffre:
        if caractere in "abcdefghijklmnopqrstuvwxyz":
            if caractere in dico:
                dico[caractere] = dico[caractere] + 1
            else:
                dico[caractere] = 1

    valeurs = list(dico.values())
    clefs = list(dico.keys())

    return clefs[valeurs.index(max(valeurs))]
```

Seconde version utilisant une liste :

```
def analyse_frequentielle_avec_liste(texte_chiffre):
    alphabet = "abcdefghijklmnopqrstuvwxyz"
    occurences = [0 for i in alphabet]
    for caractere in texte_chiffre:
        if caractere in alphabet:
            occurences[alphabet.index(caractere)] += 1
    return alphabet[occurences.index(max(occurences))]
```

Fonction qui déchiffre le message passé en argument à partir de la recherche de la lettre la plus fréquente dans le texte passé en argument :

```
def dechiffrement_cesar_automatique(texte_chiffre):
    lettre_la_plus_frequente = analyse_frequentielle(texte_chiffre)
    n = ord(lettre_la_plus_frequente) - ord('e')
    print("La clef partagée semble être", n)
    return dechiffrement_cesar(texte_chiffre, n)
```

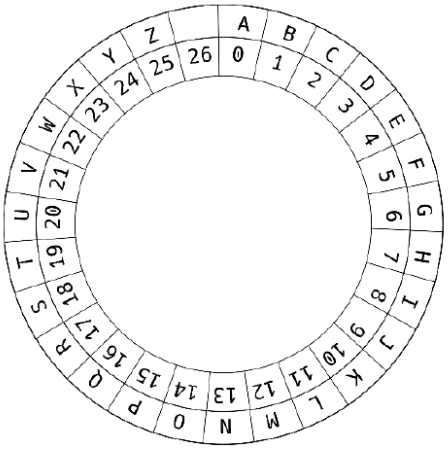
Le chiffre de César est un exemple simple de chiffrement symétrique. Dans une des dernières utilisations connues, l'armée russe chiffra certaines de ses communications avec le chiffre de César durant la Première Guerre mondiale, mais les analystes autrichiens et allemands cassèrent ce code très simplement. De nos jours, le chiffre de César n'est uniquement utilisé qu'à des fins pédagogiques, car il est trop facilement déchiffrable.

Le chiffre de Vernam

Le chiffrement symétrique dit de Vernam utilise comme clef partagée une chaîne de caractères aussi longue que le message à chiffrer. L'algorithme consiste alors à :

- faire correspondre chaque lettre du message à chiffrer avec chaque lettre de la clef partagée ;
- convertir chaque lettre en nombre. Par exemple : 0 pour A, 1 pour B, 2 pour C… Le caractère « espace » est aussi codé par le nombre 26. Les décalages sont alors modulo 27 : la lettre est codée par le reste de la division euclidienne par 27.
- Il est aussi possible d'utiliser le codage ASCII pour convertir chaque lettre en nombre : 65 pour A, 66 pour B… ;
- décaler dans l'alphabet chaque lettre du message à chiffrer. Ce décalage est égal au nombre correspondant à la lettre qui a la même position dans la clef partagée. **Cela revient à effectuer un chiffrement de César avec un décalage différent pour chaque lettre du message à chiffrer.**

Attention : pour garantir la sécurité de la méthode, la clef doit être choisie de façon aléatoire et ne pourra servir qu'une fois.



En utilisant le codage de l'illustration, le texte :

L	A		T	E	R	R	E		E	S	T		R	O	N	D	E

chiffré avec la clef :

Q	W	R	B	O	K	L	E	P	Y	C	F	O	R	A	H	Z	Q

devient :

A	W	Q	U	S	A	B	I	O	B	U	Y	N	H	O	U	B	U

Prenons l'exemple de la lettre S à chiffrer. Sa lettre correspondante dans la clef est C. Ceci occasionne un décalage de 2 dans l'alphabet et donc la lettre S devient U.

Fonction Python qui calcule l'index de la lettre du texte à chiffrer (sens = 1) ou à déchiffrer (sens = -1) puis qui renvoie le texte chiffré ou déchiffré. Le texte et la clef ne doivent contenir que des lettres en minuscules et des espaces.

```
def vernam(texte, cle, sens = 1):
    assert len(texte) == len(cle)
    a = "abcdefghijklmnopqrstuvwxyz "
    return "".join([a[(a.index(t) + sens * a.index(c)) % len(a)] \
                    for t, c in zip(texte, cle)])
```

Seconde version plus adaptée aux élèves :

```
def vernam_eleve(texte, cle, sens=1):
    assert len(texte) == len(cle)
    a = "abcdefghijklmnopqrstuvwxyz "
    chaine = ""
    for t, c in zip(texte, cle):
        indice = (a.index(t) + sens * a.index(c)) % len(a)
        chaine = chaine + a[indice]
    return chaine
```

Le fichier **Chiffrement_Vernam.py** en complément de cette ressource (téléchargeable ici) propose, en plus de l'ensemble du code, une fonction qui calibre le texte à chiffrer et la clef pour éviter les problèmes dus à la présence de majuscules ou de ponctuation.

Pour aller plus loin : dans le cas où les données sont codées en binaire, le décalage est alors un modulo 2 et la méthode est encore plus simple et rapide. En effet, on peut démontrer qu'elle consiste à utiliser un opérateur booléen « ou exclusif » ou « XOR » sur chaque bit du message en clair et du bit correspondant de la clef. Le déchiffrement est tout aussi simple en utilisant l'opérateur booléen « XOR » entre le message chiffré et la clef.

On rappelle ci-contre la table de vérité de l'opérateur XOR :

Reprenons l'exemple de la lettre S à chiffrer (position 18), avec sa lettre correspondante dans la clef qui est U (position 21).

$18_{10} = 10010_2$ et $21_{10} = 10101_2$

Alors : 10010 XOR 10101 = 00111

Pour déchiffrer : 00111 XOR 10101 = 10010 qui correspond bien à 18 en base décimale.

Table de vérité de XOR

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Une attaque par force brute n'est pas réalisable, car on peut faire dire « n'importe quoi » avec une clef adéquate. Par exemple, si on déchiffre « awqusabiobuynhoubu » avec « qwrboklepycfo**tdujq** », on décode la phrase « la terre est plate ».

Utilisé correctement, cet algorithme est théoriquement inviolable. Il faut néanmoins faire attention à certains écueils :

- une clef de chiffrement déjà utilisée ne doit pas être réutilisée puisque si deux messages ont été chiffrés avec la même clef de chiffrement, il suffit d'effectuer une soustraction pour annuler l'effet du masque et avec un peu plus de travail, de retrouver les deux textes en clair ;
- la clef se doit d'être aléatoire, par exemple en la générant avec un générateur aléatoire. Si ce n'est pas le cas, il est possible de retrouver la séquence qui l'a générée ;
- comme la clef, aussi grande que les données échangées, n'est pas réutilisable, il faut au préalable partager, par un autre canal sécurisé, une assez grande quantité de données pour permettre les futurs échanges.

Ce système a été utilisé durant la guerre froide par le « Téléphone rouge » entre le Kremlin et la Maison-Blanche, qui était en fait plus un téléscripteur qu'un réel téléphone. La « clef », quant à elle, était échangée par valise diplomatique.

Le chiffrement AES

Le chiffrement AES (*Advanced Encryption Standard*) est un des algorithmes les plus utilisés actuellement et il est à l'heure actuelle considéré comme sûr.

Dans les utilisations les plus fréquentes de l'AES, on peut citer :

- le chiffrement des données lors de l'ajout d'un mot de passe à un fichier PDF ou ZIP ;
- la sécurisation des connexions via l'utilisation de VPN ;
- la sécurisation des données utilisateurs lors de l'utilisation d'outils de gestion de mots de passe ;
- la protection de serveurs multi-joueurs contre les attaques. C'est le cas par exemple de l'entreprise Rockstar, développeur et éditeur de la série des Grand Theft Auto (GTA) ;
- le chiffrement des communications lors de l'utilisation de messageries instantanées. C'est le cas de l'application WhatsApp.

Ce chiffrement symétrique utilise une clef de 128, 192 ou 256 bits et elle est utilisée pour paramétrer une suite de transformations qui permettent de chiffrer ou de déchiffrer le message.

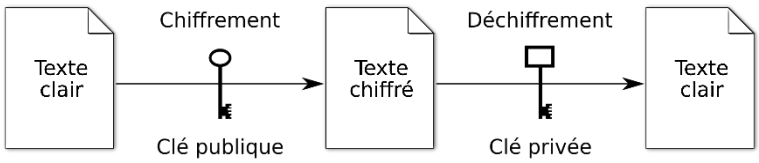
Le fonctionnement plus complexe de ce chiffrement par bloc dépasse le cadre des programmes. Pour aller plus loin, il est possible de prendre connaissance, par exemple, du cours sur le sujet édité par le laboratoire lorrain de recherche en informatique et ses applications (LORIA)[4].

Les principes du chiffrement asymétrique

Dans le cas du chiffrement symétrique, la clef est partagée. Elle est alors commune à l'émetteur et au récepteur du message chiffré. Le chiffrement est symétrique, car une unique clef permet à la fois de chiffrer et de déchiffrer le message.

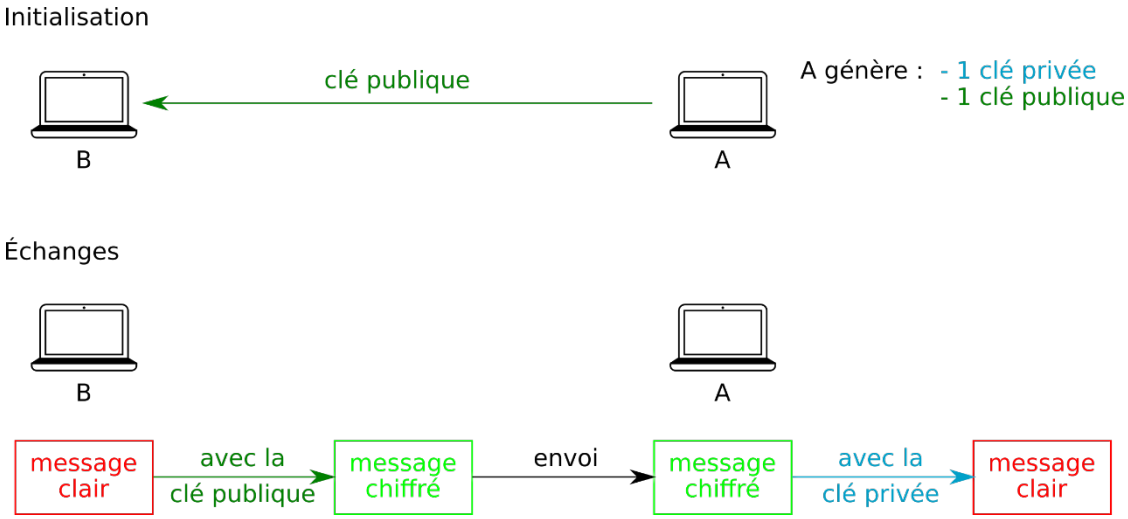
Pour le **chiffrement asymétrique**, la clef est en fait un **couple de clefs**, appelées **clef privée** et **clef publique**. Chaque clef peut être utilisée pour chiffrer un message, mais il n'est alors déchiffrable qu'avec l'autre clef du couple.

Dans les faits, la clef publique chiffre et la clef privée déchiffre :

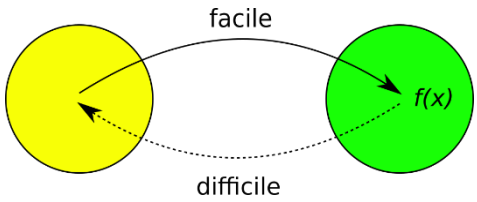


Plus précisément, dans une communication, lorsqu'on chiffre un message avec un algorithme asymétrique, on utilise la **clef publique du destinataire**. Le message ou les données ne sont alors déchiffrables que par la personne qui possède l'autre clef du couple, normalement la **clef privée du destinataire**. Contrairement à la clef publique, la clef privée n'est jamais diffusée.

4. https://members.loria.fr/PZimmermann/cours/id12-2005-4.pdf



Le chiffrement asymétrique repose sur des fonctions mathématiques à sens unique. En effet, une fois qu'elles ont été appliquées, il est très difficile de les inverser sans connaître l'information contenue dans la clef de déchiffrement.



Les chiffrements asymétriques sont plus coûteux en temps de traitement que les chiffrements symétriques, et à niveau de sécurité équivalent, les clefs sont généralement plus longues que les clefs des algorithmes symétriques.

Le chiffrement RSA

Le concept de chiffrement asymétrique est attribué à Whitfield Diffie et à Martin Hellman, qui l'ont présenté en 1976. Le premier exemple de chiffrement asymétrique est le chiffrement RSA (abréviation des noms de ses inventeurs Ronald Rivest, Adi Shamir, Loenard Adleman).

La sécurité du RSA repose sur la facilité à multiplier deux nombres premiers entre eux et la difficulté de factoriser le produit de deux nombres premiers, pourvu qu'ils soient suffisamment grands. C'est à l'heure actuelle l'algorithme de chiffrement asymétrique le plus utilisé.

Pour comprendre les mécanismes sous-jacents au chiffrement RSA, et ceci n'est pas un attendu des programmes, on pourra lire l'article édité sur Interstices.info[5] qui propose aussi une animation interactive de ce chiffrement.

À noter qu'il existe d'autres algorithmes asymétriques, qui dépassent eux aussi les attendus des programmes. On peut citer, par exemple, la « cryptographie sur les courbes elliptiques[6] » qui permet d'avoir un niveau de sécurité comparable à celui de RSA, mais en utilisant une clef beaucoup plus courte. Cette clef plus courte engendre des calculs plus rapides ainsi qu'une utilisation moins importante de mémoire et d'énergie.

5. https://interstices.info/nombres-premiers-et-cryptologie-lalgorithme-rsa/#2
6. http://www.lix.polytechnique.fr/~goncalves/Downloads/Cours2_Courbes_elliptiques.pdf#Outline0.3

Autre utilisation du chiffrement symétrique : la signature numérique

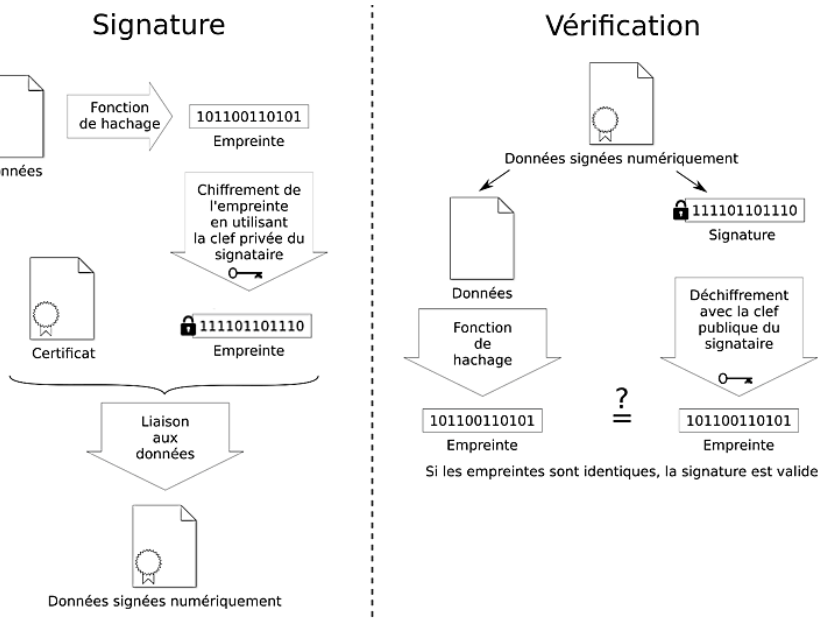
Le chiffrement asymétrique permet également de réaliser une signature numérique. Par analogie avec la signature traditionnelle d'un document papier, la signature numérique :

- relie un document à son auteur ;
- est difficilement imitable.

Contrairement à une signature « papier », la signature numérique possède des propriétés supplémentaires :

- elle appartient à un seul message. Il est donc impossible de la copier pour la coller sur un autre message ;
- elle ne peut pas être falsifiée ni reniée.

La signature numérique est calculée à partir de la clef privée du signataire et peut être vérifiée en utilisant la clef publique du signataire.



Le protocole HTTPS

Le protocole HTTPS est la version sécurisée du protocole HTTP.

L'objectif du protocole HTTP est de recevoir et d'envoyer des informations de et vers des serveurs web sans se soucier de la façon dont ces informations se déplacent d'un endroit à un autre. Le protocole HTTP n'étant pas sécurisé, il est donc possible de :

- falsifier un site : on ne peut avoir la garantie que le site auquel on se connecte est bien le bon ;
- intercepter et altérer les communications : n'importe quel élément du réseau sur le chemin de la connexion peut consulter les données échangées, et même les modifier au passage. Par exemple, modifier le montant d'une transaction bancaire, sans qu'il soit possible de le détecter.

Pour résoudre ces problèmes, HTTPS utilise la suite de protocoles SSL/TLS qui met en jeu les 3 principes vus précédemment : signature numérique, cryptographie asymétrique et cryptographie symétrique.

Authentification du site

Lors de l'accès à un site utilisant le protocole HTTPS, le serveur envoie sa clef publique, ainsi qu'une signature numérique de cette clef, c'est ce qu'on appelle le **certificat du site**. Pour être valide, la signature du certificat du site doit avoir été réalisée par une autorité en laquelle le navigateur a confiance.
En pratique, le navigateur ou le système d'exploitation maintiennent une liste de clefs publiques en laquelle ils ont confiance, appelés **certificats racines**, et avec lesquelles ils vérifient le certificat présenté par le serveur web. Un principe résume correctement cette vérification : « **les amis de mes amis sont mes amis** ». Cela permet de ne pas avoir à stocker dans le navigateur l'intégralité des certificats des sites sûrs.

L'inconvénient de cette méthode est que cette liste de certificats racines est critique. En effet si une entrée « malveillante » est ajoutée, tous les sites qui auront été signés avec cette entrée seront considérés comme sûrs par le navigateur.

En pratique, d'autres informations sont vérifiées comme la date de validité et la liste de révocations.

Chiffrement des communications

Une fois le serveur authentifié, la communication peut commencer. Les algorithmes de chiffrement asymétrique sont en général assez coûteux en temps de calcul, c'est donc un algorithme de chiffrement symétrique qui va être utilisé pour chiffrer les échanges. Le **problème** est alors celui du **partage de la clef**.

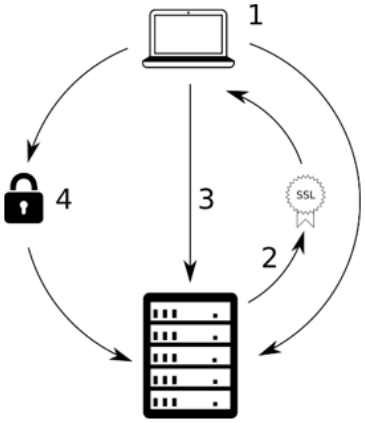
Une possibilité est que le client génère une clef, puis qu'il la partage avec le serveur en utilisant la clef publique du serveur pour la chiffrer. De cette façon, seul le serveur pourra déchiffrer la clef partagée et l'échange pourra continuer en utilisant cette clef et un algorithme symétrique.

Une autre possibilité est d'utiliser un autre protocole appelé l'« **Échange de clefs Diffie-Hellman**[7] » du nom de ses auteurs, en 1976. Cet algorithme permet aux deux participants de se mettre d'accord sur la clef partagée, sans que celle-ci ait à transiter, même cryptée, sur le réseau. Cet échange est d'ailleurs obligatoire dans les versions récentes du protocole et il est utilisé, entre autres, par le réseau Tor.

7. Explication complète, mais hors-programme : http://www.acrypta.com/telechargements/fichecrypto_205.pdf
Pour une explication simplifiée : https://www.youtube.com/watch?v=-PoJqTNY1eI

En résumé

L'implémentation du protocole SSL/TLS, et donc du protocole HTTPS, est assez complexe. Les algorithmes utilisés en pratique sont en quelque sorte négociés entre le client et le serveur lors de l'établissement de la connexion. En revanche, il faut être attentif à tous les détails pour que la sécurité soit assurée.



1. Demande d'initialisation d'une connexion par le protocole SSL.

2. Présentation du certificat :
- validité,
- signature par un tiers de confiance.

3. Transmission d'une clé de chiffrement unique, encodée avec la clef publique du serveur.

4. Déchiffrement de la clé de chiffrement par le serveur, à l'aide de sa clef privée :
- établissement de la connexion sécurisée.

Conclusion

Aujourd'hui, la cryptographie est partout, notamment dans les échanges informatiques pour protéger sa vie privée, sécuriser les échanges entre entreprises ou clients, acheter ou vendre en ligne, etc.

La recherche dans le domaine est permanente, qu'elle soit d'ordre :

- mathématique : nouveaux algorithmes, nouvelles attaques ;
- ou informatique : augmentation de la puissance de calcul, innovation technologique (ordinateur quantique…).

Pour ne citer qu'un seul exemple, le chiffrement DES (*Data Encryption Standard*), adopté comme standard en 1976, est devenu obsolète en 2001 à cause de l'augmentation de la puissance de calcul des ordinateurs.

Bibliographie

- Singh, S., *Histoire des codes secrets*, J.-C. Lattès (1999).
- Ubertret, G., *Initiation à la cryptographie*, Vuibert (2018).
- Cormen, T. H., Leiserson, C. E. *et al.*, *Introduction à l'algorithmique*, Dunod (2002).