Ce document est une compilation de ressources disponibles sur InterNet.

En rassemblant des informations éparpillées dans plusieurs publications ou sites distincts et en choisissant parmi les formulations (ou les traductions) qui semblent les plus intelligibles, il essaye de fournir une présentation générale et cohérente de la cryptographie.

Une grande partie provient directement de :

portail crypto de Wikipedia,
http://en.wikipedia.org/wiki/Portal:Cryptography

Sommaire :

Partie 1 : cryptographie moderne, notions essentielles

CRYPTOLOGIE = CRYPTOGRAPHIE + CRYPTANALYSE

CRYPTOGRAPHIE :
science qui utilise les mathématiques pour chiffrer et déchiffrer des données

CRYPTOSYSTEME :
algorithmes cryptographiques + plus clés + protocoles d'utilisation.

CRYPTANALYSE :
science de l'analyse des données chiffrées afin d'en retrouver la version 'claire'

La sécurité d'un cryptosystème ne doit reposer que sur le secret de la clef.
(Principe de Kerckhoffs)

Tous les systèmes de sécurité dépendent du fait de garder quelque chose secret.
Ce qui est gardé secret doit être ce qui est le moins coûteux à changer.
(Schneier)

Objectifs

La cryptographie est généralement utilisée pour sécuriser les informations.
On recherche en particulier les propriétés suivantes :

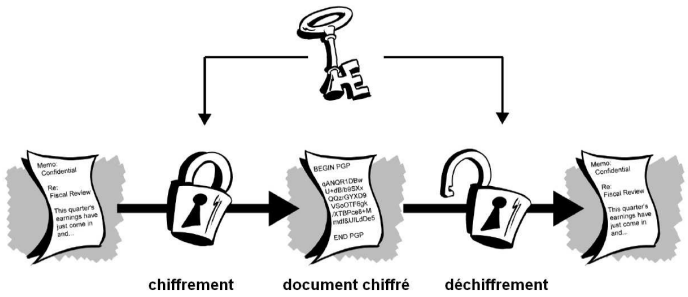
1. Confidentialité (ou secret) : seul le destinataire doit être capable de déchiffrer les messages. Il ne doit pas être possible à une autre personne d'obtenir des informations significatives à partir de ce qu'elle observe.
2. Intégrité : le destinataire doit être capable de déterminer si le message a été modifié [durant sa transmission].
3. Authentification : le destinataire doit être capable d'identifier l'expéditeur du message et de vérifier qu'il en est bien l'auteur.
4. Non-répudiation : en conséquence des points 2. et 3., l'expéditeur ne peut nier être l'auteur du message.
5. Non-rejouabilité (anti-replay) : le message ne peut-être envoyé plusieurs fois sans que le destinataire ne puisse le distinguer.
6. Preuve de délivrance : l'expéditeur doit avoir moyen de prouver que le destinataire à bien reçu le message.

La cryptographie fournie des mécanismes pour atteindre tous ces objectifs.
Cependant, certains ne sont pas toujours nécessaires, pratiques ou même souhaitable dans certains contextes. Par exemple, un expéditeur peut vouloir rester anonyme.

Cryptographie symétrique (à clé secrète)

La clé secrète est l'information devant permettre de chiffrer et de déchiffrer un message et à laquelle doit se réduire la sécurité de la communication. Il s'agit généralement d'un fragment binaire de longueur fixe, sans signification mathématique, produit à partir d'une phrase ou mot de passe.

En conséquence un même secret est partagé par l'émetteur et le destinataire du message. Le principal inconvénient réside dans la difficulté de stocker et distribuer les clés de manière fiables.

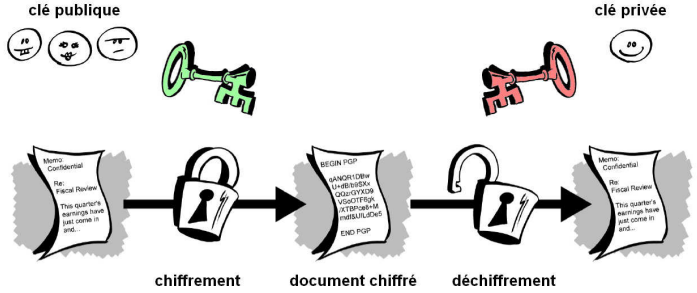


exemple : AES, Cast, Idea, Blowfish

Cryptographie asymétrique (à clé publique)

La cryptographie à clé publique ou asymétrique, est un système qui permet de communiquer en toute sécurité sans avoir besoin d'échanger au préalable une clé secrète. Ceci est réalisé grâce à une paire de clés, dont l'une est gardée secrète et l'autre diffusée publiquement. La relation (mathématique) qui les lie ne permet pas de déduire l'une à partir de l'autre.

Le chiffrement est réalisé en utilisant la clé publique du destinataire.
Seul le détenteur de la clé privée correspondante pourra déchiffrer le message.

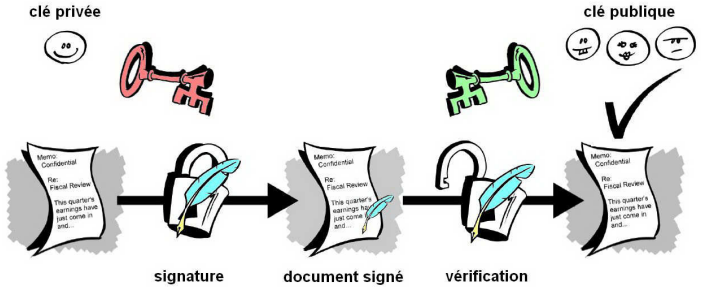


exemple : RSA, ElGamal

Signature numérique

Un des avantages majeurs de la cryptographie à clé publique est qu'elle procure au destinataire le moyen de contrôler l'origine d'un message, et également de vérifier que son contenu n'a pas été altéré. Ainsi, elle empêche l'expéditeur de contester ultérieurement avoir bien émis cette information.
Ces éléments sont au moins aussi importants que le chiffrement des données, sinon davantage.

La signature est produite en utilisant la clé privée, et vérifiable par tout-un-chacun au moyen de la clé publique.



En pratique, et afin d'éviter que la signature ne soit aussi volumineuse que le document original, elle est calculée à partir d'une empreinte de ce dernier.

Empreinte (condensé)

Il est possible de ‘résumer’ ou de ‘caractériser’ des données au moyen de fonctions dites de ‘hachage’. Pour ce faire, on convertit un grand ensemble (de taille variable) en un plus petit (de taille fixe), appelé ‘empreinte’.

document **empreinte (condensé)**

Dans un contexte cryptographique, il faut en particulier :

- ne rien permettre de connaître du message à partir de son empreinte,
- résister aux collisions, c’est-à-dire trouver deux messages distincts qui produiraient le même condensé
  (de par sa nature, tout algorithme de hachage génère des collisions),
- rendre très difficile la génération d’un message différent à partir d’une empreinte et d'un message donné

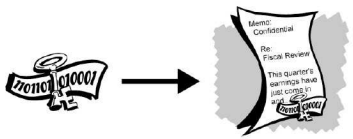
De par leur nature ‘irréversible’, les condensés sont souvent impliqués dans les stratégies de stockage des mots de passe. Pour identifier un utilisateur, il suffit de comparer l'empreinte du mot de passe d'origine (stocké) avec l'empreinte du mot de passe demandé. Le mot de passe n’est jamais stocké.

La plupart des algorithmes de hachage sont aussi d’excellents générateurs de valeurs pseudo-aléatoires. A ce titre ils interviennent dans la génération de clés secrètes à partir de phrases ou mots de passe.

exemple : SHA, RipeMD

Authentification de message

Lorsque la confidentialité d’un message n’est pas requise, il est néanmoins parfois utile de s’assurer qu’il ne puisse être modifié.
En protégeant son empreinte par une clé secrète, on produit un ‘code d’authentification de message’ qui protège contre toute altération extérieure
(au groupe qui partage la clé).



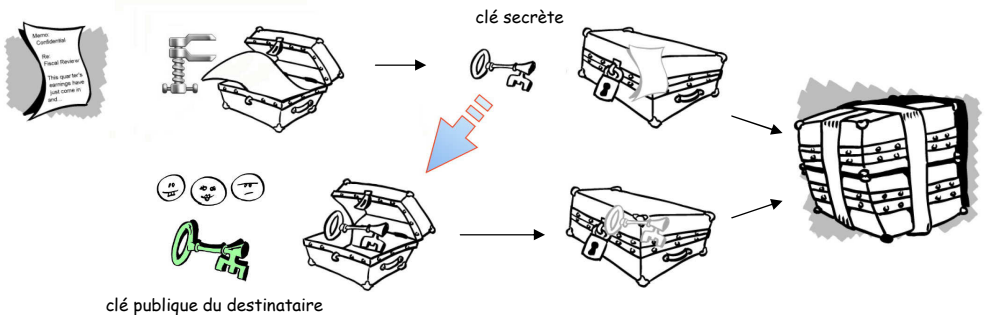
Toutefois, il ne s’agit pas de signature, car quiconque peut vérifier le message peut aussi produire le code de protection.

Cryptographie hybride – principe général

La cryptographie asymétrique est intrinsèquement lente à cause des calculs complexes qui sont nécessaires alors que la cryptographie symétrique est beaucoup plus rapide (environ 1000 fois).
Utilisés conjointement, la performance et la distribution de la clé sont améliorées sans aucun sacrifice sur la sécurité.

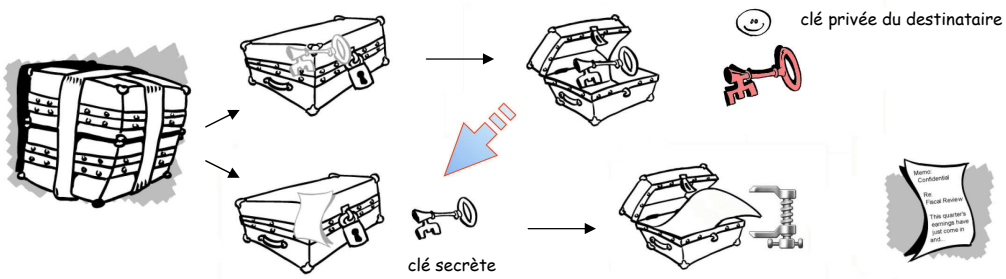
chiffrement :

1. compression des données : outre la diminution de taille, diminue la redondance du document original et augmente la résistance à la cryptanalyse
2. génération d'une clé de session (symétrique)
3. chiffrement du document avec la clé de session
4. chiffrement de la clé de session avec la clé publique du destinataire
5. transmission au destinataire de l'ensemble : document crypté + clé de session cryptée + ...



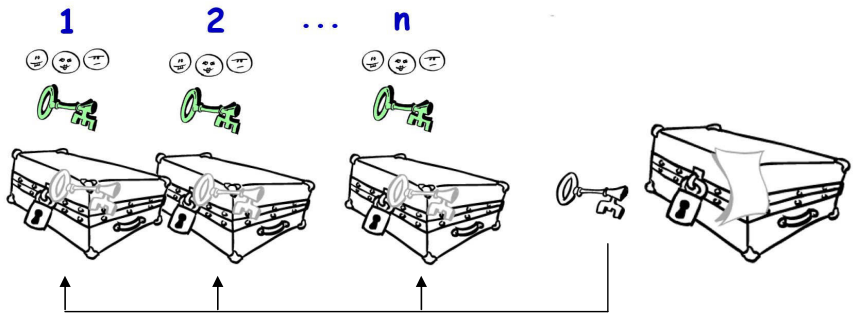
déchiffrement :

1. utilisation de la clé privée pour décrypter la clé de session (symétrique)
2. décryptage du document avec la clé secrète
3. décompression du document



Multiples destinataires :

Ce procédé offre un intérêt supplémentaire lorsqu'un même document doit être envoyé à plusieurs personnes : seule la clé de session est chiffré avec la clé publique de chaque destinataire, ce qui réduit d'autant le volume total du message, et garanti à chacun que tous lisent le même [seul] exemplaire.



Secret partagé

Le problème inverse est plus délicat : il est parfois souhaitable d'éviter qu'une information ne soit connue (déchiffrée) ou qu'une décision ne puisse être prise (signée) que par une seule personne.

Une solution réside dans la division du secret (secret splitting) : il est facile de découper une clé en fragments de manière telle qu'il faille l'apport de tous les détenteurs pour reconstituer la clé. L'inconvénient majeur de cette approche, c'est que si un seul des fragments vient à manquer, la clé (ou le secret) sera définitivement inaccessible.

Une alternative plus sophistiquée consiste à produire des fragments de façon à ce que seul un nombre minimum de morceaux soit requis pour reconstruire la clé (secret sharing / threshold scheme).
Imaginons par exemple qu'en l'absence du responsable d'un service de 7 personnes, un engagement financier puisse être lancé si au moins 3 membres du service donnent leur accord.
Il faudra alors partager la clé en 10 parties, en donner une à chaque personne (qui ne pourra donc l'utiliser qu'avec l'assistance de 2 autres collègues), et 3 au patron.

Autre exemple d'utilisation : sécuriser une information partagée entre plusieurs serveurs, mais qui doit rester accessible en cas de pannes ou d'indisponibilité d'un certain nombre d'entre eux.

exemple : SSSS (Shamir Secret Sharing Scheme)

Secret partagé vérifiable :

Un protocole de secret partagé est dit 'vérifiable' si chacun peut s'assurer qu'un fragment est bien un constituant du secret, et non pas une valeur binaire quelconque (si le distributeur est malhonnête par exemple).

Chiffrement et Signature

On peut bien sur chiffrer et signer à la fois. Reste à définir dans quel ordre. En application du principe général 'Only What is Seen Should be Signed' (ne doivent être signés que des données intelligibles), on signe d'abord et on chiffre ensuite, mais il existe des situations et des protocoles ou il est préférable d'agir autrement.

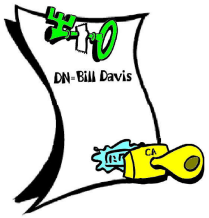
Certificat

Un problème, avec les cryptosystèmes à clés publiques, est que les utilisateurs doivent être constamment vigilants pour s'assurer qu'ils chiffrent leurs messages en utilisant la véritable clé du destinataire. Dans un environnement où l'on peut échanger des clés à travers des serveurs publics, les attaques utilisant une personne interposée sont un danger potentiel. Dans ce type d'attaque, un imposteur fournit une clé bidon portant le nom et l'identifiant d'utilisateur du destinataire réel des messages de l'utilisateur. Les données chiffrées pour – et interceptées par – le vrai propriétaire de cette fausse clé, seront tombées en de mauvaises mains.

On pourrait imaginer se limiter à utiliser les clés publiques remises physiquement, de la main à la main, par leur propriétaire. Mais la plupart du temps, on doit échanger des informations avec des gens qu'on n'a jamais rencontrés.

Une manière d'établir la validité d'une clé est de faire confiance à quelqu'un d'autre, qui aura lui-même effectué le processus de vérification.

Un certificat numérique est un document qui permet de s'assurer qu'une clé publique appartient bien à une personne ou une entité nommément désignée.



Il contient notamment les parties suivantes :

- la clé publique objet du certificat,
- des informations identifiant clairement le propriétaire de la clé, telles que son nom, son adresse électronique, ses coordonnées, …
- et enfin la signature numérique des données ci-dessus par la personne ou l'organisation qui atteste de leur véracité.

Dans le modèle hiérarchique X.509, une Autorité de Certification (A.C.), est tenue de s'assurer soigneusement qu'une clé appartient bien à son propriétaire supposé, avant d'apposer sa signature sur un certificat.
Quiconque a confiance en l'A.C. considérera automatiquement comme valides tous les certificats signés par cette A.C.

> Avoir confiance en une clé (être absolument sûr que la clé appartient réellement à une personne) n'est pas la même chose que d'avoir confiance dans le propriétaire de la clé.

PKI, X.509 et Confiance sur InterNet

« PKI n'est pas la mesure de sécurité la plus efficace jamais imaginé. »

« Malheureusement, les certificats X509 ont été conçus pour répondre à un problème qui n'a rien en commun avec ceux auxquels il faut faire face aujourd'hui. »

« Le résultat inévitable d'une telle complexité est que la plupart des normes sont à peine respectées dans les logiciels les mieux reconnus, et que, quand c'est le cas, elles le sont de manière si fragmentée et erratique qu'il devient dangereux de compter sur leurs fonctionnalités. »

"Why does X.509 do otherwise straightforward things in such a weird way ?"
- *Standards have been written by little green monsters from outer space in order to confuse normal human beings and prepare them for the big invasion.*
- *Someone tried to explain public key based authentication to aliens. Their universal translators were broken and they had to gesture a lot.*
- *They were created by the e-commerce division of the Ministry of Silly Walks.*

https://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt

« Plusieurs cas ont démontré que des autorités de certification commerciales n'avaient procédé à aucun contrôle (hormis celui d'avoir bien encaissé l'argent). »

« Jusqu'en 2011, il n'y a eu aucun cas de révocation de certificat d'une autorité commerciale publique, quelque soit le niveau de négligence dont elle a fait preuve […] Et lorsque les choses se sont par la suite aggravées, il a été intéressant de constater que les éditeurs de navigateurs ne faisaient pas confiance à leur propre PKI et préféraient pousser en catimini des mises à jours de leur produit contenant des listes noires de certificats codées en 'dur'. »

https://blog.torproject.org/blog/detecting-certificate-authority-compromises-and-web-browser-collusion

Sans compter les CA qui ont trahis la confiance qui leur avait été accordée en choisissant délibérément de vendre des dispositifs générant des certificats intermédiaires permettant d'espionner les communications TLS.

http://blog.cryptographyengineering.com/2012/02/how-to-fix-internet.html

Considérations sur la sécurité et la vulnérabilité

La cryptographie peut être forte ou faible.
La force de la cryptographie est mesurée par le temps et les ressources qui seraient nécessaires pour retrouver le texte clair.

La sécurité des données chiffrées est entièrement dépendante de deux choses : la force de l'algorithme cryptographique et le secret de la clé.

Pas de réelle protection [du message] sans vigilance, rigueur et secret.

- Les techniques cryptographiques ne protègent les données que lorsqu'elles sont chiffrées – une violation directe de la sécurité physique peut encore compromettre les données en clair ou les informations écrites ou orales.
(si on sait que vous possédez le secret, il est plus facile de faire pression sur vous, voire d'attenter à vous)
- Analyse de trafic : pour l'attaquant, c'est comme examiner votre facture de téléphone pour voir qui vous appelez, quand et pour combien de temps, quand bien même le contenu réel des appels demeure inconnu.
- Attaques 'Tempest' : la détection à distance des signaux électromagnétiques émis par l'ordinateur. Cette coûteuse et parfois laborieuse attaque est probablement toujours moins coûteuse que l'attaque cryptanalytique directe.

http://www.schneier.com

« Security is a chain; it's only as strong as the weakest link.
The security of any CA-based system is based on many links and they're not all cryptographic. People are involved ».

Les clés plus longues ne signifient pas toujours plus de sécurité.
Comparez l'algorithme cryptographique au verrou de votre porte d'entrée. […]
Des cambrioleurs n'essayent pas toutes les clés (attaque systématique); la plupart ne sont pas assez intelligents pour crocheter la serrure (attaque cryptographique contre l'algorithme). Ils fracassent les fenêtres, donnent des coups de pieds dans les portes, se déguisent en policiers, ou bien dévalisent les détenteurs de clés avec une arme. Un groupe de voleurs en Californie mettait en défaut les systèmes de sécurité en attaquant les murs à la tronçonneuse. Contre ces attaques, de meilleures serrures ne sont d'aucun secours.

> La cryptographie forte est très puissante quand elle est bien faite, mais ce n'est pas une panacée. Se focaliser sur les algorithmes cryptologiques tout en ignorant les autres aspects de la sécurité revient à défendre votre maison, non pas en dressant une barrière autour, mais en plantant un seul immense poteau devant en espérant que votre adversaire va juste s'y heurter. Les attaquants intelligents se contentent de contourner les algorithmes.
> […]

Nous n'avons pas à essayer toutes les clés possibles, ni même à trouver une faille dans les algorithmes; nous exploitons les erreurs de conceptions, les erreurs de réalisations, et les erreurs d'installations. […] les vieilles fautes classiques qui se répètent indéfiniment. […]

> Un système cryptologique ne peut pas être plus solide que les éléments sur lesquels il repose. De la même façon qu'il est possible de construire une structure faible avec de solides matériaux, il reste possible de construire un système cryptologique faible en utilisant des algorithmes et des protocoles forts. Nous trouvons souvent des systèmes qui "annulent la garantie" de leur cryptologie en ne l'utilisant pas correctement.

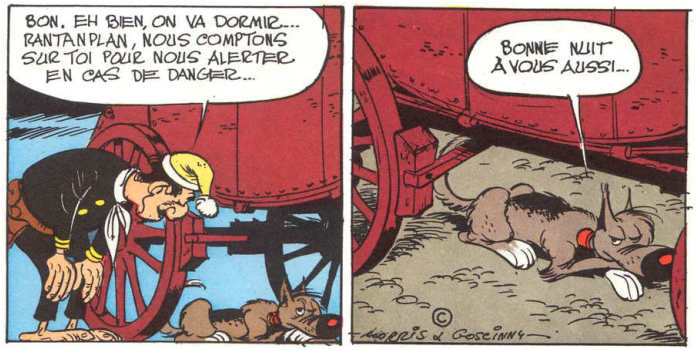
[…]
Beaucoup de systèmes font défaut suite à des erreurs de réalisation. Quelques systèmes ne garantissent pas que le texte clairement lisible soit détruit après avoir été chiffré.

[…]
Beaucoup de systèmes sont cassés car ils reposent sur des mots de passe choisis par l'utilisateur. Laissés à eux-mêmes les utilisateurs ne choisissent pas des mots de passe forts. Et s'ils sont forcés de le faire, ils ne peuvent s'en souvenir.
Beaucoup des attaques les plus intéressantes visent le modèle sous-jacent de la confiance dans le système : à qui ou à quoi fait-on confiance, de quelle façon, et à quel degré ? […] Souvent, un système sera conçu pour un modèle de confiance, et implanté avec un autre.

[…]
Beaucoup de systèmes logiciels font de mauvaises hypothèses quant à la confiance à accorder aux ordinateurs sur lesquels ils tournent; Ces programmes peuvent souvent être cassés par des vers qui reniflent les mots de passe, lisent les textes clairs, ou contournent de toute autre façon les mesures de sécurité.

[…]
Cela ne prend que quelques jours pour reconstituer par rétro-analyse l'algorithme à partir du code exécutable. Le système pour les disques DVD prenait un algorithme faible et le rendait encore plus faible…

[…]
Même quand un système est sécurisé sous réserve d'une utilisation correcte, ses utilisateurs peuvent mettre en danger la sécurité par accident ou négligence. L'exemple le plus classique en est l'utilisateur qui donne son mot de passe à ses collègues pour qu'ils dépannent des problèmes en son absence.



[Partie 2 : cryptographie moderne, compléments]

Cryptographie symétrique – compléments

On distingue deux grandes catégories de chiffrement symétrique :

- le chiffrement par flots (stream cipher)
- le chiffrement par bloc (block cipher)

Les méthodes 'par flot' chiffrent les bits ou octets individuellement en appliquant une transformation qui varie avec le temps. Par contraste, les méthodes 'par bloc' opèrent une transformation fixe et répétitive sur un bloc complet de données.

Chiffrement par flot

Un chiffrement par flot traite des données de longueur quelconque sans avoir besoin de les découper. Il se présente souvent sous la forme d'un générateur de nombres pseudo-aléatoires avec lequel on opère un XOR (eXclusive OR) entre un bit à la sortie du générateur et un bit provenant des données.

Son principe repose sur celui du 'masque jetable', également appelé chiffre de Vernam, (One-Time Pad). C'est un algorithme inventé par Gilbert Vernam en 1917. Ce chiffrement est le seul qui soit théoriquement impossible à casser, mais il présente d'importantes difficultés d'utilisation pratique :

- une clé ne doit être utilisée qu'une seule fois (d'où le titre de masque jetable),
- la clé doit être de la même taille que le message,
- la clé doit être aussi aléatoire que possible.

Si ces trois règles sont respectées, il est théoriquement impossible de déchiffrer le message chiffré. L'argument est très simple : si on ne connaît que le texte chiffré, et que toutes les clés sont équiprobables alors tous les textes clairs sont possibles et avec la même probabilité puisqu'il y a bijection, une fois le chiffré fixé, entre clés et textes clairs. Même une attaque par force brute ne donnera aucune information.

Ce type d'impossibilité, qui ne repose pas sur la difficulté du calcul, est appelé sécurité sémantique. Autrement dit, le système du masque jetable est sémantiquement (ou inconditionnellement) sûr. Shannon a prouvé (1949) que le seul algorithme à offrir cette sécurité est le masque jetable.
Dans la pratique, les contraintes théoriques étant trop difficiles à respecter, on utilise une clé plus petite (par exemple de128 bits) à partir de laquelle on génère un flux de clé pseudo-aléatoire, ce qui a pour conséquence de diminuer la sécurité du système.

> Utiliser des algorithmes pour créer des nombres aléatoires, sachant ce qu'est un algorithme, peut paraître assez douteux. En pratique, il est possible de limiter les conséquences négatives de l'utilisation d'une méthode algorithmique pour générer des nombres [pseudos-]aléatoires.

Génération du flux de clé

Registre à décalage (shift register) : en informatique et en électronique, un registre à décalage est une mémoire de taille fixe dans laquelle les bits sont décalés à chaque coup d'horloge (dimension temporelle).
Registre à décalage avec rétroaction linéaire (LSFR, Linear Feedback Shift Register) : il s'agit d'un registre à décalage dans lequel les bits en sortie subissent une série de transformations et d'opérations avant d'être réinsérés à l'entrée du registre.

Un grand nombre de générateurs sont basés sur des LSFR, parce qu'ils sont faciles à réaliser matériellement, qu'ils peuvent produire des séquences ayant de bonnes propriétés statistiques, de larges périodes, et qu'ils sont facilement analysable algébriquement.
Malheureusement, ils sont trop prédictibles, c'est pourquoi il ne faut pas utiliser un registre initialisé directement avec la donnée secrète, car une connaissance même très partielle de la suite produite permet de retrouver toutes les informations souhaitées (algorithme de Berlekamp-Massey).

Dans la pratique, il faut casser la linéarité du résultat en filtrant la sortie d'un registre, en utilisant une fonction non linéaires sur la sortie de plusieurs registres combinés, ou en utilisant un ou plusieurs registres pour contrôler l'horloge d'un ou plusieurs autres.

Utilisation

Les chiffrements par flot sont souvent utilisés dans des situations ou la taille du message ne peut-être déterminée à l'avance, par exemple quand les données doivent être traitées aussitôt qu'elles sont reçues (télécommunications). Dans de telles situations, si on devait utiliser un chiffrement par bloc, il faudrait sacrifier l'efficacité ou compliquer la conception. Comme ils ne propagent pas ou peu les erreurs, ils sont d'un intérêt particulier dans les situations de transmission difficiles.

Les chiffrements par flots ne sont pas adaptés pour sécuriser les informations d'un disque (gaspillage d'espace pour stocker l'état initial de chaque secteur du disque).

Chiffrement synchrone : lorsque le flux de clé est généré indépendamment des données du message et du chiffré résultant.
Chiffrement asynchrone (auto-synchronisé) : un partie fixe du chiffré précédent est impliqué dans la génération du flux de clé.

Les chiffrements par flots sont plus rapides que les chiffrements par blocs, néanmoins ils sont susceptibles de présenter des failles importantes s'ils sont mal conçus (initialisation)

RC4 est la méthode de chiffrement par flot la plus répandue.
Elle a été conçu par Ron Rivest en 1987 (RC = Rivets Cipher ou Ron's Code) et est mise en œuvre dans des protocoles très populaires tels que SSL. Cependant certaines formes d'utilisation résultent en des systèmes très peu sûr (dont WEP).

SEAL (Software-optimized Encryption Algorithm) est un chiffrement par flot proposé en 1993 dont la fonction pseudo-aléatoire est basé sur SHA-1.

Chiffrement par bloc

Les chiffrements par bloc sont plus répandus que ceux qui traitent des flots de données. La principale différence vient du découpage des données en blocs de taille généralement fixe (souvent une puissance de deux comprise entre 32 et 512 bits). Les blocs sont ensuite chiffrés les uns après les autres au moyen de la clé et d'un mode opératoire.
Certains modes opératoires capables de travailler sur un sous-ensemble réduit du bloc rendent le chiffrement par bloc assimilable à un chiffrement par flot (moins performant).

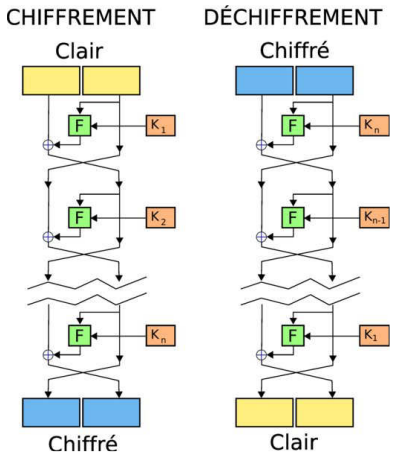
Les chiffrements par bloc sont regroupés en fonction de leur 'schéma' ou 'structure' : un chiffrement « par produit » est un type répandu de chiffrement par bloc qui exécute de manière itérative un certain nombre de transformations simples : des substitutions, des permutations et des opérations d'arithmétique modulaire.
Chaque itération est un 'tour'.
Un chiffrement « par produit » qui n'utilise que des substitutions et des permutations est nommé « réseau de substitutions-permutations ».

Une des variantes les plus répandue est le « réseau de Feistel ». La construction de Lai-Massey se différencie de ce dernier par l'utilisation d'opérations arithmétiques telles que l'addition et la multiplication.

Dans sa version équilibrée, un réseau de Feistel traite les données en deux parties de taille identique. A chaque tour, les deux blocs sont échangés puis un des blocs est combiné avec une version transformée de l'autre bloc. Ces opérations sont spécifiques à chaque algorithme. Chaque tour utilise une clé intermédiaire, en général tirée de la clé principale via une génération appelée 'key schedule'.

Chaque tour applique plusieurs transformations sur les données provenant du tour précédent :

- permutation des bits via des P-Boxes
- fonction non-linéaire (substitution) avec des S-Boxes
- mixage linéaire en utilisant la fonction XOR
- application de la clé du tour (intégrée dans une fonction ou via un XOR)



Toutes ces opérations sont facilement réalisables à la fois au niveau logiciel et matériel.

P-Box (permutation box) : table de permutation. Elle indique comment échanger les éléments d'une structure. Une P-Box contribue à la «diffusion» en mélangeant les données et en améliorant l'effet avalanche.

Une S-Box (substitution box) contribue à la «confusion» en rendant l'information originale inintelligible. Les S-Boxes permettent de casser la linéarité de la structure de chiffrement.

Les tables sont souvent définies à l'avance mais il arrive parfois qu'elles soient générées par l'algorithme (par exemple dans Blowfish).

La confusion et la diffusion sont les deux propriétés fondamentales des algorithmes de chiffrement identifiés par Shannon en 1949.

La confusion est le fait de rendre la relation entre la clé et le document crypté aussi compliqué et obscure que possible.
La diffusion est le fait de 'dissiper' les redondances statistique du document original dans le document crypté.

L'effet avalanche est une propriété recherchée dans les fonctions cryptographiques. Elle provoque des modifications de plus en plus importantes au fur et à mesure que les données se propagent dans la structure de l'algorithme.
De ce fait, en perturbant un seul bit en entrée, on obtient idéalement une sortie totalement différente, d'où le nom de ce phénomène.
L'effet avalanche permet de rendre l'inversion de la fonction plus difficile grâce à ses propriétés chaotiques (s'il est bien conçu).

Key Schedule

Le 'Key Schedule' (préparation des clés) consiste à générer des sous-clés à partir de la clé principale pour un algorithme de chiffrement par bloc. Le terme est également employé dans le cadre des fonctions de hachage cryptographiques même si la notion de clé est ici différente (la clé provenant en général du message à hacher).

Certains chiffrements ont des algorithmes de préparation relativement simples. L'algorithme TEA se contente de découper la clé de 128 bits en quatre morceaux de 32 bits qui sont utilisés dans les différentes rondes.

DES utilise sa clé de 56 bits qui est coupée en deux. Chaque partie de 28 bits est traitée séparément. À chaque tour, les deux sous-clés subissent une rotation vers la gauche par un ou deux bits (selon le tour). Des sous-clés de 48 bits sont ensuite extraites en prenant 24 bits dans chaque clé de 28 bits. Les 24 bits retenus varient selon les tours de telle façon que chaque bit soit utilisé dans approximativement 14 des 16 sous-clés employées dans les 16 tours.

Les chiffrements plus récents utilisent généralement des procédures plus élaborées, à base de fonctions à sens unique par exemple, pour générer des clés ‘élargies’. Certains, comme Rijndael (AES) et Blowfish impliquent leur propre algorithme de chiffrement interne dans cette ‘expansion’.

Lars Knudsen et Mathiassen ont montré en 2004 que le key schedule joue un rôle primordial dans la résistance aux cryptanalyses de type linéaire ou différentielle. Les générations de sous-clés, lorsqu'elles sont bien conçues, permettent d'arriver rapidement à l'absence de biais statistiques potentiellement exploitables par la cryptanalyse.

Clés faibles

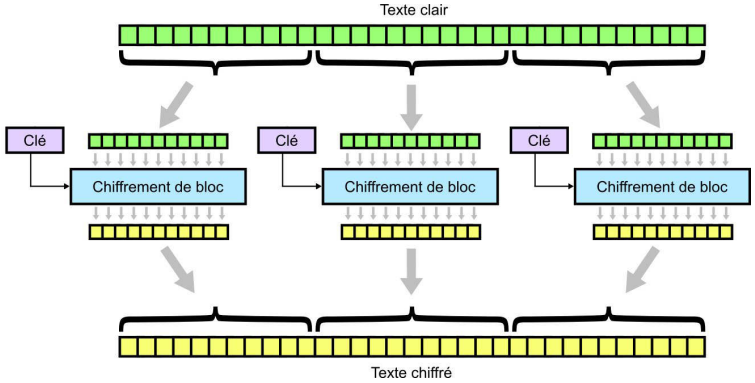
En cryptologie, une clé faible est une clé telle que son utilisation dans une procédure de chiffrement produit un comportement indésirable. Les clés faibles sont généralement peu nombreuses par rapport à l'ensemble des clés possibles. De ce fait, la probabilité d'obtenir une clé faible avec des chiffrements modernes étant très petite (lors d'une génération de clé aléatoire) il est peu probable que cela induise un problème de sécurité. On considère qu'il n'est pas souhaitable qu'un chiffrement ait des clés faibles, cependant si elles sont connues ou facilement identifiables, ce n’est pas un inconvénient majeur.

Mode d’opération

Historiquement, les modes d'opération ont été abondamment étudiés pour leur propriétés de propagation d'erreurs lors de divers scénarios de modification de données durant le chiffrement. Les développements suivants ont considéré que la protection de l'intégrité était un objectif à atteindre par des moyens complètements différents. Mais aujourd'hui il existe des modes d'opérations qui associent chiffrement et authentification de manière efficace.

Dictionnaire de codes : «Electronic codebook» (ECB)

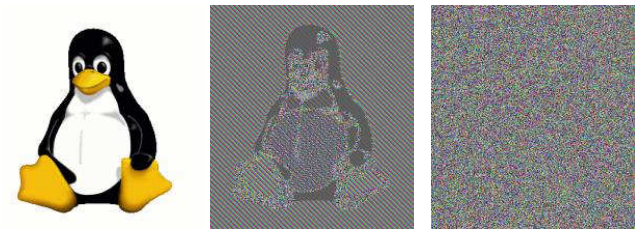
Il s'agit du mode le plus simple. Le message à chiffrer est subdivisé en plusieurs blocs qui sont chiffrés séparément les uns après les autres.



Le gros défaut de cette méthode est que deux blocs avec le même contenu seront chiffrés de la même manière, on peut donc tirer des informations à partir du texte chiffré en cherchant les séquences identiques. On obtient dès lors un «dictionnaire de codes» avec les correspondances entre le clair et le chiffré d'où le terme ‘codebook’.

Ce mode est pour ces raisons fortement déconseillé dans toute application cryptographique. Le seul avantage qu'il peut procurer est un accès rapide à une zone quelconque du texte chiffré et la possibilité de déchiffrer une partie seulement des données. Mais un mode bien plus sûr basé sur un compteur permet également ces accès aléatoires et les déchiffrements partiels.

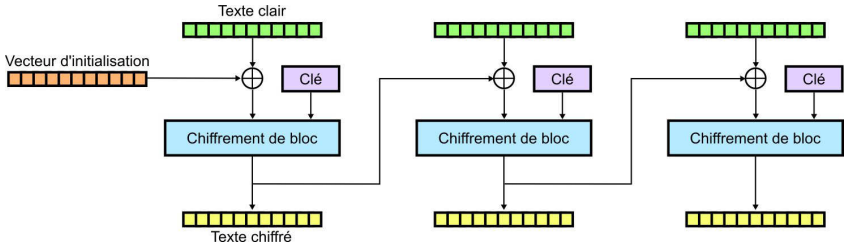
Appliqué à une image bitmap, on obtient un exemple saisissant de ce que ce mode peut révéler sur les motifs du document original, alors que le résultat obtenu avec un autre mode ne peut être distingué d’un bruit aléatoire :



De par sa nature, ECB est sensible à des «attaques par répétition» qui consistent à réinjecter dans le système des données identiques à celles interceptées auparavant.

Enchaînement des blocs : «Cipher Block Chaining» (CBC)

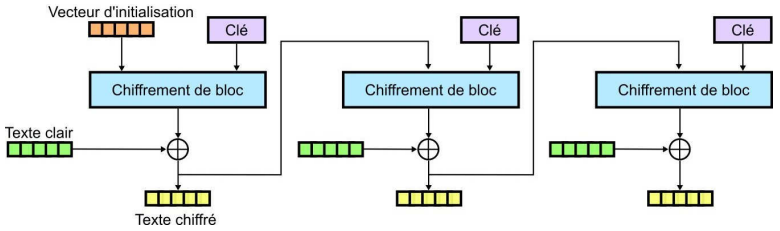
Dans ce mode, on applique sur chaque block un ‘OU exclusif’ avec le chiffrage du bloc précédent avant qu’il soit lui-même encrypté. De plus, afin de rendre chaque message unique, un vecteur d’initialisation est utilisé.



Chiffrement à rétroaction : «Cipher Feedback» (CFB, CFB-1, CFB8, …)

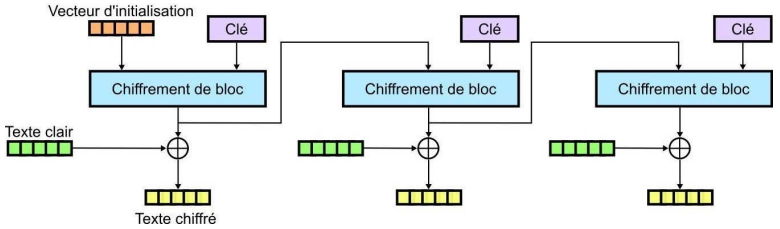
Ce mode et les suivants sont assimilables à un chiffrement par flux.
Ils génèrent un flux de clés qui est ensuite appliqué au document original.
Les données peuvent être chiffrées par morceaux de longueur inférieure au bloc (CFB-r, r bits feedback avec généralement r égal à 1, 8, …)

Dans ce mode, le flux de clé est obtenu en encryptant le précédent bloc chiffré.



Chiffrement à rétroaction de sortie : «Output Feedback» (OFB)

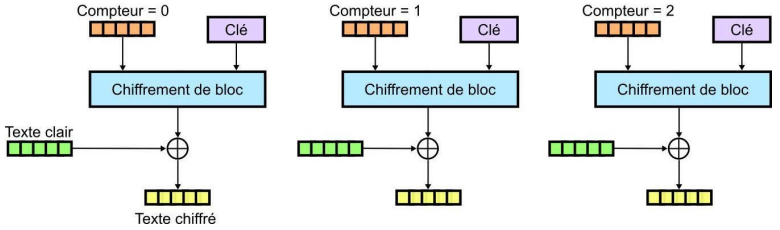
Variante du mode précédent, parfois appelée ‘internal feedback’
Dans ce mode, le flux de clé est obtenu en encryptant le précédent flux de clé.



(ISO version : OFB full-feedback, versions FIPS : OFB r-bits feedback, moins sécurisées, peu conseillées)

Chiffrement basé sur un compteur (CTR)

Dans ce mode, le flux de clé est obtenu en chiffrant les valeurs successives d’un compteur. C’est une simplification du mode OFB ou le compteur joue le rôle du feedback.



Vecteur initial - IV

Tous les modes (à l’exception d’ECB) requièrent un ‘vecteur d’initialisation’.
C’est un bloc de données aléatoires pour démarrer le chiffrement du premier bloc et fournir ainsi une forme de hasard indépendant du document à chiffrer.
Il n’a pas besoin d’être lui-même chiffré lors de la transmission, mais il ne doit jamais être réemployé avec la même clé.

Rembourrage (padding)

En mode CBC notamment, il est nécessaire de rembourrer le dernier bloc. Plusieurs schémas sont proposés à cette fin, dont le plus simple consiste à compléter le bloc avec des zéros binaires.

Les modes CFB, OFB et CTR n’ont nul besoin de mesures spéciales pour gérer les messages qui ne sont pas de longueur multiple de la taille du bloc, puisque le ‘OU’ logique est effectué après exécution du chiffrement. Ils peuvent produire un chiffré de longueur exactement égale à celui du document original.

Propagation d’erreur

Aucun des modes ci-dessus ne protège l’intégrité du message.
Pour ce besoin, appliquer sur le message et son IV un code d’authentification (HMAC).

- En mode ECB, une erreur dans un bloc chiffré résulte en un bloc défectueux du document original.
- En mode CBC, c’est 2 blocs qui seront affectés.
- En mode CFB, une erreur sur N bits chiffrés affectera la même portion en sorti, ainsi que la suite tant que le dernier bit en erreur ne sort pas du bloc CFB en entré.
- En mode OFB, un bit en erreur provoque seulement un bit en erreur lors du déchiffrement.

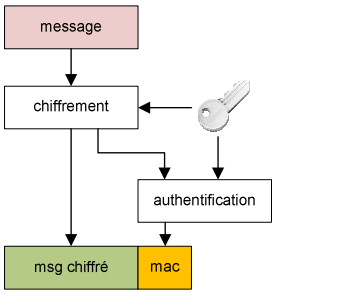
Autres modes

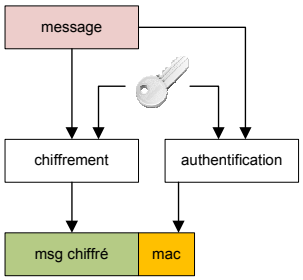
Il existe d’autres modes adaptés à des utilisations particulières, par exemple pour le chiffrement des disques : LRW, XEX, XTS, …

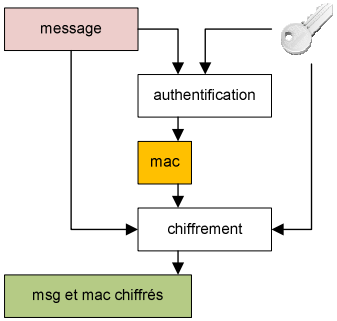
Modes avec authentification - Authenticated Encryption (AE) / with Associated Data (AEAD)

L'observation a montré que les modes de chiffrement et les codes d'authentifications sont des abstractions de trop bas niveau, et que les constructions qu'on peut en tirer devaient être étudiées et standardisées.

De fait, le chiffrement utilise une clé et le calcul d'un code d'authentification aussi. Dans certains cas, utiliser la même clé pour ces deux opérations peut engendrer des failles. Des considérations identiques peuvent concerner le vecteur d'initialisation.
Et [théoriquement], un code d'authentification peut laisser fuir des informations sur les données à chiffrer. Ce qui donne lieu à débat : chiffrer d'abord puis calculer un code ou l'inverse ?







Différentes combinaisons :

Encrypt-then-MAC (EtM),

Encrypt-and-MAC (E&M),

MAC-then-Encrypt (MtE)

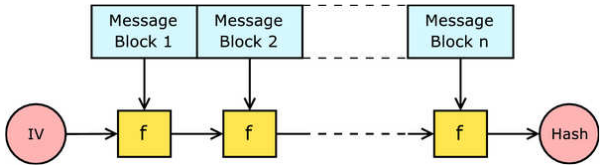
Enfin entrent aussi en considération l'efficacité de l'algorithme, les possibilités d'optimisation (parallélisme), les fonctionnalités spécifiques (pouvoir mettre à jour rapidement le code d'authentification suite à une petite modification des informations (chiffrement de disque), …

C'est en réponse à ces différentes problématiques qu'ont été proposées les modes OCB, CCM, EAX et GCM.

Données Associées

Il s'agit d'informations liées au message, non chiffrées, mais authentifiées (au moyen des données chiffrées). Un exemple d'utilisation est le transport d'entêtes dans un contexte réseau. Les équipements intermédiaires ne possédant pas la clé de déchiffrement doivent quand même pouvoir les lire et vérifier.

Condensé (complément)



La valeur initiale – IV – est spécifique à l'algorithme de hachage.
Pour pouvoir calculer des condensés de flux de taille arbitraire, les données doivent être 'capitonnées/rembourrés' afin d'atteindre une taille qui soit un multiple de la longueur du bloc. La méthode la plus répandue consiste à ajouter un bit '1', suivit de zéros, puis de la taille du message original sous la forme d'un entier binaire de 64 bits. Ce schéma protège contre les collisions entre messages très similaires et de longueur peu différentes.

La plupart des fonctions de hachage produisent des empreintes radicalement différentes si l'entrée est légèrement modifiée. Toutefois, il existe des fonctions de hachage qui tolèrent une certaine marge d'erreur. Elles sont particulièrement utiles pour hacher des données qui peuvent subir des perturbations comme les sons ou encore les images. Il est à noter que ces techniques, alliées au tatouage numérique, sont principalement destinées à lutter contre le piratage, la contrefaçon et les autres infractions liées au copyright. Elles sont également intéressantes pour gérer des bases de données et trouver rapidement des échantillons présentant de fortes similitudes.

SHA-1 (Secure Hash Algorithm) est une fonction de hachage cryptographique conçue par la NSA. Elle produit un condensat de 160 bits (soit 20 octets) .
Voici le condensé (en hexa) obtenue avec la phrase :

```
"Wikipedia, l'encyclopedie libre et gratuite"
  c18cc65028bbdc147288a2d136313287782b9c73
```

En modifiant un seul caractère, le condensé change radicalement :

```
"Wikipedia, l'encyclopedie libre et gratuitE"
  3981d4f03f2732e582f629ba27af75a213cfc7f3
```

> Les attaques connues (depuis 2005) ne concernent que des collisions quelconques. C'est-à-dire que l'on peut trouver deux messages au contenu aléatoire qui produisent la même signature. Par contre, à partir d'une signature donnée, il est impossible de forger un second message qui génère la même valeur. Or, c'est ce type d'attaque qui pourrait mettre en péril les applications cryptographiques.

Sel – production [/dérivation] de clé symétrique à partir d'un mot de passe (PKCS#5)

Si deux utilisateurs décident d'utiliser le même mot de passe alors le condensé obtenu sera identique. Cette faille pourrait être potentiellement utilisée lors d'une attaque par dictionnaire. Pour contrer ce type d'attaque, on ajoute une composante aléatoire lors de la génération initiale de l'empreinte. Cette composante, aussi appelée «sel», est souvent stockée en clair. On peut simplement utiliser l'heure de l'attribution du mot de passe ou un compteur qui varie selon l'utilisateur.

PKCS#5 préconise un minimum de 1000 itérations de la fonction utilisée dans la génération de la clé.

Le sel et le nombre d'itérations n'ont pas besoin d'être protégés, et doivent être connus lors de l'utilisation du mot de passe.

Quelques algorithmes de chiffrement symétriques comparés

		3xDES	IDEA	BLOWFISH	CAST 5	TWOFISH	AES
	Publication	1978	1991	1993	1996	1998	1998
	Taille de bloc (bits)	64	64	64	64	128	128
	Taille de clé	168 (112 effectif)	128	32-448 (par 8)	40-128 (par 8)	128, 192, 256	128, 192, 256
	Structure	Feistel	substitution / permutation	Feistel	Feistel	Feistel,	substitution / permutation
	Nbr. de Tours	3x16	8 ½	16	12 ou16	16	10, 12 ou 14
	Clé faible		OUI	OUI	NON		NON
LOGICIELS	PGP 9.0	OUI	OUI		OUI	OUI	OUI
	GnuPG	OUI	OUI	OUI	OUI	OUI	OUI
	OpenSSL	OUI	OUI	OUI	OUI		OUI

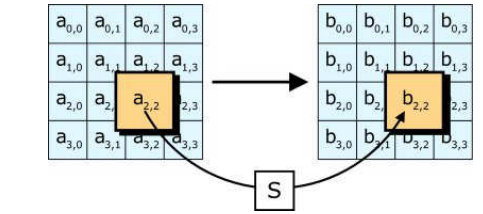
CAST apparaît comme étant exceptionnellement bien conçu, par des gens jouissant d'excellentes réputations dans ce domaine. La conception est fondée sur un nombre d'assertions formellement démontrables qui donnent de bonnes raisons de penser qu'il exige une recherche exhaustive pour casser sa clé de 128 bits. CAST n'a pas de clés faibles ou semi faibles. Il existe de solides arguments permettant de penser que CAST est complètement immunisé aussi bien contre la cryptanalyse linéaire que différentielle, les deux formes de cryptanalyse les plus puissantes dans la recherche publique.

AES - Advanced Encryption Standard

AES, est un algorithme de chiffrement symétrique, choisi en octobre 2000 par le NIST pour être le nouveau standard de chiffrement pour les organisations du gouvernement des États-Unis. Il est issu d'un appel d'offre international lancé en janvier 1997 pour remplacer son prédécesseur DES.
Parmi les 15 propositions retenues, c'est Rijndael, du nom de ses deux concepteurs Joan Daemen et Vincent Rijmen (tous les deux de nationalité belge) qui a été choisi.

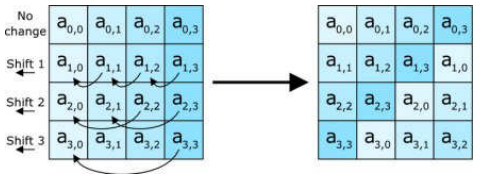
Son utilisation est très pratique car il consomme peu de mémoire et n'étant pas basé sur un schéma de Feistel, sa complexité est moindre et il est plus facile à implémenter.

Chaque tour est constitué de quatre étapes :



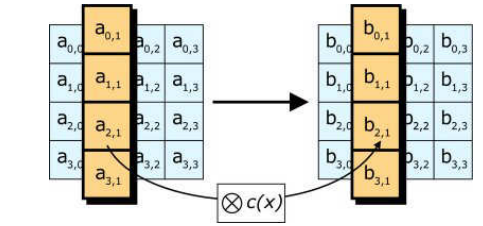
Substitution

Chaque octet est remplacé par un autre en fonction d'une table



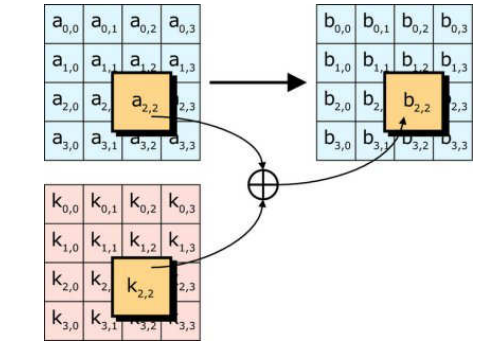
Transposition

Chaque ligne est décalé cycliquement en fonction du numéro de ligne



Mélange

On applique une transformation linéaire, fonction d'une matrice auxiliaire.



Cycle de clé

Chaque octet est combiné avec une sous-clé, elle-même dérivée de la clé principale (key schedule)

L'AES n'a pour l'instant pas été cassé et la recherche exhaustive («force brute») demeure la seule solution. Rijndael a été conçu de manière à rendre des méthodes classiques comme la cryptanalyse linéaire ou différentielle très difficiles.

Cryptographie asymétrique – compléments

Fondements théoriques

Les crypto-systèmes asymétriques reposent sur des fonctions mathématiques ‘à sens unique’ c'est-à-dire facile à calculer mais difficile à inverser.

On n’a aucune certitude sur le fait que de telles fonctions existent, mais plusieurs classes de fonctions sont supposées candidates :

- celles s’appuyant sur la factorisation des nombres entiers
- celles reposant sur le problème du logarithme discret
  - modulo un nombre premier
  - appliquées aux courbes elliptiques

Les fonctions «à sens unique avec trappe» représentent une variation pour laquelle une information particulière (la trappe) permet l’inversion.
C’est sur ce schéma que repose le système le plus répandu : RSA.

Les fonctions basées sur le problème du logarithme discret - problème inverse de l’exponentiation modulo n - ne sont pas des fonctions avec trappe, car les groupes considérés n'ont pas de trappes connues.

Horodatage (TimeStamp)

Un horodatage peut simplement être produit en intégrant une information temporelle à une empreinte de document, l'ensemble étant signé par une autorité adéquate qui garanti de ce fait que le document n'a pu être produit postérieurement à cette date.
Afin d'augmenter la confiance et réduire la vulnérabilité d'un tel procédé, on peut faire appel à plusieurs autorités différentes.

DSA

Le ‘Digital Signature Algorithm’, plus connu sous le sigle DSA, est un algorithme de signature numérique standardisé par le NIST aux États-Unis, du temps où le RSA était encore breveté. Cet algorithme fait l’objet de la spécification DSS pour Digital Signature Standard adoptée en 1993 (FIPS 186). Une révision mineure a été publiée en 1996 (FIPS 186-1) et le standard a été amélioré en 2002 dans FIPS 186-2. Cette dernière version de la spécification impose SHA-1 comme méthode d’empreinte et une longueur de clé de 1024 bits.
La version FIPS 186-3 spécifie l’emploi de condensés plus longs, (jusqu'à SHA-512) et des clés de longueur allant jusqu'à 15360 bits.

DSA qui repose sur le problème du logarithme discret est sensiblement identique au schéma de signature El Gamal.

DSA est couvert par un brevet que le NIST a mis gratuitement à la disposition du reste du monde. Ce brevet est cependant contesté par Dr. Claus P. Schnorr, lui même concepteur d'un cryptosystème portant son nom.

NIST = National Institute of Standards and Technology
FIPS = Federal Information Processing Standards

Signature Aveugle

En cryptographie, une signature aveugle est une signature effectuée sur un document qui a été masqué avant d'être signé, afin que le signataire ne puisse prendre connaissance de son contenu. De telles signatures sont donc employées lorsque le signataire et l'auteur du document ne sont pas la même personne. La signature (aveugle) résultante peut être publiquement vérifiée avec le document original (démasqué) comme toute signature numérique. Certains protocoles, dont RSA, permettent même d'annuler directement l'effet du masque sur la signature, avant la vérification.

On compare souvent la signature aveugle au fait de soumettre à la signature une enveloppe fermée contenant un document. Le signataire ne peut donc en lire le contenu, mais un tiers pourra plus tard s'assurer que la signature est valide, dans les limites du respect du protocole.

Les protocoles de vote électronique ou les systèmes de porte-monnaie électronique sont des exemples typiques d'utilisation des signatures aveugles.

RSA

RSA a été décrit en 1977 par Ron Rivest, Adi Shamir et Len Adleman;
les lettres RSA résultant des initiales de leurs noms respectifs.
RSA a été breveté par le MIT en 1983 aux États-Unis d'Amérique, mais le brevet a expiré le 21 septembre 2000. RSA repose sur l'exponentiation modulaire.

RSA permet le chiffrement et la signature.

La sécurité de cet algorithme repose sur deux conjectures :

- casser RSA nécessite la factorisation d’un [très grand] nombre
- la factorisation est un problème difficile. Par difficile, on entend qu'il n'existe pas d'algorithme rapide pour résoudre cette question. Si l'on veut être un peu plus précis, on pense qu'il n'existe pas d'algorithme ayant une complexité polynomiale en temps qui donne les facteurs premiers d'un nombre quelconque.

Il est possible que l'une des deux conjectures soit fausse, voire que les deux le soient. Si c'est le cas, alors RSA n'est pas sûr. Cela fait néanmoins maintenant plus de 20 ans que RSA est cryptanalysé et celui-ci n'ayant pas encore été cassé, on peut raisonnablement le considérer comme un algorithme sûr.

En 2005, le plus grand nombre factorisé par les méthodes générales et l'état de l'art en matière de calculs distribués, était long de 663 bits. Les cléfs RSA sont habituellement de longueur comprise entre 1024 et 2048 bits. Quelques experts croient possible que des clefs de 1024 bits soient cassées dans un proche avenir (quoique ce soit controversé); mais peu voient un moyen de casser des clefs de 4096 bits dans un avenir prévisible. On présume donc que RSA est sûr si la taille de la clé est suffisamment grande. On peut trouver la factorisation d'une clé de taille inférieure à 256 bits en quelques heures sur un ordinateur individuel, en utilisant des logiciels déjà librement disponibles. Pour une taille allant jusqu'à 512 bits, et depuis 1999, il faut faire travailler conjointement plusieurs centaines d'ordinateurs. Par sûreté, il est couramment recommandé que la taille des clés RSA soit au moins de 2048 bits.

Attaque par chronométrage (timing attacks)

Kocher décrit en 1995 une nouvelle attaque ingénieuse contre RSA : en supposant que l'attaquante Eve en connaisse suffisamment sur le matériel d'Alice et soit capable de mesurer les temps de déchiffrement de plusieurs documents chiffrés, elle serait en mesure d'en déduire rapidement la clef de déchiffrement. Il en irait de même pour la signature. En 2003, Boneh et Brumley ont montré une attaque plus pratique permettant de retrouver la factorisation RSA sur une connexion réseau (SSL) en s'appuyant sur les informations que laissent filtrer certaines optimisations appliquées au théorème des restes chinois. Une façon de contrecarrer ces attaques est d'assurer que l'opération de déchiffrement prend un temps constant. Cependant, cette approche peut en réduire significativement la performance. C'est pourquoi la plupart des mises en oeuvre RSA utilisent plutôt une technique différente connue sous le nom d'aveuglement [cryptographique] (blinding).

L'aveuglement se sert des propriétés multiplicatives de RSA en insérant dans le calcul une valeur secrète aléatoire dont l'effet peut être annulé en multipliant par son inverse. Cette valeur étant différente à chaque chiffrement, le temps de déchiffrement n'est plus directement corrélé aux données à chiffrer, ce qui mets en échec l'attaque par chronométrage.

Attaque par 'chiffrement choisit' (Adaptive chosen ciphertext attacks)

RSA doit être utilisé avec un schéma de remplissage de manière telle qu'aucune valeur de message, une fois chiffré, ne donne un résultat peu sûr. En 1998, Daniel Bleichenbacher décrit la première attaque pratique de type 'chiffré choisit adaptable', contre des messages RSA. En raison de défauts dans le shéma de remplissage PKCS #1 v1, il fut capable de récupérer des clefs de session SSL. Suite à ce travail, les cryptographes recommandent maintenant l'utilisation de méthodes de remplissage formellement sûres telles que OAEP, et les laboratoires RSA ont publié de nouvelles versions de PKCS #1 résistantes à ces attaques.

Perfect Forward Secrecy (FPS), Confidentialité Persistante

Cette propriété garantie que la découverte par un adversaire de la clé privée d'un correspondant (secret à long terme) ne compromet pas la confidentialité d'une communication. Elle peut être fournie, par exemple, par la génération des clefs de session au moyen du protocole d'échange de clés Diffie-Hellman. Ces clés de session (temporaires) ne pourront pas être retrouvées à partir des clés des participants, et inversement.

Lors d'une connexion TLS, c'est la méthode 'Ephemeral DH' qui est utilisée pour fournir une telle caractéristique. Cela signifie que, même si vous avez obtenu la clé privée du serveur en soudoyant un administrateur, en la volant, ou même grâce à une décision de justice, vous ne serez pas en mesure de déchiffrer les échanges que vous auriez pu enregistrer dans le passé. La clé privée n'est d'aucune aide dans ce cas, car elle n'a été utilisée que pour signer les paramètres DH lors du message "Server Key Exchange".

Les clés

Une clé est une valeur qui est utilisée avec un algorithme cryptographique pour produire un texte chiffré spécifique.
Les clés sont, à la base, de très, très, très grands nombres.
La taille d'une clé se mesure en bits.

Un chiffrement symétrique au moyen d'une clé de 128 bits propose 2^{128} (un nombre à trente-huit chiffres) façons de chiffrer un message. Un pirate qui essaierait de décrypter le message par la force brute devrait donc les essayer une par une. Pour les systèmes à clé publique, il en va autrement. Tout d'abord les clés sont plus longues (par exemple 1024 bits minimum pour RSA) ; ceci est dû au fait qu'elles possèdent une structure mathématique très particulière (on ne peut choisir une suite de bits aléatoire comme en clé secrète). Ensuite, il y a clairement mieux à faire qu'une recherche exhaustive, à savoir exploiter la structure mathématique de la clé (pour RSA, cela mène à la factorisation).

Longueurs de clés asymétriques recommandées pour protéger contre une attaque par une seule entreprise : (les clés doivent êtres substantiellement plus grandes pour protéger contre une agence gouvernementale)

Année	Longueur
1995	1280
2000	1280
2005	1536
2010	1536
2015	2048

La taille d'une clé publique n'est absolument pas comparable avec la taille des clés secrètes employées en cryptographie conventionnelle. Une clé conventionnelle de 80 bits offre une sécurité équivalente à celle d'une clé publique de 1024 bits. Une clé conventionnelle de 128 bits équivaut à une clé publique de 3072 bits. Une clé symétrique de 256 bits serait équivalente à une clé RSA de 15360 bits.

Il faut noter le développement actuel de la cryptographie utilisant les courbes elliptiques (E.C.), qui permettent (au prix d'une théorie et d'implémentations plus complexes) l'utilisation de clés nettement plus petites que celles des algorithmes classiques, pour un niveau de sécurité équivalent. La théorie au sujet de ces courbes étant relativement récente, la cryptographie par courbe elliptique n'est pas très connue et souffre d'un grand nombre de brevets qui empêchent son développement.

- DSA — ECDSA.
- Diffie-Hellman — ECDH
- ElGamal— ECElGamal

Comparaison entre longueurs de clés de différents types :

Bloc Sym.	RSA	E. C.
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Encore une fois, plus grande est la clé, plus grande est la sécurité, mais les algorithmes utilisés pour chaque type de cryptographie sont très différents, et donc, comparer les tailles de clés revient à comparer des pommes et des oranges.

voir aussi : http://www.keylength.com/fr/3/

Exemple : retour sur la sécurité de la carte bancaire, 1998, l'affaire Humpich

Lorsqu'on utilise une carte bancaire pour payer une petite somme chez un commerçant, l'opération est effectuée hors ligne, sans échange d'informations avec la banque (ces informations sont regroupées et communiquées en fin de journée). L'unique contrôle au moment du paiement (en plus du code confidentiel) consiste à vérifier que la carte utilisée est valide, c'est-a-dire qu'elle a bien été émise par un organisme bancaire. Cette procédure est effectuée au moyen d'une signature RSA : chaque carte possède un identifiant qui a été signé par la banque. C'est cette signature, inscrite dans la puce, qui est vérifiée à chaque transaction par le terminal du commerçant. Toute carte comportant une signature valide est donc considérée comme authentique puisque seule l'autorité bancaire possède la clef secrète RSA qui permet de signer.

L'unique faille de ce système, qui a pu être exploitée pour fabriquer de fausses cartes bancaires, était la taille des clefs RSA utilisées. La signature était produite à l'aide d'une clef de 320 bits alors qu'un nombre de cette taille se factorise aisément en quelques jours avec la puissance actuelle d'un ordinateur grand public (cette opération demandait par contre une grande puissance de calcul il y a une dizaine d'années). Pour fabriquer de fausses cartes, il suffisait donc d'obtenir la clef publique de la banque, ce qui est relativement facile puisque cette donnée est présente dans tous les terminaux de paiement. La factorisation de ce nombre fournit alors directement la clef secrète correspondant qui peut être utilisée pour imiter la signature de la banque. Il ne restait plus qu'à choisir un identifiant arbitraire et à inscrire dans la puce d'une carte vierge la signature associée.
Une telle carte sera acceptée par le terminal, et comme il ne s'agit pas d'une vraie carte bancaire modifiée mais d'un leurre, elle n'est (délibérément) pas pourvue d'une protection par code confidentiel.

Ni la sécurité de la carte à puce, ni celle de l'algorithme de signature RSA ne sont remises en cause par cette attaque.
Le seul problème était la taille de la clef utilisée, qui est passé par la suite à 768 bits.

Dans tout système cryptographique digne de ce nom, la taille des clefs doit évidemment évoluer en même temps que la puissance des ordinateurs.

Illustration de la différence de nature des clés asymétriques :

	RSA	DH	DSA	E. Curves
paramètres (publics)		**p** **g**	**p** **g** **q**	**p** *ou* **m, f(x)** **a,b** **G** **n** **h**
clé publique	**p × q** **e**	**y**	**y**	**d × G**
aléa (temp.)			**k**	**k**
secret		**s**		
clé privée	**d** *soit* **p** **q** **dmp1** **dmq1** **iqmp**	**x**	**x**	**d**

p, q premiers
e, d entiers
g entier (générateur choisi)
x aléa
$y = g^x \bmod p$
dmp1 = d mod (p-1)
dmq1 = d mod (q-1)
iqmp = (1/q) mod p

p premier, caractéristique du corps fini F_p,
m entier positif
f(x) polynôme binaire irréductible de degré m caractérisant F_{2^m}
a, b constantes (coefficients) déterminant la courbe elliptique
$G = (x_G, y_G)$ point, générateur
n nombre premier (ordre)
h entier (cofacteur)
d entier aléatoire

http://www.secg.org/collateral/sec1_final.pdf

Affichage de différents types de clé avec OpenSSL (www.openssl.org) :

```
>openssl rsa -in TestK005.prv.pem -text -noout

Private-Key: (1024 bit)
modulus:                                        n = p x q,
    00:cb:a6:34:d9:39:06:7a:c9:22:9c:9f:ee:44:cb:
    58:c8:1a:f9:35:f5:e3:5a:3b:f5:3e:75:01:01:45:
    39:9c:0e:a1:b9:44:db:be:47:3f:8a:67:f9:85:7c:
    97:94:9c:f0:6f:f2:1d:0b:99:1a:08:35:1b:8e:d6:
    5f:ef:30:18:4d:1b:a6:6d:62:0c:dc:56:cb:53:77:
    b2:cf:d8:fd:13:3f:bb:e6:8f:64:63:69:fc:12:db:
    98:3c:25:8e:db:2a:20:14:31:a5:e2:38:05:aa:24:
    79:5b:9a:31:58:77:9f:21:9c:5f:20:10:ef:7c:e1:
    9c:82:07:77:ee:18:93:29:1d
publicExponent: 65537 (0x10001)                 e,
privateExponent:                                d,
    00:94:6c:b5:b5:e0:27:05:d4:94:62:5c:f9:d6:af:
    f2:2c:1d:e4:a6:5c:68:f0:7a:24:9a:f9:c1:da:c0:
    2e:65:bc:10:48:ac:94:0f:91:74:11:17:08:b8:2e:
    7f:77:b4:0e:55:38:bb:cc:99:30:6c:ec:f0:01:e1:
    e2:97:bc:90:e8:4c:38:ed:a5:af:12:e0:8a:9c:be:
    14:87:04:e9:0b:2f:6c:c8:0e:dc:d1:85:36:78:10:
    c8:38:22:21:7b:d7:27:13:12:33:65:0c:97:39:1e:
    90:73:d7:d5:51:40:19:c9:62:fc:66:a3:55:3e:df:
    c4:5a:02:3e:7a:d4:69:d8:21
prime1:                                         p,
    00:f6:d2:b7:1d:94:d0:e3:72:2f:a5:2e:12:46:c4:
    b5:c7:92:fe:27:83:c7:24:f0:da:dc:ca:b1:f5:33:
    bd:24:94:9a:0d:ab:84:db:bf:af:d9:a0:89:57:ee:
    aa:1d:e4:d9:89:a0:d7:18:5d:e8:d7:dc:14:a0:75:
    42:6b:38:76:75
prime2:                                         q,
    00:d3:38:8e:f3:a3:28:d4:15:fb:a3:4d:f9:d2:36:
    90:54:47:13:04:2b:26:39:95:c9:fb:bf:14:83:21:
    97:d7:62:1f:17:34:20:fa:c5:42:26:38:d0:7f:d5:
    f3:71:a1:93:9f:7b:41:3f:cf:09:0f:16:ae:1f:13:
    b5:53:31:23:09
exponent1:                                      dmp1 = d mod (p-1),
    00:d8:45:a0:a0:31:f2:ab:29:35:a8:65:eb:2b:c9:
    57:82:cd:41:17:bc:b7:35:9e:3a:18:37:1f:a1:bc:
    39:22:a1:77:2f:3d:38:48:18:f9:6c:16:e7:e1:7d:
    c5:e0:35:d3:8b:6d:bc:ab:a4:35:cf:57:0f:57:de:
    07:59:cd:fa:b5
exponent2:                                      dmq1 = d mod (q-1),
    00:cc:07:01:06:d4:df:16:66:99:a8:b8:24:8d:eb:
    08:e5:6f:b6:2d:bb:a4:73:d1:7d:c7:00:5f:46:ff:
    87:15:95:01:65:3f:84:6f:d7:65:3b:58:7c:06:4f:
    db:95:32:b7:4f:41:16:d9:15:1a:b2:09:7d:6e:25:
    72:6c:86:b5:49
coefficient:                                    iqmp = (1/q) mod p
    5d:7d:52:7c:b2:af:f1:93:3e:1f:24:6b:d2:b5:fe:
    81:1d:2e:c3:e4:cb:18:19:cd:5f:01:85:90:fb:57:
    18:fa:98:96:17:ec:76:a7:aa:1f:22:ae:49:42:4c:
    4f:1d:d4:42:08:e7:46:9b:4b:67:b4:d1:cb:6f:fb:
    8d:8b:07:d2
```

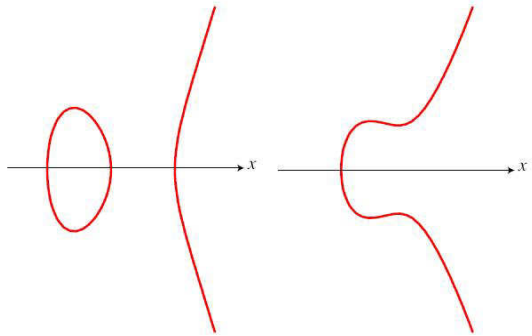


```
>openssl dsa -in dsakey.pem -text -noout

read DSA key
Private-Key: (1024 bit)
priv:
    6f:05:75:da:28:c6:3f:83:d2:ff:4c:09:a7:cc:7d:
    09:89:98:01:fb
pub:
    03:f3:d2:cb:14:99:84:ac:55:a9:63:26:76:dc:e9:
    2d:16:92:e0:90:e7:3a:00:e6:9e:4e:46:9a:b1:d8:
    1f:5e:48:2e:a3:1c:15:8a:97:7c:51:a4:a7:5d:b3:
    be:b2:7d:d4:b4:b5:e1:35:6d:43:89:c8:15:f9:5d:
    95:2a:84:b5:dd:04:1f:60:eb:83:a7:e8:ea:84:5a:
    35:5e:bc:da:4d:5c:4b:e5:3a:29:9a:34:dd:c4:ba:
    3d:e7:41:d4:04:75:a5:f7:cf:8b:33:c8:ab:ea:ad:
    66:a1:7e:12:61:a8:d2:1e:cf:ad:92:95:67:09:58:
    49:80:87:c8:c2:10:60:a4
P:
    00:e3:91:8b:5c:84:32:9a:15:52:13:9e:76:32:cf:
    b0:3b:ad:c6:bc:5e:0a:01:23:e7:d1:da:22:5b:eb:
    4f:27:2b:2e:c4:e4:b3:02:c2:cd:88:e6:b1:77:5c:
    6c:4c:f7:d1:e2:2a:f4:ec:a4:b0:0e:64:a3:49:3d:
    54:38:37:2a:bb:5d:b1:2c:25:5a:c6:f8:c7:2f:0a:
    d7:d9:d2:ff:1e:14:9f:83:56:9e:a2:1b:50:c4:fb:
    ca:d5:33:39:47:46:5b:5d:51:05:55:a5:fa:63:01:
    4d:ad:19:ad:70:10:71:5d:ea:55:78:81:65:37:13:
    a6:74:07:d2:aa:5c:9e:5c:3f
Q:
    00:90:06:2e:da:8b:ed:04:81:da:f9:ec:b5:df:57:
    6d:7a:0e:8d:63:27
G:
    2e:ef:69:9f:41:6c:3d:01:8b:6c:23:03:98:fe:8d:
    82:dc:74:8c:52:1e:0d:6f:bc:b4:59:61:dc:88:9a:
    92:51:3e:16:c8:51:ab:bc:8f:07:99:27:59:44:a4:
    93:27:b9:be:0a:a5:e1:05:30:9d:3a:0b:4b:ab:f5:
    9b:80:b5:43:44:af:1b:0e:98:dc:30:1d:fb:26:a7:
    b6:56:50:31:3f:ff:cb:e8:9a:fc:24:a5:41:7b:bb:
    0c:87:21:20:f7:24:2f:88:1b:e7:18:b2:dc:6a:0c:
    9e:7f:a4:86:f0:98:8b:11:1f:46:1e:07:e8:d1:44:
    69:21:60:72:81:21:28:d2
```

```
>openssl ec -in ecckey.pem -text -noout

Private-Key: (256 bit)
priv:
    5a:6a:90:e6:a1:74:1d:4f:86:3f:0a:b3:b6:3b:f4:
    6c:a1:35:ae:20:f2:7b:fc:e6:1d:fe:64:fd:cd:0d:
    41:d1
pub:
    04:77:94:4d:d7:04:7a:95:ce:47:87:cd:d9:e0:ba:
    20:a1:ea:67:ea:e0:6b:5e:3f:2e:91:58:26:77:c1:
    c5:1e:5d:53:c4:4b:f0:e0:7a:8f:0d:c9:5e:1b:66:
    34:3a:7e:d6:5d:86:e1:3e:53:48:4a:ce:7a:0c:71:
    94:32:74:b6:7e
Field Type: prime-field
Prime:
    00:ff:ff:ff:ff:00:00:00:01:00:00:00:00:00:00:
    00:00:00:00:00:00:ff:ff:ff:ff:ff:ff:ff:ff:ff:
    ff:ff:ff
A:
    00:ff:ff:ff:ff:00:00:00:01:00:00:00:00:00:00:
    00:00:00:00:00:00:ff:ff:ff:ff:ff:ff:ff:ff:ff:
    ff:ff:fc
B:
    5a:c6:35:d8:aa:3a:93:e7:b3:eb:bd:55:76:98:86:
    bc:65:1d:06:b0:cc:53:b0:f6:3b:ce:3c:3e:27:d2:
    60:4b
Generator (uncompressed):
    04:6b:17:d1:f2:e1:2c:42:47:f8:bc:e6:e5:63:a4:
    40:f2:77:03:7d:81:2d:eb:33:a0:f4:a1:39:45:d8:
    98:c2:96:4f:e3:42:e2:fe:1a:7f:9b:8e:e7:eb:4a:
    7c:0f:9e:16:2b:ce:33:57:6b:31:5e:ce:cb:b6:40:
    68:37:bf:51:f5
Order:
    00:ff:ff:ff:ff:00:00:00:00:ff:ff:ff:ff:ff:ff:
    ff:ff:bc:e6:fa:ad:a7:17:9e:84:f3:b9:ca:c2:fc:
    63:25:51
Cofactor:  1 (0x1)
Seed:
    c4:9d:36:08:86:e7:04:93:6a:66:78:e1:13:9d:26:
    b7:81:9f:7e:90
```



$y^2 = x^3 - 7x$ $y^2 = x^3 - x + 5$

Les logiciels

PGP

PGP est certainement le logiciel de chiffrement le plus réputé et le plus utilisé au monde. Les fonctionnalités de base sont toujours disponibles gratuitement, et il existe une version 'libre' (GnuPG).

PGP propose un modèle de chiffrement hybride basé, au choix, sur des clés RSA ou DH/DSS (en fait ElGamal), accompagnées d'un sous-ensemble d'algorithmes symétriques parmi les meilleurs.

Les clés PGP sont en fait formées de 2 paires de clés liées entre elles (ou plus).
La paire principale à vocation à offrir les services de signature, alors que la/les sous-clés seront utilisées pour le chiffrement. L'idée sous-jacente est de permettre l'utilisation sur le long-terme de la paire de clé principale - sur laquelle est basé l'empreinte et l'identifiant de clé, et qui porte la confiance des autres (signatures) - tout en permettant de changer périodiquement les clés de chiffrement.

GnuPG (pas PGP) peut aussi générer des sous-clés pour la signature.

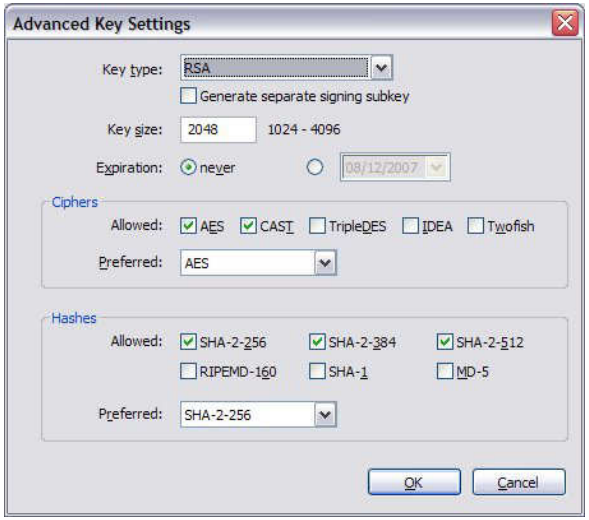
Lorsqu'une clef ne convient plus, que ce soit parce que la clef privée a été découverte ou parce que la longueur de la clef s'avère trop petite ou pour n'importe quelle autre raison, il est possible de la révoquer. Un certificat de révocation permet de faire savoir publiquement qu'une paire de clefs ne doit plus être utilisée. Dans le modèle PGP, pour éviter que n'importe qui puisse générer un tel certificat, ce dernier est signé par la clef privée de la paire à révoquer. La validité d'un certificat de révocation est ainsi aisément vérifiable par tout un chacun. Pour révoquer une clé, il faut donc toujours en contrôler la partie privée, et sa passphrase. C'est pourquoi il n'est pas superflu de générer un certificat de révocation juste après la création de la clé, car il sera peut-être impossible de le produire quand on en aura vraiment besoin (perte de clé) (pgpfaq ch. 5).

Les clefs de chiffrement peuvent avoir à être présentées en cas d'enquête.
Ce n'est pas le cas des clés de signature. Personne (pas même la police) ne peut exiger l'accès à votre clef secrète de signature.
Elles ne doivent pas pour autant être plus fortes ou mieux protégées (mais c'est discuté), car si une clé de chiffrement est compromise, c'est la confidentialité de toutes les données chiffrées avec cette clé qui est perdu, alors que l'horodatage ou le traçage des documents offrent un moyen de prouver l'authenticité d'une ancienne signature (qui pourrait avoir été contrefaite à l'aide d'une technologie nouvelle).

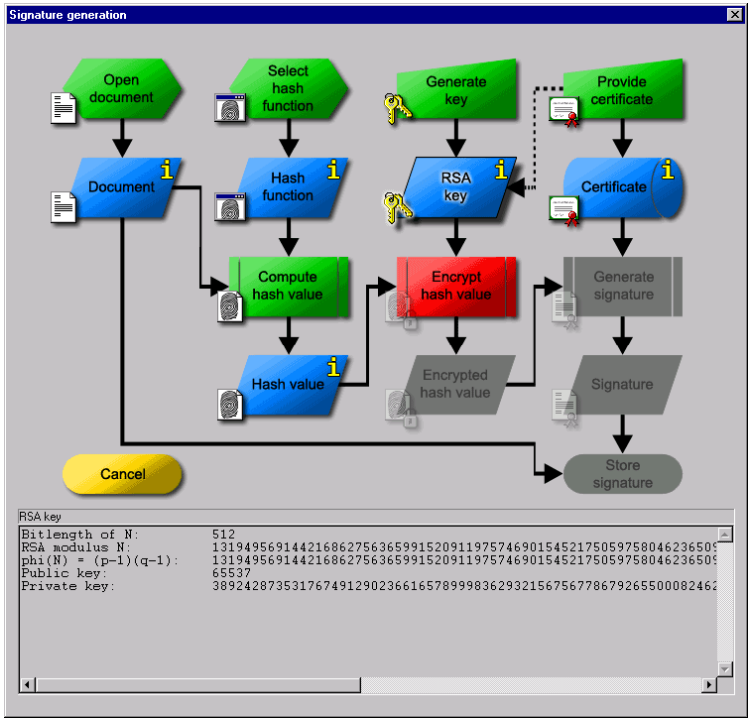
C'est pourquoi le fait de pouvoir produire - dans un avenir plus ou moins proche - une fausse signature n'est pas forcément aussi catastrophique que la perte d'une clé de chiffrement.

Les différentes versions de PGP et/ou GnuPG ne sont pas toutes compatibles entre-elles. Cette situation qui résulte des évolutions et divergences dans les technologies et standards tend à s'améliorer depuis la parution de la spécification OpenPGP (RFC2440), et les efforts de convergences vers/depuis les autres modèles de sécurité (X509).

.Préférences dans PGP : (www.pgp.com)



Cryptool (www.cryptool.org) :



[Partie 3 : études de cas]

SET – Secure Electronic Transaction

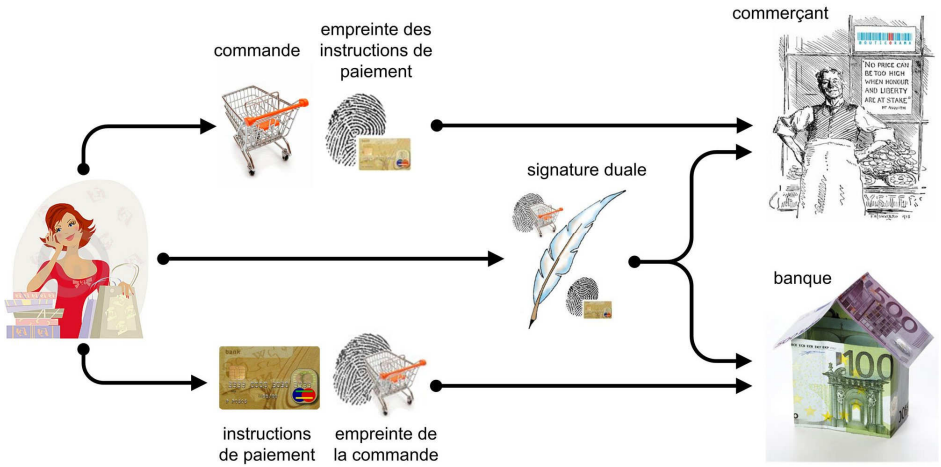
En 1996, MasterCard et Visa ont annoncé le développement d'une norme pour assurer la protection des transactions de paiement effectuées par cartes de crédit sur les réseaux ouverts.
Le protocole SET permet une identification préalable à la transaction de l'acheteur et du marchand. Ceci permet aux deux parties de s'assurer que le paiement sera effectué de la même manière que dans un contexte physique traditionnel. Cette identification s'effectue à l'aide de certificats électroniques qui sont émis par les institutions financières membres de Visa ou de MasterCard. Le protocole utilise également la cryptographie afin de protéger la confidentialité des renseignements personnels et financiers transmis lors des transactions, et pour assurer l'intégrité des ordres de paiement, et ce, indépendamment de la sécurité de la couche de transport.

Grâce a ce procédé, l'acheteur envoie simultanément son offre au commerçant et les instructions de paiement à la banque, en tenant compte des deux contraintes ci-dessous :

- Les deux sont mutuellement conditionnés, car le paiement ne doit être effectué que si l'offre est acceptée par le commerçant et la commande n'est effective que si la banque approuve le paiement.
- Le contenu de la commande doit être caché à la banque et les instructions de paiement doivent être cachées au commerçant.

Pour éviter des allers et retours complexes, une manœuvre élégante à été imaginée à cette occasion. Les deux messages (**O** = offre et **I** = instructions) sont réduits en deux empreintes électroniques **E(O)** et **E(I)**. Les empreintes sont concaténées puis signés par l'acheteur **S({ E(O), E(I) })**. C'est la 'signature duale'.

L'acheteur transmet l'offre au commerçant et les instructions à la banque.
Il joint à chacun des documents l'empreinte de l'autre et la signature duale.



Les deux destinataires peuvent s'assurer de l'authenticité du message qu'ils reçoivent.

Si le commerçant accepte l'offre, il transmet son acceptation à la banque, accompagnée de l'empreinte de l'offre.
La banque est donc en mesure de faire le lien avec les instructions.

Le procédé réellement utilisé par SET est simplifié par rapport au procédé théorique ci-dessus, car aucun message n'est transmis de l'acheteur à la banque. Celle-ci est en mesure de reconstituer les mêmes instructions de paiement **(I)** à l'aide des informations bancaires sur l'acheteur et du prix présenté par le commerçant.
Elle vérifie les intentions de l'acheteur en comparant l'empreinte des instructions **E(I)**.

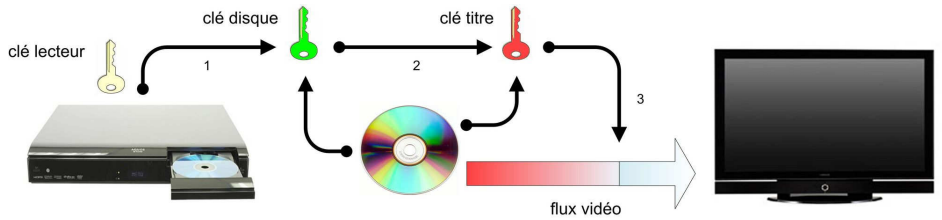
Malgré ses atouts, SET n'a pas réussi à être massivement adopté par le marché, notamment à cause du coût et de la complexité supportée par le commerçant (comparé à l'alternative proposée par la seule protection du transport par SSL) et à cause de la logistique nécessaire à la distribution des certificats et l'installation des logiciels clients.

SET a été abandonné au début des années 2000 au profit du programme 3D Secure.

La protection des DVD – CSS – Content Scramble System

Un fichier contenant un flux vidéo numérique est copiable comme n'importe quel autre fichier. Alors comment faire pour empêcher qu'il soit partagé ou recopié sans empêcher le propriétaire de l'utiliser ?

Un lecteur de DVD ne copie pas le contenu vidéo, il peut seulement le lire et le diffuser. La solution retenue fut de mettre en place un système cryptographique entre le DVD et le lecteur de DVD, afin de s'assurer que SEUL un lecteur 'agréé' pourrait en extraire le contenu. Une façon de compliquer le système était d'enchainer plusieurs étapes de chiffrement impliquant plusieurs clés et un chiffrement par flot propriétaire nommé CSS (Content Scramble System) :



1 - après la phase d'authentification, la clé du lecteur sert à déchiffrer la clé du disque,
2 - la clé du disque sert à déchiffrer la clé des titres,
3 - la clé des titres sert à déchiffrer le flux vidéo.

Chaque disque contient donc la clé 'disque' chiffrée avec toutes les clés 'lecteur' existantes (environ 400). Une organisation nommé DVD CCA (DVD Copy Control Association) eut la charge de distribuer les clés aux fabricants de lecteurs (Sony, Panasonic, ...), qui doivent payer pour l'obtenir, et qui ensuite s'engagent à l'intégrer dans leur lecteur avec les circuits de déchiffrement.
Comme il est extrêmement dur, voire impossible de récupérer une clé enfouie dans l'électronique du matériel, on pensait [naïvement] que tout était réglé.

Le problème avec cette solution, c'est que les clés et le mode de chiffrement devaient rester secrets. Or, même en faisant confiance aux fabricants, vu leur nombre et vu les millions d'utilisateurs concernés, il était prévisible que des fuites se produisent ou que quelqu'un trouve un moyen de pirater un tel système. Et c'est exactement ce qui arriva.

A cause même de cette volonté de secret, le CCA refusa de fournir des licences pour l'utilisation sur des systèmes ouverts comme Linux. Dés lors, les développeurs Linux n'avaient d'autres choix que de trouver un moyen de contourner cet empêchement.
Or, en plus de ne dépendre que du secret d'un petit nombre de clé définies une fois pour toute (environ 400), l'algorithme CSS révéla rapidement ses faiblesses.
La première provenait essentiellement des contraintes imposées par l'administration américaine qui interdisait l'exportation de système utilisant des clés de plus de 40 bits, une taille dont on savait déjà à l'époque qu'elle était trop faible au regard de la puissance de calcul des machines disponibles et de son augmentation prévisible.
Pour ne rien arranger, des défauts de conception exposaient l'algorithme à une attaque par force brute en moins d'une minute, au moyen d'une machine équipée d'un processeur à 450MHz, configuration minimale requise pour décompresser en temps réel un flux vidéo MPEG-2. En conséquence, n'importe quelle machine capable de lire un DVD était aussi capable de le pirater.

Vers la fin octobre 1999 un adolescent de 15 ans, nommé depuis 'DVD Jon', publia sur InterNet le logiciel DeCSS qui permettait de déchiffrer le contenu des DVDs.
La découverte fut facilité par le fait que les concepteurs d'un des lecteurs distribué par Xing Technologie, une filiale de RealNetworks, avait oublié de protéger la clé de déchiffrement…
Avec ces 5 octets (40bits) connus, et s'appuyant sur les défauts de l'algorithme CSS, 170 autres clés furent rapidement dévoilées. C'était la fin du secret et il y avait peu de chance pour l'industrie du disque de trouver un moyen de renforcer le système tout en restant compatible avec les millions de lecteurs et les milliards de disques déjà en circulation à travers le monde.

AACS (Advanced Access Content System)

Le nouveau standard de DVD Haute Définition utilise AACS qui tente de renforcer la protection des contenus, [entre autre] grâce aux moyens suivants :
- utilisation d'un chiffrement public, robuste et réputé sûr : AES,
- chaque lecteur individuel (et non chaque modèle de lecteur) se voit attribuer un jeu unique de plusieurs clés, impliquées dans un schéma de chiffrement pour large diffusion (broadcast encryption),
- un système de tatouage numérique (watermarking) des contenus est utilisé pour identifier les clés compromises. De cette manière il est envisageable de 'révoquer' ces clés en ne les utilisant pas pour le chiffrement des futurs média.

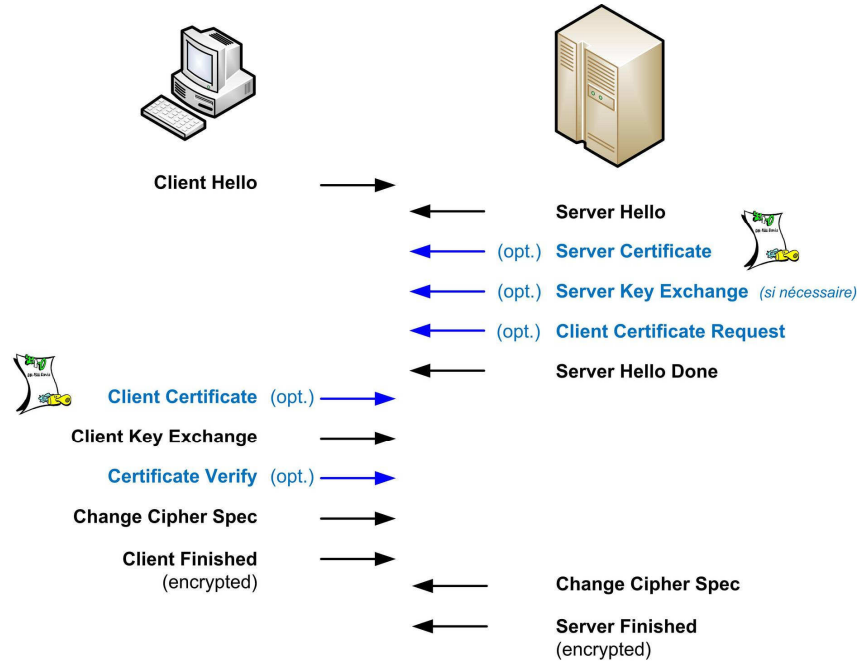
Néanmoins il semble vain d'espérer une totale protection des contenus dans le contexte d'une architecture PC standard. C'est pourquoi certains proposent déjà une nouvelle architecture 'de confiance' - Trusted Computer -, mais de confiance pour qui ? l'industrie des médias ou l'utilisateur de la machine ?

"Digital files cannot be made uncopyable, any more than water can be made not wet."
https://www.schneier.com/crypto-gram/archives/2001/0515.html#3

TLS – Transport Layer Security

TLS, anciennement nommé SSL (Secure Socket Layer), est le protocole de sécurisation des communications sur Internet. Il a été développé à l'origine par Netscape au début des années 1990.
Il permet d'authentifier (optionnellement) les parties (client et serveur) et garantit la confidentialité et l'intégrité de leurs futurs échanges, en négociant dés le début de la connexion les algorithmes et paramètres nécessaires.

Ci-dessous les grandes étapes de cette négociation (Handshake) :



On peut noter que seule la clé publique du serveur est utilisée pour faire du chiffrement; celle du client ne sert que pour de la signature.

Chaque fois que des failles ont étés découvertes, le protocole a évolué.

Récemment encore (en 2009), des attaques ont été publiées qui exploitent la grande permissivité du protocole : on ne peut imposer l'authentification du client, et la manière de procéder à la vérification des certificats 'serveur' peut s'interpréter de manière trop laxiste (signatures, extensions, listes de révocation, …)

Par ailleurs, le protocole étant très généraliste, il existe des possibilités d'exploitation indirecte de la naïveté ou la négligence des utilisateurs peu avertis (phishing, …).

Réutilisation de session

Pour établir une connexion TLS, le client et le serveur doivent échanger quatre messages. Avec une latence de 50 ms, cet échange correspond à une pénalité de 200 ms. De plus, afin de s'échanger un secret commun, les deux parties doivent mettre en œuvre des primitives cryptographiques à clef publique qui sont coûteuses en temps de calcul.

C'est pourquoi la réutilisation de session est l'un des mécanismes les plus importants pour améliorer les performances : le client peut effectuer une 'poignée de mains' raccourcie en soumettant dès sa seconde requête une donnée présentée précédemment par le serveur, permettant ainsi de réduire la latence et économiser des calculs.

Pour cela il existe 2 mécanismes distincts :

• les identifiants de session (RFC 5246)

Un identifiant de session est ajouté au message 'Server Hello'. Le client doit le conserver et le présenter dans le message 'Client Hello' lors du prochain établissement de session. Si les données correspondantes sont toujours présentes dans le cache du serveur et qu'il accepte de les réutiliser, il retourne le même identifiant et continue avec la version abrégée de la négociation. C'est le mécanisme le plus répandu car le plus ancien. Son inconvénient principal réside dans le cout du maintien des informations de session coté serveur, et la difficulté de les partager entre plusieurs hôtes quand on a besoin de répartir la charge.

• les tickets de session (RFC 5077)

Lors du dernier échange de la 'poignée de main', le serveur peut inclure un message 'New Session Ticket' qui contient l'état complet de la session (y compris le secret partagé), le tout chiffré et protégé par une clef connue du serveur.
Avec une répartition de charge classique, tous les serveurs coopérants doivent générer la même clef s'ils veulent être en mesure d'accepter les tickets des copains.

Renégociation

La renégociation est une demande de changement des paramètres de chiffrement, durant une session déjà établie. Elle peut être entreprise à l'initiative du client ou du serveur. Ce dernier peut initier une renégociation si le client demande une ressource nécessitant des accréditations particulières ou une élévation de privilège. Le client n'a habituellement aucune raison de demander une renégociation, et c'est source de vulnérabilité

http://vincent.bernat.im/fr/blog/

Vote Electronique

La réalisation d'un système de vote électronique n'est pas une tâche facile.
En effet, une telle procédure nécessite la satisfaction de plusieurs propriétés essentielles, en apparence contradictoires :

- Précision :
  - le résultat du vote ne doit pas pouvoir être altéré, par exemple en ajoutant ou modifiant des bulletins (intégrité), sans être découvert.

- Démocratie :
  - seuls les votants éligibles peuvent voter.
  - chaque votant ne peut voter qu'une seule fois.

- Confidentialité :
  - personne ne doit pouvoir faire le lien entre un votant et son vote (anonymat).
  - un votant ne peut prouver comment il a voté : les électeurs ne peuvent vendre leurs votes que s'ils sont capables de prouver à l'acheteur qu'ils ont réellement voté d'après leurs vœux (prévention contre l'achat du vote).

- Vérifiabilité :
  - individuelle : chaque votant peut vérifier que SON propre bulletin a été correctement pris en compte.
  - universelle : toute personne peut vérifier que TOUS les bulletins ont été comptés correctement.

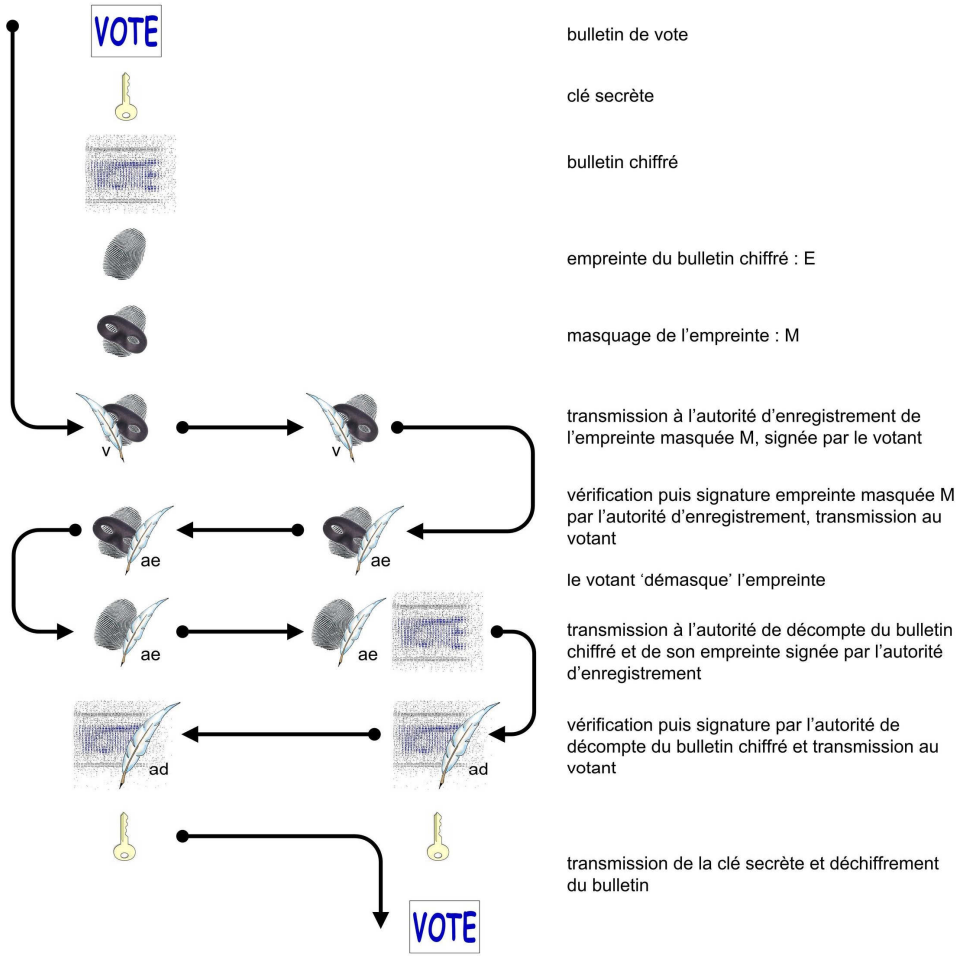
Accessoirement on peut ajouter :
- Commodité : système rapide et d'utilisation simple,
- Souplesse : permet la mise en œuvre de tout type d'élections,
- Mobilité : pouvoir voter de n'importe quel lieu.

Depuis la première publication de Chaum en 1981, de nombreuses propositions ont été faites, mais il ne semble pas qu'on soit encore (en 2009) arrivé à les satisfaire toutes à la fois, sauf à faire confiance à un ou plusieurs participant ou composant du système.

Il est à noter que certaines propositions présentent des caractéristiques de sécurité intéressantes sans utiliser aucun mécanisme cryptographique
(les 3 bulletins de Rivest par exemple).

Certains protocoles utilisent le chiffrement homomorphique, et les preuves à divulgation nulle. D'autres se basent sur les signatures aveugles et les canaux anonymes, via des réseaux de mélangeurs (mix-net).

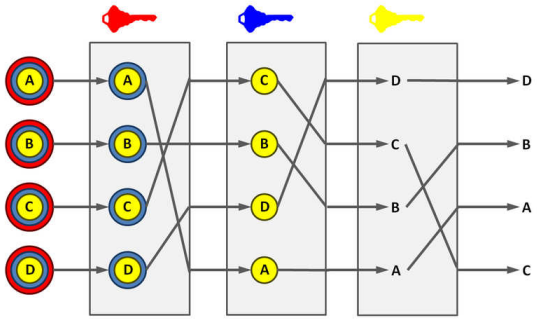
Ci-après, représentation d'une architecture ou les signatures aveugles sont une condition nécessaire, mais pas suffisante :



Réseaux de mélangeurs (mix-net)

Les Mix Nets s'efforcent de permuter les votes de manière aléatoire afin d'éliminer la correspondance entre le texte chiffré, et son équivalent déchiffré.

Le votant chiffre son vote avec les clefs publiques des machines prenant part au réseau. Les messages sont déchiffrés avec la clef privée du premier serveur, qui les permute de manière aléatoire avant de les transmettre au serveur suivant du réseau, qui procède de même jusqu'au déchiffrement total des messages.



La confidentialité est assurée tant qu'au moins un serveur reste honnête.

Que se passe-t-il si un serveur venait à tomber ? Aucun message ne pourrait plus être déchiffré. Une solution envisageable est de séparer les clefs privées de chaque serveur, et de distribuer ces parties aux autres serveurs. Ainsi, si un serveur tombe, sa clef privée peut être reconstruite `a l'aide des autres serveurs. Dés lors, la confidentialité n'est assurée que lorsqu'une majorité de serveurs reste honnête.

Il est également possible qu'un serveur en vienne à falsifier un bulletin.
Des solutions existent pour pallier à ce problème : chaque serveur doit prouver qu'il a permuté et déchiffré de manière correcte. Cette preuve doit être signée numériquement et transmise avec le bulletin vers le prochain serveur

Chiffrement Homomorphique

Un algorithme de chiffrement à clef publique est dit homomorphique s'il satisfait la propriété suivante : $E(v_1) \times E(v_2) \times E(v_3) \times \ldots \times E(v_n) = E(v_1 + v_2 + v_3 + \ldots + v_n)$.
Si l'on «multiplie » n bulletins chiffrés (pour la même clef publique), on obtient le chiffrement de la « somme » des bulletins originaux. De cette manière, déchiffrer le produit des bulletins chiffrés donne le résultat du scrutin sans avoir eu besoin de déchiffrer chaque bulletin individuellement.

Les protocoles basés sur cette propriété présentent l'inconvénient de ne pas pouvoir prendre en compte les questions de type ouvertes.

« Informatique de confiance - Trusted Computing »

http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html

http://www.gnu.org/philosophy/can-you-trust.fr.html

Partie 4 : cryptographie ancienne

Cryptographie ‘classique’ (ancienne)

Chiffrements par transposition

Modification de l’ordre des lettres

Principe : écriture en colonne, lecture en ligne
Origine dans la Grèce antique (scytale)

Un tel chiffrement est en fait une variante des codes basés sur des grilles que l'on remplit ligne par ligne avec le texte en clair : les transpositions rectangulaires.
Un texte chiffré correspond aux colonnes de la grille.
Un tel code est faible et peut être facilement cassé.

Chiffrements par substitution

Remplacement des lettres par des lettres d’un autre alphabet

- mono-alphabétique : remplace chaque lettre du message par une autre lettre de l'alphabet

  Chiffrement de César, ROT13, …

- poly-alphabétique : le chiffrement est défini au moyen d’une suite de symboles mono-alphabétiques (la clé) réutilisée périodiquement et d’un tableau de correspondance d’alphabet (1467)

  Chiffre de Vigénère (1585)

- homophonique : fait correspondre à chaque lettre du message en clair un ensemble possible d'autres caractères

  carré de Polybe, grand chiffre de Louis XIV (Rossignol)

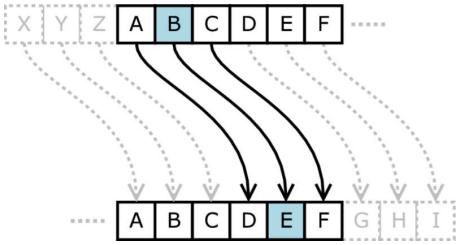
- polygrammes : substitue un groupe de caractères dans le message par un autre groupe de caractères.

  Pour des groupes de 2 caractères (digramme) nécessite un nouvel alphabet de $26^2 = 676$ symboles maximum, les combinaisons rares pouvant être négligées. Le premier code de ce genre vraiment utilisable fut le Chiffre de PlayFair (1854).
  Le Chiffre de Hill (1929) peut combiner des groupes de lettres plus importants en utilisant l’algèbre linéaire.

Chiffre de César / ROT13

Le code de César est la méthode cryptographique, mono-alphabétique, la plus ancienne communément admise par l'Histoire.

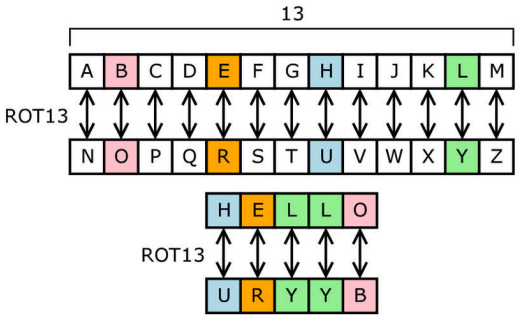
Méthode simple, qui consiste à décaler les lettres de l'alphabet d'un nombre *n*.
Par exemple, si on remplace A par D (n=3), on remplace B par E, C par F...
Chiffrement de césar avec un décalage de 3 :



exemple :
texte à coder : «decaler les lettres de l'alphabet»
résultat : «ghfdohu ohv ohwwuhv gh o'doskdehw»

Système très peu sûr, facilement déchiffrable.
Pourtant sa simplicité conduisit les officiers sudistes à le réemployer durant la guerre de Sécession. L'armée russe fit de même en 1915.

Il est à noter que le code de César a été utilisé sur des forums internet sous le nom de ROT13 (rot-ation de 13 lettres ou A-->N...).
Le ROT13 n'a pas pour but de rendre du texte confidentiel, mais plutôt d'empêcher la lecture involontaire (d'une réponse à une devinette, ou de l'intrigue d'un film, etc.).
Son utilisation est simple: il suffit de re-chiffrer un texte, codé en ROT13, une deuxième fois pour obtenir le texte en clair.



Chiffre de Vigénère (substitution poly-alphabétique)

L'idée de Vigenère est d'utiliser un chiffre de César, mais où le décalage utilisé change de lettres en lettres.
Ce chiffrement introduit la notion de clé.
Une clé se présente généralement sous la forme d'un mot ou d'une phrase.
Pour pouvoir chiffrer le texte, à chaque caractère il faut utiliser une lettre de la clé pour effectuer la substitution.
Évidemment, plus la clé sera longue et variée et mieux le texte sera chiffré.

L'outil indispensable du chiffrement de Vigenère est : « La table de Vigenère »

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Pour chaque lettre en clair, on sélectionne la colonne correspondante et pour une lettre de la clé on sélectionne la ligne adéquate, puis au croisement de la ligne et de la colonne on trouve la lettre codée. La lettre de la clé est à prendre dans l'ordre dans laquelle elle se présente et on répète la clé en boucle autant que nécessaire.

```
Texte en clair :   J'ADORE ECOUTER LA RADIO TOUTE LA JOURNEE
Clé répétée    :   M USIQU EMUSIQU EM USIQU EMUSI QU EMUSIQU
                   | ^^^
                   | ||Colonne O, ligne I: on obtient la lettre W.
                   | |Colonne D, ligne S: on obtient la lettre V.
                   | Colonne A, ligne U: on obtient la lettre U.
                   Colonne J, ligne M, on obtient la lettre V.
```

Le texte chiffré est alors :

```
V'UVWHY IOIMBUL PM LSLYI XAOLM BU NAOJVUY.
```


Chiffre de Playfair (substitution polygrammique)

Le chiffre de Playfair utilise un tableau de 5x5 lettres, contenant un mot clé ou une phrase. La mémorisation du mot clé et de 4 règles faciles à suivre suffisait pour utiliser ce chiffrement.

Pour construire le tableau, on devait d'abord le remplir avec les lettres du mot clé (en ignorant les doublons), puis le compléter avec les autres lettres de l'alphabet dans l'ordre (soit en omettant la lettre Q, soit en occupant une même case pour les lettres I et J suivant les versions). Le mot clé peut être écrit en ligne, en colonne ou même en spirale.

Pour chiffrer un message, il faut prendre les lettres 2 par 2 et appliquer les règles suivantes en fonction de la position des lettres dans la table :
- si les 2 lettres sont identiques (ou s'il n'en reste qu'une) mettre un 'X' aprés la première lettre. Chiffrer la nouvelle paire ainsi constituée et continuer avec la suivante. Dans certaines variantes, on utilise 'Q' au lieu du 'X', mais n'importe quelle lettre peu utilisée fait l'affaire,
- si les lettres se trouvent sur la même ligne de la table, il faut les remplacer par celles se trouvant immédiatement à leur droite (en bouclant sur la gauche si le bord est atteint),
- si les lettres apparaissent sur la même colonne, les remplacer par celles qui sont juste en dessous (en bouclant par le haut si le bas de la table est atteint),
- sinon, remplacer les lettres par celles se trouvant sur la même ligne, mais dans le coin opposé du rectangle défini par la paire originale.

Pour chiffrer le digramme 'OR' par exemple, trois configurations peuvent se présenter dans le tableau :

1) sur la même ligne :

* * * * *
* **O Y R Z**
* * * * *
* * * * *
* * * * *

alors, OR -> YZ

2) sur la même colonne :

* * **O** * *
* * **B** * *
* * * * *
* * **R** * *
* * **Y** * *

alors, OR -> BY

3) forment un rectangle :

Z * * **O** *
* * * * *
* * * * *
R * * **X** *
* * * * *

alors, OR -> ZX

Pour déchiffrer, utiliser la méthode inverse en ignorant les 'X' ou les 'Q' qui n'ont pas leur place dans le message final.

En supposant que la clé soit "exemple playfair" :

Tableau :

E	X	M	P	L
A	Y	F	I	R
B	C	D	G	H
J	K	N	O	Q
S	T	U	V	Z

Chiffrement du message :

« Cache l'or dans la souche de l'arbre »

```
CA CH EL OR DA NS LA SO UC HE DE LA RB RE
```
soit :
```
BY DB XE QI BF JU ER VJ TD BL BM ER AH AL
```

Elements de cryptanalyse classique : l'analyse fréquentielle

L'analyse fréquentielle est basée sur le fait que, dans chaque langue, certaines lettres ou combinaisons de lettres apparaissent avec une certaine fréquence.
Par exemple, en français, le 'e' est la lettre la plus utilisée, suivie du 's' et du 'a'. Inversement, le 'w' est peu usité.
Ces informations permettent aux cryptanalystes de faire des hypothèses sur le texte clair, à condition que l'algorithme de chiffrement conserve la répartition des fréquences, ce qui est le cas pour des substitutions mono-alphabétiques et poly-alphabétiques.
Une deuxième condition est nécessaire pour appliquer cette technique avec efficacité : la longueur du message à déchiffrer doit être suffisante pour refléter la répartition générale des fréquences.

	A	B	C	D	E	F	G	H	I	J	K	L	M
Français	9,42	1,02	2,64	3,39	15,87	0,95	1,04	0,77	8,41	0,89	0,00	5,34	3,24
Anglais	8.08	1.67	3.18	3.99	12.56	2.17	1.80	5.27	7.24	0.14	0.63	4.04	2.60
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Français	7,15	5,14	2,86	1,06	6,46	7,90	7,26	6,24	2,15	0,00	0,30	0,24	0,32
Anglais	7.38	7.47	1.91	0.09	6.42	6.59	9.15	2.79	1.00	1,89	0,21	1,65	0,07

On peut aussi analyser la fréquence des bigrammes dans un texte, c'est-à-dire la fréquence des groupes de deux lettres.

L'analyse fréquentielle ne peut être utilisée que pour des codes de substitutions simples (elle est par complètement inefficace contre les méthodes de chiffrement moderne). Elle n'est d'aucun secours pour les codes dits de transposition, qui changent la place des lettres ou des symboles dans le message. Pour savoir si l'on a affaire à un code de substitutions, on peut utiliser l'indice de coïncidence avant l'analyse fréquentielle. Cela permet également d'avoir une longueur de mot-clé conseillée qui pourra servir de base à l'analyse statistique.

Pour se prémunir d'une cryptanalyse par analyse fréquentielle, il existe plusieurs parades utilisées dans les algorithmes de chiffrements. On peut utiliser un chiffre qui attribue plusieurs symboles pour une seule lettre, en fonction de sa fréquence.
(par exemple, on utilise 4 ou 5 symboles pour le 'e' mais un seul pour le 'k').
On dit alors que l'on utilise un code homophonique.
Il est également possible d'utiliser le surchiffrement, qui consiste à recoder le texte chiffré pour ne pas permettre de faire des hypothèses sur les lettres les plus fréquentes. Un texte surchiffré sera donc plus difficile à déchiffrer.

L'analyse fréquentielle est une technique de cryptanalyse fréquemment relatée dans les fictions. On la retrouve par exemple dans Le scarabée d'or d'Edgar Allan Poe ou dans Les Hommes dansants, une aventure de Sherlock Holmes écrite par Arthur Conan Doyle. Dans cette dernière énigme, les messages étaient alors codés avec différents symboles, en forme de personnages dansants.

Partie 5 : éléments Mathématiques

Considérations sur les grandeurs

Des nombres dont l'exposant comporterait plus de 2 chiffres ne sont pas aisément concevables. Bien que les mathématiques puissent en fabriquer facilement, de tels nombres sont dénués de sens physique.
On pressent de la sorte que l'infini est une notion qui n'a aucun rapport avec le réel, puisqu'elle implique des exposants dépassant toute limite.
Nous avons de la difficulté à «concrétiser» l'échelle des puissances de 10 sur laquelle nous nous déplaçons en physique. Il est d'autant plus nécessaire de se montrer vigilant sur des raisonnements faisant intervenir des nombres qui, trop grands, s'avèreraient dépourvus de signification concrète. Se rappeler que 10^{80} est vraiment un nombre considérablement grand dépassant toute imagination peut servir de garde-fou en la matière.

Nombres d'atomes dans l'univers :	10^{80} (10^{78})
Dans notre galaxie :	10^{68}
Dans planète Terre :	10^{50}
Age de l'univers en secondes ≈	10^{18}

Essayer toutes les clés de 256 bits possibles (2^{256} possibilités) serait plus long que de compter un par un tous les atomes du système solaire (2^{189} atomes ≈ 10^{57}).

$\log_a b = (\log_c b)/(\log_c a)$ d'où $\log_2 2^{189} / \log_2 10 = 189 / (\log_{10} 10 / \log_{10} 2)$

XOR

Le «ou exclusif» (XOR, eXclusive OR) est un opérateur logique de l'algèbre de boole.
Il s'entend comme :
«l'un ou l'autre mais pas les deux» ou «vrai quand entrées différentes».
Il se compose de la manière suivante : a **xor** b = (a **ou** b) **et non** (a **et** b)
Table de vérité :

⊕	V	F
V	F	V
F	V	F

Le «ou exclusif» est parfois noté par le signe arithmétique ≠ «différent de», auquel il est équivalent. Fonctionnellement on utilise aussi un + entouré : a ⊕ b.

En pratique :

soit	A le message en clair	=	`0110101011010100`	
et	B la clé	=	`0101011011100110`	
Chiffrement :	S = A ⊕ B	S =	`0011110000110010`	(message chiffré)
Déchiffrement :	A = S ⊕ B	A =	`0110101011010100`	(message déchiffré)

Division de secret (secret spillting)

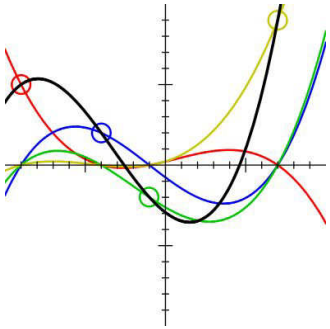
Pour diviser une clé **K** en X fragments, on peut générer X-1 nombres aléatoires N et calculer $N_x = N_1 \oplus N_2 \oplus N_3 \oplus \ldots \oplus N_{x-1} \oplus \mathbf{K}$

On peut avoir le sentiment que seul N_x est en relation avec la clé, mais du fait de l'utilisation de la fonction XOR, chaque morceau est aussi important lors de la reconstruction : $\mathbf{K} = N_1 \oplus N_2 \oplus N_3 \oplus \ldots \oplus N_{x-1} \oplus N_x$

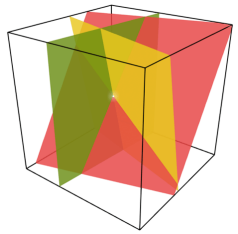
Secret réparti (secret sharing)

Les opérations pour partager un secret avec un seuil s'appuient généralement sur des propriétés mathématiques bien particulières (mais PGP utilise un algorithme à base de clés intermédiaires et de condensés). Parmi celles-ci, on peut distinguer :

- l'approche de Shamir : dans un corps fini, on génère un polynôme de degré k dont le terme constant est la donnée secrète. On donne à chaque dépositaire les coordonnées d'un point distinct, choisi sur la courbe. k dépositaires peuvent alors, par interpolation, retrouver les coefficients du polynôme, et donc la donnée secrète.
- l'approche de Blakley : dans un corps fini, on constitue un système linéaire à n équations et k inconnues, et dont la seule solution est la donnée secrète. Le terme constant est public, et chaque participant reçoit une ligne du système.



Shamir : l'approche la plus répandue. 2 points suffisent à définir une droite, 3 une parabole, 4 une courbe cubique … et de manière générale k points pour un polynôme de degré k-1



Blakley : chaque partie du secret est un plan et le secret est le point d'intersection entre les plans.

Arithmétique modulaire

Idée intuitive : «arithmétique de l'horloge»
L'arithmétique modulaire est un système arithmétique d'entiers modifiés, où les nombres sont abaissés lorsqu'ils atteignent une certaine valeur.
L'idée de base est de travailler non sur les nombres eux-mêmes, mais sur les restes de leur division par quelque chose.

Congruence modulo n

Deux entiers a et b sont dits congruents modulo m, où m est un entier non nul et différent de 1 et -1, si l'une des conditions équivalentes suivantes est vérifiée :

1. ils ont même reste lorsqu'ils sont divisés par m, c'est-à-dire que le reste de la division euclidienne de a par m est égal à celui de la division de b par m;
2. leur différence est divisible par m;
3. il existe un entier k tel que a−b=km
4. a−b ∈ nZ , l'idéal de tous les entiers divisibles par m.

On écrira alors : $a \equiv b\ (n)$ ou $a \equiv b\ [n]$ ou encore $a \equiv b \bmod m$,
ce qui se lit dans tous les cas «a est congru à b modulo m».
Par exemple $14 \equiv 26 \bmod 12$.

Nombre premier, semi-premier

Un nombre premier est un entier naturel qui admet exactement deux diviseurs distincts entiers été positifs : 1 et lui-même.
ex : 2; 3; 5; 7; 11; 13; 17; 19; 23; 29; 31; 37; 41; 43; 47; 53; 59; 61 …

Un nombre semi-premier ou bi-premier est un entier naturel produit de deux nombres premiers, pas nécessairement distincts.
ex : 4; 6; 9; 10; 14; 15 …

Nombres premiers entre eux

En mathématiques, des entiers a et b sont dit premiers entre eux s'ils n'ont aucun facteur premier en commun; en d'autres termes, ils n'ont aucun diviseur autre que 1 et -1 en commun. On dit aussi que a est premier avec b.
De manière équivalente, ils sont premiers entre eux si et seulement si leur plus grand commun diviseur est égal à 1.
Par exemple, 6 et 35 sont premiers entre eux, mais 6 et 27 ne le sont pas parce qu'ils sont tous les deux divisibles par 3. 1 est premier avec tout entier; 0 est uniquement premier avec 1 et -1.

Inverse modulaire

L'inverse modulaire d'un entier a pour la multiplication modulo m est un entier x tel que : $ax \equiv 1 \bmod m$. C'est le même concept que la division pour les réels.

Exemple : l'inverse modulaire de 3 modulo 11 est 4 car $3 \times 4 = 12 \equiv 1 \bmod 11$.
On peut trouver une infinité d'autres entiers qui satisfont aussi cette congruence.
Il suffit d'ajouter à 4 des multiples de 11. Mais 4 est la seule solution < 11.

L'inverse de a modulo m existe si et seulement si a et m sont premiers entre eux.

Factorisation entière en nombres premiers

La décomposition d'un entier strictement positif en produit de facteurs premiers est toujours unique. Par définition, un nombre premier ne peut pas être décomposé.

Exemples :

11	= 11	
45	= 3 x 3 × 5	= 3^2 x 5
125	= 5 × 5 × 5	= 5^3
360	= 2 × 2 × 2 × 3 × 3 × 5	= $2^3 \times 3^2 \times 5$
1 001	= 7 × 11 × 13	

Quand les nombres sont très grands, on ne connaît pas d'algorithme de factorisation efficace. Tous les nombres ne sont pas aussi difficiles à factoriser, les plus durs étant les nombres [semi-premiers] résultants du produit de nombres premiers sensiblement de même taille. Dans ce cas, aucun algorithme n'est connu pour pouvoir le factoriser en temps polynômial. Les meilleurs algorithmes connus sont sous-exponentiels, mais super-polynômiaux. En particulier, le meilleur algorithme connu s'exécutant en temps asymptotique est le crible général de corps de nombres (GNFS).

En utilisant cet algorithme, il a fallu prés de 5 mois pour trouver la factorisation d'un nombre de 193 chiffres, en 2005, au moyen d'une grappe de 80 machines à processeurs Opteron à 2,2 GHz connectées par un réseau Gigabit.

En 2013, le plus grand nombre factorisé est long de 232 chiffres (RSA-768).

Groupe cyclique, générateur

On appelle groupe cyclique un groupe de cardinal fini dans lequel il existe un élément 'a' tel que tout élément x du groupe puisse s'exprimer sous forme d'une puissance de a. Cet élément a est appelé générateur du groupe.

Logarithme discret

En algèbre abstraite et dans ses applications, le logarithme discret est défini en théorie des groupes par analogie avec le logarithme ordinaire.

Les logarithmes discrets sont peut-être plus simples à comprendre dans le groupe cyclique $(Z_p)^x$ qui est l'ensemble {1, …, p−1} doté de la multiplication modulo le nombre premier p. Pour trouver la $k^{ième}$ puissance de l'un des nombres de ce groupe, ce qui se nomme «exponentiation discrète», il faut élever ce nombre à la puissance k et calculer le reste de la division par p.
Par exemple : $3^5 = 243$. Le reste de la division entière de 243 par 7 étant 5, 3^5 dans le groupe $(Z_7)^x$ est 5.

Le logarithme discret est le problème inverse : étant donné $3^k \equiv 5 \pmod 7$ quel est le plus petit 'k' pour lequel cette proposition est vrai ?

Petit théorème de Fermat (1636)

Le petit théorème de Fermat affirme que si p est un nombre premier,
alors pour tout entier a, $a^p \equiv a \pmod p$
Cela signifie que si vous considérez un entier a, et que vous le multipliez par lui-même p fois et que vous lui soustrayez a, le résultat est divisible par p. Il s'appelle le petit théorème de Fermat pour le différencier du dernier théorème de Fermat.

Le petit théorème de Fermat est généralisé par le théorème d'Euler.

Indicateur d'Euler

Si n est un entier plus grand que 2, l'indicateur d'Euler de n, noté $\varphi(n)$ désigne le nombre d'entiers compris entre 1 et n, et premiers avec n.

Si n est premier $\varphi(n) = n-1$.

Il découle de la définition que $\varphi(1) = 1$, et $\varphi(n)$ est $(p-1)p^{k-1}$ quand n est la $k^{ième}$ puissance d'un nombre premier p^k.

Si n est produit de 2 premiers, n = pq, alors $\varphi(n) = (p-1)(q-1)$.

De plus, si m et n sont premiers entre eux alors $\varphi(mn) = \varphi(m)\varphi(n)$

Théorème d'Euler

En théorie des nombres, le théorème d'Euler s'énonce comme suit :

Si n est un entier naturel et a est premier avec n, alors $a^{\varphi(n)} \equiv 1 \pmod n$.
C'est sur ce théorème que s'appuie l'algorithme RSA

RSA

on choisit p et q deux nombres premiers avec $p \neq q$
calcul de n = pq
calcul de $\varphi(n) = (p-1)(q-1)$
on choisit e premier avec $\varphi(n)$ tel que $1 < e < \varphi(n)$
calcul de d tel que $ed \equiv 1 \pmod{\varphi(n)}$

la clé publique se compose de :	la clé privée se compose de :
– *n* le modulo	– *n* le même modulo
– *e* l'exposant public (encryption exponent)	– *d* l'exposant privé (decryption exponent)

Dans la pratique, on conserve une forme différente de la clé privée pour permettre un déchiffrement plus rapide à l'aide du théorème des restes chinois :
– *p* et *q*, les nombres premiers choisis pour calculer n
– *d mod (p-1)* et *d mod (q-1)* (souvent nommés *dmp1* et *dmq1*)
– *(1/q) mod p* (souvent nommé *iqmp*)

Chiffrement : $C = M^e \pmod n$	Déchiffrement : $M = C^d \pmod n$

Exemple chiffrons '35' avec :
p = 23, q = 41
n = pq = 943, et $\varphi(n) = 22x40 = 880$
e premier avec 880, e = 7
d tel que $7d \equiv 1 \pmod{880}$, d = 503 ($7x503 = 3521 = 4(880)+1$)

C = encrypt(35) = $35^7 \pmod{943} = 545$
M = decrypt(545) = $545^{503} \pmod{943} = 35$

Grâce aux algorithmes probabilistes on sait produire des nombres premiers de plusieurs milliers de chiffres de long en peu de temps; le calcul de p et q est donc rapide. L'algorithme d'Euclide qui permet de calculer e et d est efficace. Au total, pour un ordinateur, la constitution du quadruplet (p, q, e, d) est donc peu couteux.

On remarque la valeur particulière $e = 65537 = 2^{16} + 1$ qui permet un calcul rapide de $M^e \bmod n$, c'est-à-dire d'un chiffré ou d'une vérification de signature.

Échange de clés Diffie-Hellman

Alice et Bob ont choisi un groupe (soit un corps fini, dont ils n'utilisent que la multiplication, soit une courbe elliptique) et une génératrice g de ce groupe.
Alice choisit un nombre au hasard a, élève g à la puissance a, et dit à Bob g^{a}.
Bob fait de même avec le nombre b.
Alice, en élevant le nombre reçu de Bob à la puissance a, obtient g^{ba}.
Bob fait le calcul analogue et obtient g^{ab}, qui est le même. Mais puisqu'il est difficile d'inverser l'exponentiation dans un corps fini, c'est-à-dire de calculer le logarithme discret, Ève ne peut pas découvrir a et b, donc ne peut pas calculer g^{ab}.

Alice				Bob		
Sec		Calc		Calc		Sec
	p, g				p, g	
a						b
		$g^{a} \pmod p$	→			
			←	$g^{b} \pmod p$		
		$(g^{b} \bmod p)^{a} \bmod p$	=	$(g^{a} \bmod p)^{b} \bmod p$		

Exemple

Alice et Bob choisissent un nombre premier p et une base g.

Dans notre exemple,	p=23 et g=3.
Alice choisit un nombre secret	a=6.
Elle envoie à Bob la valeur	$g^{a} \bmod p = 3^{6} \bmod 23 = 16$
Bob choisit à son tour un nombre secret	b=15
Bob envoie à Alice la valeur	$g^{b} \bmod p = 3^{15} \bmod 23 = 12$
Alice calcule maintenant la clé secrète	$(g^{b} \bmod p)^{a} \bmod p = 12^{6} \bmod 23 = 9$
Bob fait de même	$(g^{a} \bmod p)^{b} \bmod p = 16^{15} \bmod 23 = 9$

Paradoxe des anniversaires

Le paradoxe des anniversaires stipule que s'il y a 23 personnes dans un lieu, alors il y a légèrement plus de 50 % de chances qu'au moins deux personnes fêtent leur anniversaire le même jour.
À partir de 57 personnes, la probabilité est supérieure à 99%.
Cependant, il ne s'agit pas d'un paradoxe dans le sens de contradiction logique; c'est un paradoxe, dans le sens où c'est une vérité mathématique qui contredit l'intuition : la plupart des gens estiment que cette probabilité est très inférieure à 50%, parce qu'elles confondent avec la probabilité qu'une personne donnée fête son anniversaire le même jour qu'une autre.
Les gens pensent en général à la probabilité que 2 personnes soient nées en même temps un jour donné, probabilité qui est en effet très faible.
Pour s'en convaincre, il faut prendre le problème dans l'autre sens, et calculer la probabilité qu'aucune personne ne soit née le même jour qu'une autre. C'est exactement la proposition contraire, et sa probabilité vaut un moins la probabilité recherchée.

La probabilité que l'on recherche est donc : $1 - \prod_{i=0}^{N-1} \frac{365 - i}{365}$
avec N le nombre de personnes.

En effet, la première personne est forcément la seule à être née un jour donné. La deuxième a 364 chances sur 365 d'être née un autre jour que la première. La troisième n'a plus que 363 chances sur 365 de n'être née ni le même jour que la première, ni le même que la deuxième, etc. jusqu'à ce qu'on arrive au nombre total de personnes. On retranche ensuite le résultat à 1 pour obtenir la probabilité initiale.
En faisant l'application numérique, on trouve 50,72 % pour 23 personnes ... Le calcul exposé néglige la date du 29 février dont la prise en compte changerait très peu le pourcentage indiqué.

Le paradoxe des anniversaires est utilisé en cryptographie pour élaborer des attaques sur les fonctions de hachage. Il permet d'établir une estimation de la résistance de ces algorithmes aux collisions quelconques (deux messages qui produisent le même condensé). Plus concrètement, si une fonction de hachage a une sortie de N bits alors il faut environ $2^{N/2}$ messages distincts pour produire une collision complète avec 50% de chance; les sorties de la fonction pouvant être comparées à des personnes avec des anniversaires se répartissant sur 2^{N} valeurs.

Théorème des restes chinois

D'après D. Wells, le problème suivant fut posé par Sun Tsu Suan-Ching (4ème siècle ap J.-C.) :

- Il y a de certaines choses dont le nombre est inconnu. À plusieurs reprises divisé par 3, le reste est 2; par 5 le reste est 3; et par 7 le reste est 2. Quel est ce nombre ?
- Une vieille femme allait au marché lorsqu'un cheval marcha sur son panier et cassa tous ses oeufs. Le cavalier offrit alors de payer les dégâts et lui demanda combien d'oeufs elle avait emporté. Elle ne s'en rappelait pas le nombre exact, mais se souvenait que, quand elle les avait pris 2 par 2, il en était resté un. Il advenait de même quand elle les prenait par trois, quatre, cinq ou six à la fois. Par contre, pris par 7, il n'en restait pas. Quel est le plus petit nombre d'œufs qu'elle pût avoir ?
- Une bande de 17 pirates s'est emparée d'un butin composé de pièces d'or d'égale valeur. Ils décident de se les partager également, et de donner le reste au cuisinier chinois. Celui-ci reçoit alors 3 pièces. Mais les pirates se querellent, et 6 d'entre eux sont tués. Après un nouveau partage du même butin, le cuisinier reçoit 4 pièces. Dans un naufrage ultérieur, seuls le butin, 6 pirates et le cuisinier sont sauvés, et le partage donne alors 5 pièces d'or à ce dernier. Quelle est la fortune minimale que peut espérer le cuisinier s'il décide d'empoisonner les derniers pirates?

Les problèmes de ce genre sont tous des exemples de ce qui devint universellement connu sous le nom du théorème des restes chinois.
Mathématiquement parlant, il se réduit à trouver n connaissant le reste de sa division par différents nombres.

Il donne une formule directe pour obtenir la solution d'un système de congruence d'entier lorsqu'ils sont premiers entre eux. Sinon il faut utiliser la méthode des substitutions successives.

Corps finis

Un corps est un ensemble muni d'une opération d'addition et d'une opération de multiplication, qui se comportent «comme» l'addition et la multiplication des nombres réels : commutativité, associativité, distributivité (*qui permet de passer d'un produit de somme à une somme de produit*). En particulier, dans tout corps il y a un élément neutre pour l'addition et absorbant pour la multiplication, noté 0, ainsi qu'un élément neutre pour la multiplication, noté 1, et tout élément différent de 0 admet un inverse.

Un corps fini est simplement un corps ayant un nombre fini d'éléments, et ce nombre 'q' (*aussi appelé sa cardinalité*) est toujours une puissance (*positive*) 'm' d'un nombre premier 'p' (*appelé sa caractéristique*) : $q = p^m$.
Il existe par exemple des corps à 2, 3, 4, 5, 7, 8, 9, 11 éléments, mais pas de corps de cardinalité 6, 10 ou 12 (*on ne peut pas toujours simplifier des produits ni a fortiori diviser modulo m en général. Par exemple 3 × 2 = 0 mod 6 donc 2 et 3 n'ont pas d'inverse modulo 6*).

Les corps finis sont aussi appelés Corps de Galois.
Le corps fini de cardinal q (= p^m) est noté F_q ou GF(q) (Galois Field).

Si m = 1 alors F est appelé un corps premier. Les éléments de F_{29} sont { 0, 1, 2, ...,28 }.
Si m ≥ 2, alors F est appelé un corps d'extension, et on distingue les corps dits binaires : $q = 2^m$, particulièrement adaptés à l'architecture des ordinateurs.

Les éléments de Fp^m peuvent être représentés comme des polynômes de degré strictement inférieur à m sur Fp. Illustration :
Les éléments du corps binaire $F_2{}^4$ sont les 16 polynômes binaires de degré au plus 3 :

0	x^2	x^3	$x^3+ x^2$
1	x^2+1	x^3+1	$x^3+ x^2+1$
x	x^2+x	x^3+x	$x^3+ x^2+x$
x+1	x^2+x+1	$x^3+ x^2+1$	$x^3+ x^2+x+1$

Dans le cas des corps binaires : les éléments de $F_2{}^m$ peuvent être codées sous forme d'un vecteur de m bits numérotés de m-1 a 0 (l'ordre décroissant correspond aux expressions « bits de poids forts » et « bits de poids faibles »)

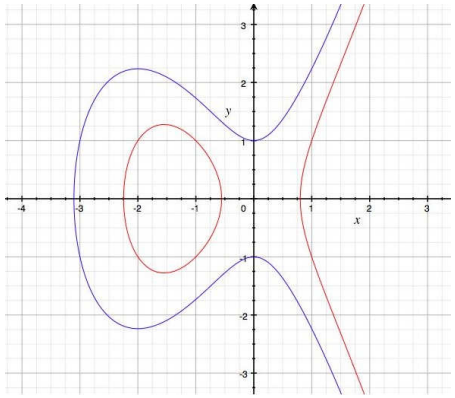
Chaque bit représente un terme du polynôme.

Par exemple $x^6 + x^4 + x + 1$ peut se représenter par : {01010011}, soit la valeur hexadécimale 0x53.

Dans ce cas, l'addition se résume à un simple XOR.
Exemple :
$(x^2 + 1) + (x^2 + x) = 2x^2 + x + 1$, qui réduit modulo 2 donne : $0x^2 + x + 1 = x + 1$
En utilisant la représentation binaire :
{101} + {110} = {011}

ECC – Elliptic Curve Cryptography

Courbes elliptiques d'équations
$y^2 = x^3 + 3x^2 + 1$ *(en bleu)*
et
$y^2 = x^3 + 2x^2 – x – 1$ *(en rouge)*



Les courbes elliptiques sont un sujet très à la mode en mathématiques. Elles sont à la base de la démonstration du grand théorème de Fermat par Andrew Wiles. Elles sont aussi à l'origine de nouveaux algorithmes de cryptographie très sûrs, et on entrevoit les prémices de leur utilisation pour la factorisation de grands nombres entiers.

Les courbes elliptiques permettent l'adoption de nouveaux crypto systèmes. Mais elles mettent aussi en danger des crypto systèmes existants, comme le RSA. Elles sont en effet à l'origine d'une méthode élégante de factorisation, due à H.W.Lenstra.
Dans la pratique, pour factoriser les entiers qui interviennent dans le RSA, l'algorithme est un peu moins performant que le crible quadratique, mais il est très efficace quand il s'agit de factoriser un entier qui a un facteur premier relativement petit.

Une courbe elliptique à pour équation générale : $y^2 = x^3 + ax + b$

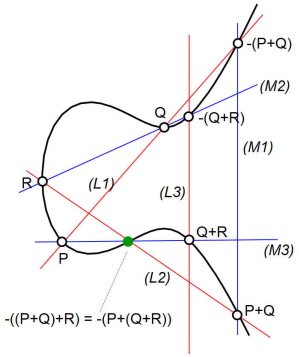
En cryptographie on s'intéresse surtout aux courbes elliptiques sur des corps finis, particulièrement :

- celles définies sur un corps premier F_p (prime field).
  Les éléments de F_p sont les entiers entre 0 et p-1, ou p est un nombre premier.
  $y^2 \bmod p = x^3 + ax + b \bmod p$
  La courbe contient tous les points (x, y) qui satisfont l'équation.
  (x, y, a, b sont dans F_p)

- celles définies sur un corps binaire $F_2{}^m$ (binary field).
  Les éléments de $F_2{}^m$ sont les entiers dont la taille ne dépasse pas m bits.
  $y^2 + xy = x^3 + ax^2 + b$ ($b \neq 0$)
  (x, y, a, b sont dans $F_2{}^m$)

Une fois doté d'un ensemble de points, et d'un point spécial, 'à l'infini' on peut définir les opérations d'addition et de multiplication (de points) :

note : une courbe elliptique étant défini par un polynôme de degré 3, une droite ne la coupe qu'en au plus 3 points.



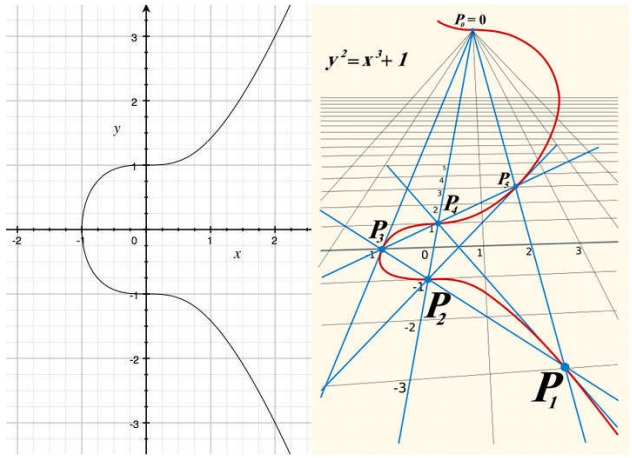
Addition : la droite passant par 2 points en détermine un troisième, ailleurs sur la courbe.

Doublement (ajout d'un point à lui-même) : la droite à considérer est la tangente en ce point.

Multiplication (un point multiplié par un nombre) : réalisé en pratique par la répétition des opérations précédentes.

Donc, avec un point g (appelons le 'générateur') :
0g = 'à l'infini', 1g = g, 2g = g + g, …

En continuant ainsi, on finira par trouver un nombre 'n', tel que n × g = 'à l'infini'. Ce nombre n est appelé l'ordre de g. Avec de la chance, on a choisit une courbe et un générateur qui ont permis de parcourir tous les points de la courbe, bien que ce soit rarement le cas. Le nombre total de points est divisible par n, le résultat étant nommé le cofacteur.



Deux images de la courbe elliptique d'équation $y^2=x^3+1$.
A droite, le point à l'infini est représenté ainsi que la relation P3 + P4 = P1 = −P5.

La sécurité du système dépend de l'incapacité à déterminer k connaissant le point k × g.
C'est le problème du logarithme discret sur courbe elliptique.
Le nombre k sera la clé privée, et le point 'k × g' la clé publique.

http://fr.wikipedia.org/wiki/Courbe_elliptique

Logarithme discret : mai 2014, des chercheurs français trouvent un algorithme plus efficace.

Bien que cette avancée ne mette pas en danger immédiat les crypto-systèmes existants, des tailles de problèmes qui semblaient insolvables sont maintenant à porté de ressources de calcul raisonnables. L'avancée récente déplace le problème de 'sous-exponentiel' vers 'quasi-polynomial'.

En pratique, les conséquences varient en fonction de la nature exacte du problème et de la taille des données en entrée, mais dans un exemple étudié par les chercheurs, les calculs sont passés de 2^{80} à 2^{70} opérations, ce qui est environ 1000 fois plus rapide.
La même configuration d'ordinateur à qui il aurait auparavant fallu 3 ans pour déchiffrer le message y arriverait en moins de 24 heures.

Malheureusement ou heureusement, l'algorithme ne s'applique pas à toutes les situations reposant sur le logarithme discret.

Certains scientifiques songent aussi à explorer cette voie pour s'attaquer au problème de la factorisation des entiers, dont les liens avec le logarithme discret existent. Des améliorations de l'algorithme sont également envisageables.

http://m.cacm.acm.org/magazines/2014/1/170850-french-team-invents-faster-code-breaking-algorithm/fulltext

La principale notion mathématique dans le calcul du coût d'un algorithme précis sont les notions de négligeabilité *(notée o(f(n)), «petit o»)* et de domination *(notée O(f(n)), «grand o»)*, où *f* est une fonction mathématique de *n*, variable désignant la quantité d'informations (en bits, en nombre d'enregistrements…) manipulée dans l'algorithme.
En algorithmique on trouve souvent des complexités du type :

- $O(1)$, indépendant de la taille de la donnée
- $O(log(n))$, complexité logarithmique
- $O(n)$, complexité linéaire
- $O(n.log(n))$, complexité quasi-linéaire
- $O(n^2)$, complexité quadratique
- $O(n^3)$, complexité cubique
- $O(n^p)$, complexité polynômiale
- $O(n^{log(n)})$, complexité quasi-polynômiale
- $O(p^n)$, complexité exponentielle
- $O(n!)$, complexité factorielle