LE CHIFFREMENT EXPLIQUE A MON VOISIN QUI N'Y CONNAIT RIEN MAIS QUI VOUDRAIT SAVOIR

Par Gérard Peliks
Président de l'atelier sécurité et VP de Forum ATENA

Mai 2017

La cryptologie expliquée à mon voisin qui n'y connait rien mais qui voudrait savoir. Devenez accros, même si vous n'avez aucune notion aujourd'hui de la science des messages cachés. Avec un peu d'attention, l'ensemble vous semblera limpide. Allez, accrochez-vous, on se lance ! :

Pour illustrer, un conseil pratique se trouve à la fin de chaque chapitre.

Certaines notions peuvent paraître un peu complexes la première fois qu'on les aborde, mais en fait tout est très simple quand on a pris du recul en s'étant penché suffisamment sur les problèmes posés.

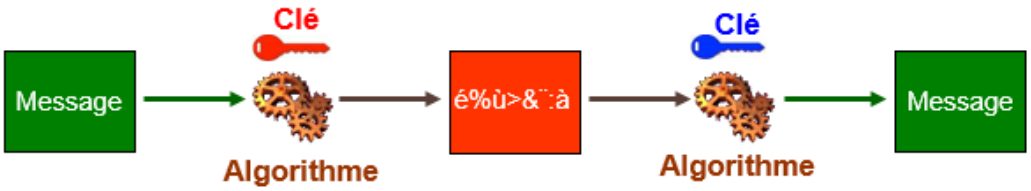


Table des matières

CHIFFRER, DECHIFFRER, DECRYPTER

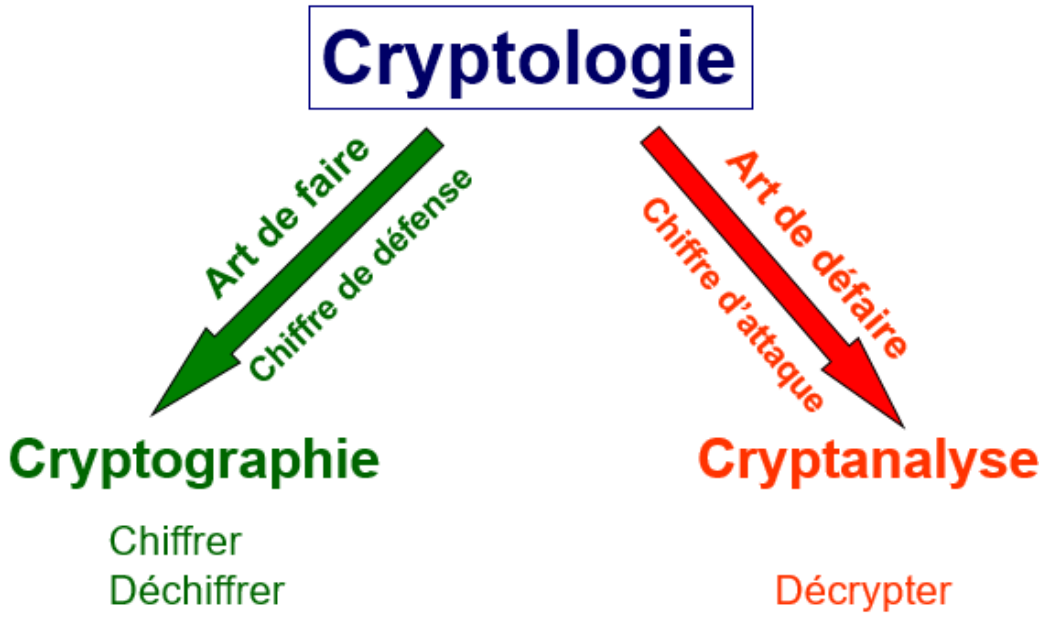
CRYPTOLOGIE : CRYPTOGRAPHIE ET CRYPTANALYSE

La **cryptologie** ou science des messages cachés se divise en deux écoles antagonistes mais liées : La **cryptographie** et la **cryptanalyse**.

Dans la cryptographie, ou « *chiffre de défense* », **on chiffre** une information, pour la protéger des regards, en la brouillant à travers un algorithme et une clé, et on permet, à qui est autorisé à la lire, de la **déchiffrer** en utilisant le même algorithme et une clé. L'algorithme opère sur le message des permutations, des substitutions, des transformations diverses en fonction du contenu de la clé. On parle de **chiffrement** et de **déchiffrement**.

Dans la cryptanalyse, ou « *chiffre d'attaque* », on détient le message chiffré mais pas la clé pour le déchiffrer. On peut parvenir avec d'autres méthodes à retrouver le message en clair en le **décryptant**. On parle de **décryptement**.

UN MOT A BANNIR DE VOTRE VOCABULAIRE



Vous avez remarqué que je n'ai pas utilisé le mot si répandu : « crypter », ni les mots dérivés : « cryptage » et « cryptement ». Pourquoi ?

Parce ce que ces mots ne veulent rien dire ! J'irai jusqu'à affirmer que ce sont des mots faux et à bannir de notre vocabulaire car ils affaiblissent notre belle langue française. Car si déchiffrer est bien l'opération inverse de chiffrer ; décrypter aurait quelle opération inverse ??? **Aucune** et décrypter n'a d'ailleurs pas besoin d'opération inverse. Décrypter, c'est trouver en clair un message chiffré sans posséder la clé pour le déchiffrer. Déchiffrer, c'est retrouver en clair un message chiffré en possédant la clé de déchiffrement. Décrypter n'a pas d'inverse. Donc, en utilisant le mot « crypter », vous admettez implicitement que déchiffrer et décrypter, c'est la même chose alors que c'est tout l'inverse, vu sous l'angle de la cryptologie.

Cette mise au point est-elle importante ? La réponse est **OUI**. Quand vous utilisez le mot « crypter », au mieux vous disqualifiez votre image de cryptologue (en herbe), au pire vous collez au plafond un vrai cryptologue, tant ce mot, qui n'est hélas que trop répandu, lui fait horreur.

Cette mise au point faite, comment s'appellent les acteurs de la cryptologie ?

LES PHARES QUI NOUS ECLAIRENT

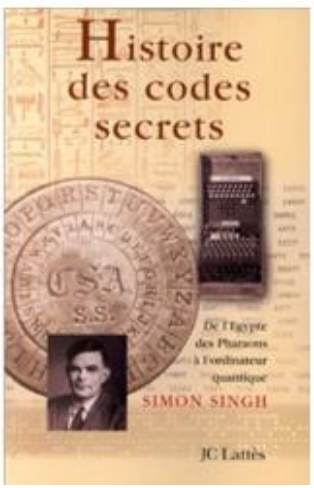
Ceux qui conçoivent les méthodes du chiffre d'attaque, c'est-à-dire les moyens de décrypter les messages chiffrés sans avoir la clé de déchiffrement, sont

les **cryptanalystes**. Citons des grands noms, parmi bien d'autres : L'Anglais Charles Babbage (19eme siècle) qui a décrypté le chiffre de Vigénère, 150 ans quand même après que le Français Blaise de Vigénère l'ait mis au point ! Citons aussi le capitaine **Georges Painvin** qui a trouvé le moyen de décrypter le chiffrement ADFVGX utilisé par les Allemands durant la première guerre mondiale. Grace à ce cryptanalyste de génie, l'armée française, connaissant l'angle d'attaque, la date et l'heure de l'offensive allemande de ce qui allait s'appeler « la deuxième bataille de la Marne » en 1918, a pu la stopper avant que l'armée allemande ne pousse vers Paris. Ce qui a beaucoup fait pour la victoire. Citons enfin, durant la 2eme guerre mondiale, notre maître à tous, **Alan Turing et son équipe** de Bletchley Parc, qui, en décryptant le chiffrement de la machine allemande Enigma a permis aux Britanniques de gagner la bataille de l'Atlantique, les batailles du désert, a raccourci la guerre de deux ans, et favorisé la victoire sur la barbarie nazie. Vous voyez pourquoi la cryptanalyse s'appelle aussi « le chiffre d'attaque » ?

Ceux qui conçoivent les méthodes du chiffre de défense, c'est-à-dire les algorithmes de chiffrement / déchiffrement et déterminent la longueur nécessaire des clés s'appellent les **cryptologues**. Notez qu'on aurait pu les appeler logiquement les cryptographes mais ce mot est plutôt réservé aux outils de chiffrement / déchiffrement. Nous appellerons ceux qui font les méthodes du chiffre de défense, les cryptologues, et c'est normal puisque non seulement un cryptologue fait avancer la cryptographie, mais aussi il teste la solidité et l'efficacité de ses algorithmes, en les soumettant à des méthodes de cryptanalyse. Donc cryptographie plus cryptanalyse constituant la cryptologie, ce sont bien des cryptologues.

Parmi ces cryptologues, citons **Blaise de Vigénère**, citons **Joan Daemen** et **Vincent Rijmen**, deux Belges qui ont conçu l'AES, très utilisé dans le chiffrement symétrique dont nous reparlerons, citons les incontournables **Rivest**, **Shamir** et **Adelman** qui ont conçu le RSA, très utilisé dans le chiffrement asymétrique (mais pour rétablir la vérité, les anglais **Ellis** et **Cocks** y sont aussi pour quelque chose), et n'oublions pas **Diffie** et **Hellman** qui ont montré que la génération spontanée d'une même clé, partagée à deux endroits du monde, sans pour autant s'échanger de secrets, était possible. Nous y reviendrons aussi. Vous aurez compris que la cryptologie est un combat des chefs (et quels chefs !). D'un côté vous avez les cryptologues qui conçoivent des méthodes de chiffrement toujours plus solides pour rendre difficile le décryptement et de l'autre les cryptanalystes qui bravent cette difficulté en concevant des méthodes pour décrypter les messages chiffrés, même sans la clé de déchiffrement.

LE CONSEIL CRYPTO :



Voici de la lecture en français pour pénétrer dans la pensée des grands cryptologues et des grands cryptanalystes d'hier et d'aujourd'hui, mais il y a bien sûr beaucoup d'autres ouvrages à conseiller :

- Le portail de Wikipédia sur la cryptologie
  http://fr.wikipedia.org/wiki/Portail:Cryptologie
- Le portail de l'ANSSI : www.ssi.gouv.fr

- Le livre de Simon Singh : Histoire des codes secrets :
  http://www.livredepoche.com/histoire-des-codes-secrets-simon-singh-97822...
- Le livre de Jacques Stern : La science du secret :
  http://www.odilejacob.fr/catalogue/sciences/mathematiques/science-du-secret_9782738105332.php
- Le livre d'Hervé Lehning : L'Univers des codes secrets
  http://www.ixelles-publishing.com/livre.php?ean=9782875151568

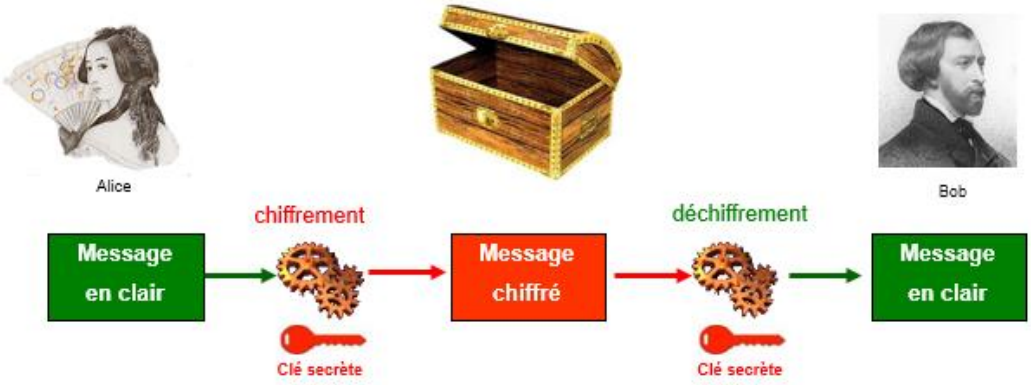
La cryptologie reste une discipline complexe, mais tellement merveilleuse. On touche aux confins du génie humain qui est sans limites ! Comme l'a dit Albert Einstein, « Il faut rendre les choses aussi simples que possible, mais pas plus simples, sinon on les dénature ».

Nous allons décrire tout d'abord deux types de chiffrement, le chiffrement symétrique utilisé pour chiffrer et le chiffrement asymétrique utilisé principalement pour acheminer la clé de chiffrement symétrique.

LE CHIFFREMENT SYMETRIQUE OU CHIFFREMENT « A CLE SECRETE »

Le **chiffrement symétrique**, dit aussi chiffrement à clé secrète met en jeu **une** clé et un algorithme. On chiffre avec la clé à l'aide de l'algorithme, on déchiffre avec la même clé et le même algorithme. La clé de chiffrement est donc **la même** que la clé de déchiffrement, c'est pourquoi le chiffrement est dit « symétrique ».

La clé doit être gardée secrète, mais être aussi, bien sûr, transmise à celui qui est autorisé à déchiffrer. Comment faire passer cette clé, surtout à travers un réseau public non protégé comme l'Internet, en devant toujours garder cette clé secrète, sauf pour celui auquel le message est destiné ? C'est le premier problème, mais nous allons voir que garder secrète la clé pose également d'autres problèmes.



LES MECANISMES ET LES STANDARDS

Le chiffrement symétrique tire parti du « ***ou exclusif*** ». L'opération « ou exclusif », que nous noterons ici ⊕ entre deux bits (qui prennent la valeur 0 ou 1) donne « **1** » si les deux bits sont différents et « **0** » s'ils sont les mêmes. Ainsi : 0 ⊕ 0 = **0** ; 1 ⊕ 1 = **0** ; 0 ⊕ 1 = **1** et 1 ⊕ 0 = **1**.

Par exemple faisons traiter la donnée en clair : **0 0 1 0 0 1 1 0 1**, bit à bit, dans l'algorithme « *ou exclusif* » avec la clé de chiffrement symétrique : **0 1 1 1 0 1 0 1 0 cela** donne la donnée chiffrée :

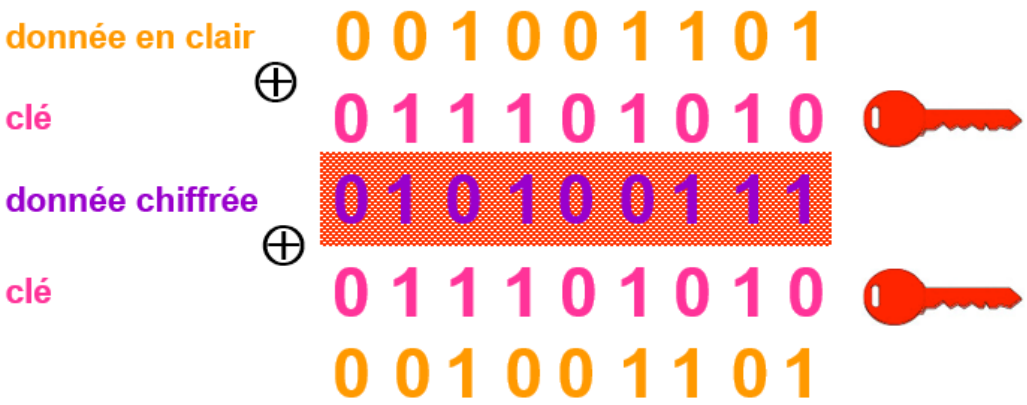
(0 0 1 0 0 1 1 0 1) ⊕ (0 1 1 1 0 1 0 1 0) = 0 1 0 1 0 0 1 1 1

(Le premier bit de la donnée en clair, ici « 0 » et de la clé, ici « 0 », donne 0 ⊕ 0 = 0, qui est premier bit de la donnée chiffrée.

Maintenant faisons un « *ou exclusif* », bit à bit, entre la donnée chiffrée et la clé :

(0 1 0 1 0 0 1 1 1) ⊕ (0 1 1 1 0 1 0 1 0) = 0 0 1 0 0 1 1 0 1

Et on constate qu'on a bien retrouvé la donnée en clair. C'est cette fonction du ⊕ qui est à la base du chiffrement symétrique.



Une vision très (trop) simplifiée du chiffrement symétrique consisterait à prendre une clé, de longueur mettons 256 bits, de découper le message à chiffrer en blocs de 256 bits et de faire un « *ou exclusif* », bit à bit, entre chaque bloc et la clé, ce qui donne le message chiffré. Sans connaître la clé, il n'est vraiment pas facile de décrypter le message chiffré !

A l'inverse, pour déchiffrer, connaissant la clé, on découpe le message chiffré en blocs de 256 bits et on fait un « *ou exclusif* » bit à bit, de chaque bloc avec la clé secrète, ce qui redonne le message en clair. Ne connaissant pas la clé, il peut sembler impossible, à partir du message chiffré, de reconstituer le message en clair, sauf à essayer toutes les combinaisons possibles de clés, ce qui nécessiterait, avec les moyens de calculs actuels, de très nombreux milliards d'années, pour une clé de 256 bits.

Mais si l'idée décrite précédemment est plaisante, son implémentation pêche car connaissant juste la longueur de la clé de chiffrement, et en faisant par exemple des « *ou exclusifs* » entre plusieurs blocs du message chiffré, les cryptanalystes peuvent fragiliser la clé, voire finir par retrouver le message en clair. Alors dans les algorithmes de chiffrement symétrique, on complique le calcul en faisant dépendre chaque bloc à chiffrer du résultat chiffré du bloc précédent. On parle de « *chiffrement symétrique par blocs chainés* ».

Précisons qu'il existe un chiffrement qui est prouvé être mathématiquement sûr et il est symétrique, c'est le **chiffre de Vernam**, dit encore méthode du masque jetable, dans lequel la longueur de la clé est égale à la longueur du message, et la clé n'est utilisée qu'une seule fois. Bien entendu, avec le chiffre de Vernam, le message à chiffrer n'est pas découpé en blocs, puisque la clé est de la même taille que le fichier à chiffrer. On parle de chiffrement par flot. Ceci dit, si le message est très long, le chiffre de Vernam n'est pas l'idéal dans une utilisation pratique.

Le standard de chiffrement le plus utilisé aujourd'hui est l'**AES** (Advanced Encryption Standard). Ce protocole a été proposé, à la fin des années 90, par deux cryptologues Belges, Joan Daemen et Vincent Rijmen, en réponse à un concours organisé par le NIST - National Institute of Standards and Technology – organisme américain. Plus léger, consommant moins d'énergie que les chiffrements symétriques qui étaient utilisés à l'époque, et très sûr, l'AES remplace les vénérables DES et 3DES, comme algorithme de chiffrement symétrique.

La clé de chiffrement/déchiffrement de l'AES a une longueur de 128, 192 ou 256 bits, au choix, et les calculs s'effectuent sur des blocs de 128 bits. À ce jour, l'algorithme AES n'a jamais été cassé. La seule solution possible pour les cryptanalystes consiste à essayer toutes les clés jusqu'à trouver celle qui, à partir d'un message chiffré, semble donner un message en clair. Mais le nombre de clés à essayer est considérable. Avec une clé de 256 bits, il y a 2^{256} combinaisons possibles, ce qui représente un nombre 1

suivi de 77 zéros ! L'algorithme AES est considéré comme très sûr, encore faut-il que ce chiffrement soit bien implémenté.

LE PROBLEME DE LA GENERATION DES CLES

Si l'algorithme AES est sûr, son implémentation doit l'être aussi. Pour cela, la clé secrète générée par celui qui veut chiffrer doit remplir plusieurs conditions.

LA CLE SECRETE DOIT EVIDEMMENT RESTER SECRETE, SAUF POUR LE DESTINATAIRE

Pour la faire parvenir à celui qui est autorisé à déchiffrer, une des méthodes est de la remettre au destinataire en main propre, ce qui se faisait, dans le temps par la valise diplomatique. Mais dans le cyberespace, où celui qui chiffre et celui qui déchiffre peuvent être très éloignés, il faut la faire transiter par un réseau, le plus souvent par Internet. Si elle transite en clair, la clé sera interceptée et utilisée, donc elle doit être chiffrée. Nous verrons dans le prochain chapitre que le chiffrement asymétrique (clé privée / clé publique) sera une solution. La physique quantique apporte une autre solution.

LA CLE DOIT ETRE GENEREE ALEATOIREMENT

Le problème est que l'état présent d'un ordinateur conditionne l'état où il sera le moment suivant. Ainsi, un ordinateur est une machine déterministe. Il existe des programmes de génération d'aléas mais le résultat de leurs calculs ne peut être que pseudo aléatoire. En effet un ordinateur, utilisant un programme de génération de clés aléatoires, a tendance à reproduire les mêmes séries de clés quand il en génère de grands nombres. Nous verrons aussi que la génération d'aléas purs peut se faire grâce aux propriétés de la physique quantique.

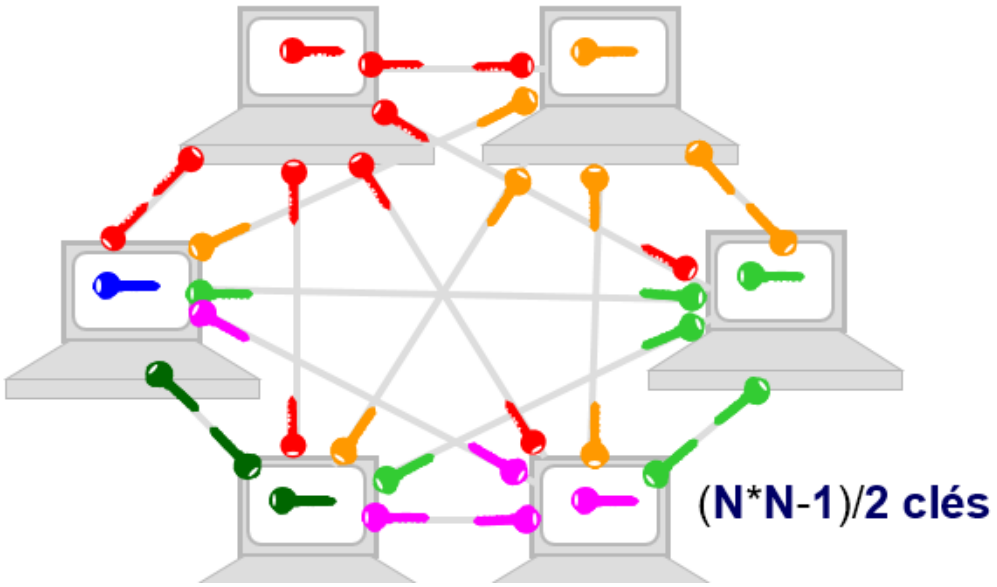
LA CLE DOIT ETRE RENOUVELEE PERIODIQUEMENT

Bien évidemment, si la clé a une durée de vie trop longue, la tâche des cryptanalystes qui vont tenter de la retrouver par force brute ou par d'autres moyens s'en trouvera simplifiée. L'idéal est de la changer à chaque transaction.

LA CLE DOIT AVOIR UNE TAILLE MINIMUM

Une clé trop courte peut être trouvée par attaque en force brute (on essaie toutes les clés possibles). Avec l'AES qui impose des clés de longueur au moins égale à 128 bits, le problème ne se pose pas aujourd'hui.

LE PROBLEME DE LA MULTIPLICATION DES CLES A ECHANGER



Si A veut chiffrer à destination de B, une clé symétrique entre A et B suffit. Si A, B, et C veulent s'échanger des informations chiffrées (et si B doit pouvoir déchiffrer le message de A, mais pas celui de C) il faut une clé entre A et B, une clé différente entre A et C, et une troisième clé entre B et C. Si n personnes veulent s'échanger des informations chiffrées, il faut n(n-1)/2 clés. Précisons que si 1000 personnes veulent s'échanger des informations chiffrées, cela mettra en jeu 499500 clés qui doivent bien évidemment rester secrètes !

EN CONCLUSION

Les algorithmes de chiffrement symétrique sont très sûrs, et de plus très rapides, c'est donc ce type de chiffrement qui est utilisé pour assurer la confidentialité des informations. Mais il faut savoir échanger les clés entre ceux qui ont droit de déchiffrer les informations ainsi chiffrées. Le problème d'échange de la clé secrète sur un réseau non sûr comme Internet, sera abordé dans un prochain chapitre.

LE CONSEIL CRYPTO :

Pour montrer une utilisation pratique du chiffrement d'un message, nous allons utiliser en écriture le logiciel de messagerie Thunderbird, qui fait partie de la suite des outils de la fondation Mozilla, auquel est ajouté le logiciel libre de chiffrement Enigmail qui utilise le GPG, version libre et gratuite du PGP. Nous soulignons comment, une fois ces logiciels installés, il est facile de chiffrer et de déchiffrer les messages.

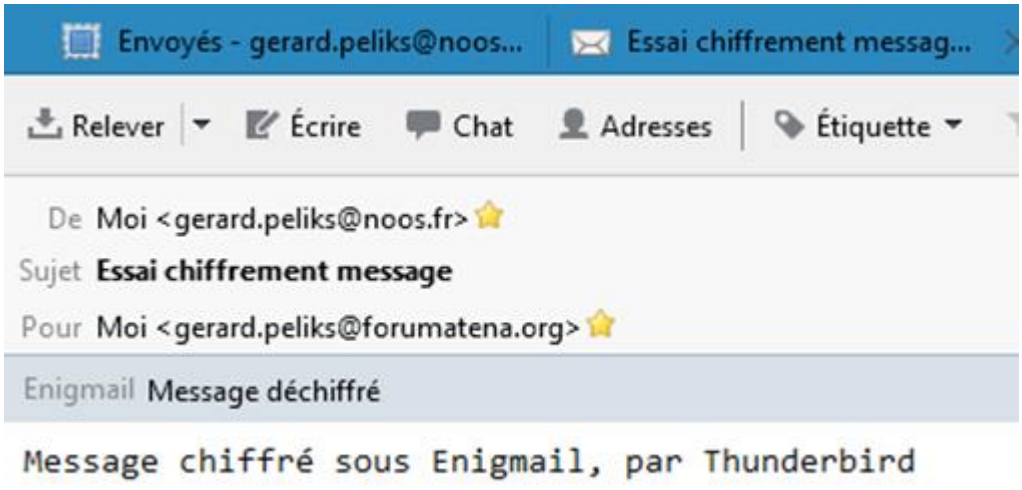
EN EMISSION DU MESSAGE

L'image qui suit est un message écrit sous le logiciel client de messagerie Thunderbird auquel est ajouté le logiciel de chiffrement GPG (qui est installé avec le logiciel libre Enigmail).

Vous remarquerez dans la troisième ligne, au-dessus de « Expéditeur », le cadenas qui est fermé quand on souhaite que le message soit chiffré, sinon, par défaut, le cadenas est représenté ouvert et le message part en clair.

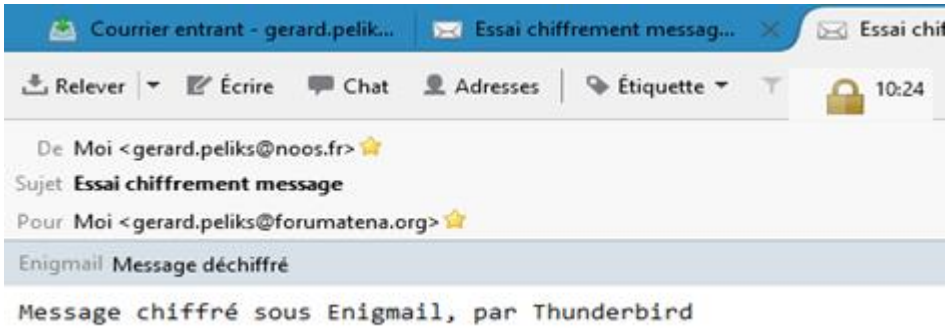
Le message à envoyer ici est : « Message chiffré sous Enigmail, par Thunderbird ».

Dans l'image ci-dessous, tirée de mon dossier « éléments envoyés » voici le message envoyé par Thunderbird. La clé publique du destinataire est utilisée pour chiffrer la clé secrète générée par l'expéditeur. Donc il faut au préalable avoir demandé au destinataire, son certificat, qui contient sa clé publique, et l'avoir intégré dans Enigmail. Seul le destinataire pourra déchiffrer le message car il est le seul à posséder la clé privée, mathématiquement liée à la clé publique, donc il est le seul à pouvoir déchiffrer la clé secrète générée par l'expéditeur, qui a chiffré le message. Ceci sera expliqué en détail dans un prochain chapitre. Ça peut paraître compliqué mais c'est en fait très simple.

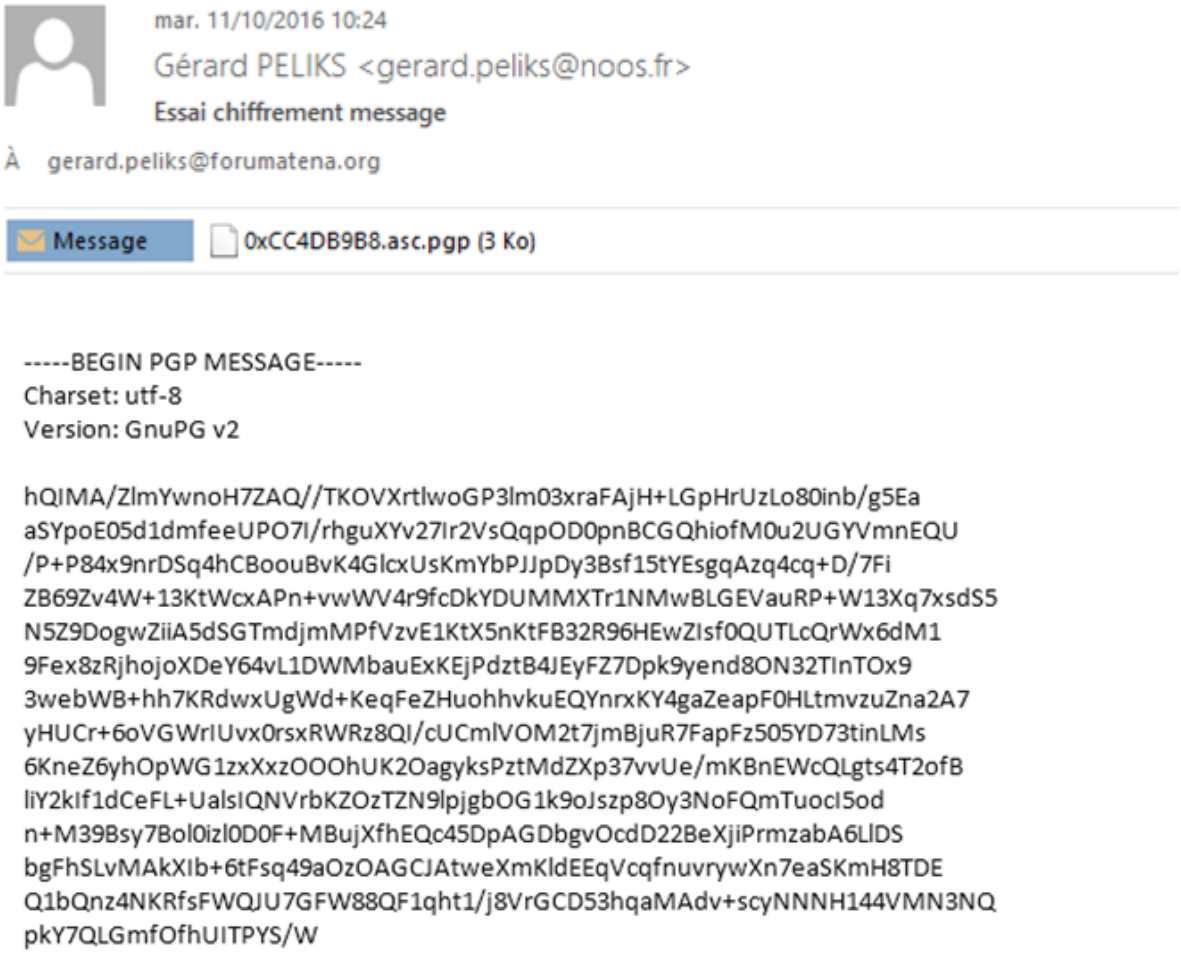


EN RECEPTION DU MESSAGE

Dans l'image ci-dessous, le message reçu est déchiffré automatiquement sous Thunderbird par Enigmail car le logiciel GPG est intégré à Enigmail



Dans l'image ci-dessous, le message est reçu sous Outlook. Il reste chiffré car aucun logiciel n'est intégré ici, à Outlook, pour le déchiffrer.



mar. 11/10/2016 10:24

Gérard PELIKS <gerard.peliks@noos.fr>

Essai chiffrement message

À gerard.peliks@forumatena.org

Message 0xCC4DB9B8.asc.pgp (3 Ko)

```
-----BEGIN PGP MESSAGE-----
Charset: utf-8
Version: GnuPG v2

hQIMA/ZlmYwnoH7ZAQ//TKOVXrtlwoGP3lm03xraFAjH+LGpHrUzLo80inb/g5Ea
aSYpoE05d1dmfeeUPO7l/rhguXYv27Ir2VsQqpOD0pnBCGQhiofM0u2UGYVmnEQU
/P+P84x9nrDSq4hCBoouBvK4GlcxUsKmYbPJJpDy3Bsf15tYEsgqAzq4cq+D/7Fi
ZB69Zv4W+13KtWcxAPn+vwWV4r9fcDkYDUMMXTr1NMwBLGEVauRP+W13Xq7xsdS5
N5Z9DogwZiiA5dSGTmdjmMPfVzvE1KtX5nKtFB32R96HEwZIsf0QUTLcQrWx6dM1
9Fex8zRjhojoXDeY64vL1DWMbauExKEjPdztB4JEyFZ7Dpk9yend8ON32TInTOx9
3webWB+hh7KRdwxUgWd+KeqFeZHuohhvkuEQYnrxKY4gaZeapF0HLtmvzuZna2A7
yHUCr+6oVGWrIUvx0rsxRWRz8QI/cUCmlVOM2t7jmBjuR7FapFz505YD73tinLMs
6KneZ6yhOpWG1zxXxzOOOhUK2OagyksPztMdZXp37vvUe/mKBnEWcQLgts4T2ofB
liY2kIf1dCeFL+UalsIQNVrbKZOzTZN9lpjgbOG1k9oJszp8Oy3NoFQmTuocI5od
n+M39Bsy7Bol0izl0D0F+MBujXfhEQc45DpAGDbgvOcdD22BeXjiPrmzabA6LlDS
bgFhSLvMAkXIb+6tFsq49aOzOAGCJAtweXmKldEEqVcqfnuvrywXn7eaSKmH8TDE
Q1bQnz4NKRfsFWQJU7GFW88QF1qht1/j8VrGCD53hqaMAdv+scyNNNH144VMN3NQ
pkY7QLGmfOfhUITPYS/W
```

LE CHIFFREMENT A CLE PUBLIQUE

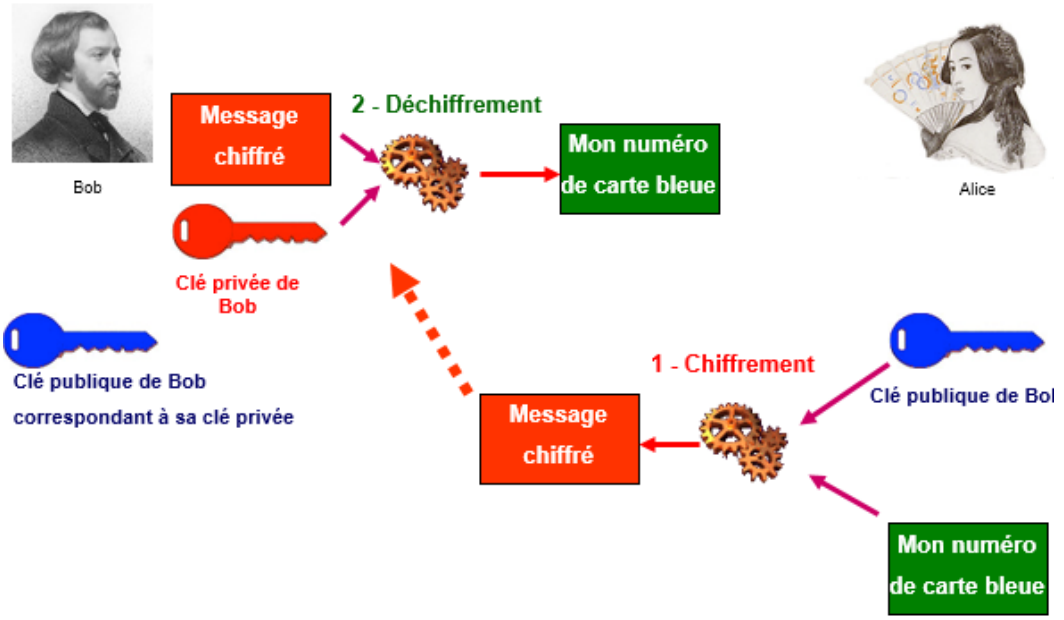
CLE PUBLIQUE, CLE PRIVEE

Le chiffrement à clé publique, dit aussi chiffrement asymétrique, fait intervenir un couple de clés mathématiquement liées. Quand on chiffre un message avec une des deux clés, on déchiffre le message avec l'autre.

Une des clés est dite « clé publique », elle n'est pas un secret et peut être donnée à tout le monde. L'autre clé, mathématiquement liée à la clé publique est par contre un secret que son possesseur doit absolument ne pas révéler, on l'appelle la « clé privée ».

Donc dans le chiffrement asymétrique, quand on chiffre avec la clé privée, on ne peut déchiffrer qu'avec la clé publique correspondante. Quand on chiffre avec la clé publique, on ne peut déchiffrer qu'avec la clé privée correspondante. C'est pourquoi ce chiffrement s'appelle « asymétrique ».

ALICE CHIFFRE, BOB DECHIFFRE



Maintenant faisons intervenir les deux personnages Alice et Bob. Alice veut chiffrer un message qu'elle envoie à Bob, et être sûre que seul Bob pourra le déchiffrer. Pourquoi Alice et Bob ? Il est certain que ce chiffrement pourrait aussi bien être fait entre Yuan Zi et Huan Huan, les deux pandas géants du zoo de Beauval, mais avouez que ce serait plutôt bizarre. Et puis dans la littérature sur la cryptologie, c'est Alice qui chiffre et Bob qui déchiffre (et Ève qui espionne passivement) alors ne changeons rien aux noms de ces personnages devenus célèbres.

Deux cas peuvent être dès lors envisagés :

- Alice donne sa clé publique à tous ceux qui en ont besoin, Bob compris. Elle garde jalousement sa clé privée mathématiquement liée à sa clé publique, par exemple sur un token USB protégé par un code PIN. Alice chiffre avec sa clé privée le message que Bob va déchiffrer avec la clé publique d'Alice. Ça marche, mais avouez que ce serait idiot puisque tout le monde pouvant avoir la clé publique d'Alice, tout le monde pourrait déchiffrer son message, alors à quoi bon le chiffrer ?
- Alice demande à Bob de lui envoyer sa clé publique (celle de Bob, donc). Elle chiffre son message avec la clé publique de Bob, et Bob va déchiffrer le message d'Alice avec sa propre clé privée (celle de Bob donc, qu'il n'a communiquée à personne). Ça marche aussi et c'est la bonne solution.

Le cas 1 est absurde et on l'oublie (sauf dans le cas de la signature électronique, mais ce sera pour un prochain chapitre).

Dans le cas 2, Alice est sûre que seul Bob aura pu déchiffrer son message puisqu'un message chiffré avec la clé publique de Bob ne peut être déchiffré que par la clé privée correspondante que seul Bob possède. Mais Bob ne sait pas si c'est bien Alice qui a chiffré le message puisque la clé publique de Bob peut être connue par tous. La clé publique de Bob, largement partagée, n'est pas un secret, donc Bob ne peut être certain que c'est bien Alice qui a chiffré le message, il ne sait pas en fait qui l'a chiffré. Alice par contre est sûre que c'est Bob, et seulement lui, qui le déchiffrera.

Il manque alors quelque chose ? Oui mais c'est l'objet du chapitre sur le chiffrement symétrique. Restons-en là pour l'instant, et résumons : Quand Alice veut chiffrer un message à destination de Bob, elle demande à Bob de lui indiquer où est sa clé publique. Elle chiffre le message avec la clé publique de Bob, et Bob le déchiffre avec sa clé privée.

LA CLE PUBLIQUE EST-ELLE CELLE DE BOB OU CELLE D'ÈVE, L'ESPIONNE ?

Il y a tout de même deux problèmes à résoudre pour que ce modèle soit opérationnel :

- À partir de la clé publique de Bob, puisque tout le monde peut l'avoir, Ève, l'espionne, ne peut-elle pas reconstituer la clé privée de Bob, qui ne serait plus alors

un secret gardé farouchement par Bob ? Si oui tout s'écroule ! Avec la clé privée de Bob reconstituée, Ève serait dès lors capable de déchiffrer les messages pour Bob, chiffrés par Alice !

Ce problème est résolu par le fait qu'à partir d'une clé publique, il est impossible, ou en tout cas trop difficile avec les moyens de calculs actuels, de reconstituer une clé privée. Cela tient à l'utilisation d'algorithmes utilisés dans la constitution des deux clés, publique et privée, de problèmes mathématiques très difficiles à résoudre. Citons la factorisation d'un grand nombre (RSA) ou la résolution du logarithme discret (Diffie-Hellman). Ce sera expliqué dans un autre chapitre. Pour l'instant, ne compliquons pas, croyez-moi sur parole.

- Qu'est ce qui prouve à Alice que la clé publique, qui est prétendument celle de Bob, est bien celle de Bob ? Car supposons qu'Ève envoi à Alice une clé qu'elle fait passer pour être la clé publique de Bob. Si Alice l'utilise pour chiffrer, c'est Ève, et non Bob, qui pourra déchiffrer le message pour Bob, puisque Ève possède sa propre clé privée correspondant à la clé publique qu'elle a envoyée à Alice, en la faisant passer pour celle de Bob. De plus si on pousse le problème plus loin, Ève qui a aussi la clé publique de Bob, rechiffre le message d'Alice avec la clé publique de Bob. Celui-ci déchiffrera le message d'Alice avec sa propre clé privée, et ne saura donc pas que le message aura été lu au passage par Ève. C'est ce qu'on appelle l'attaque du « man in the middle » ou plutôt dans ce cas de la « femme au milieu ».

Ce problème est résolu par le certificat numérique. Quand Bob envoie sa clé publique à Alice, ou la met à la disposition de tous, dans un annuaire de clés publiques, ce ne sont pas les 1024 ou 2048 bits, par exemple, constituant sa clé publique qu'il fournit mais un certificat numérique la contenant. Ce certificat numérique est un fichier qui contient le nom du propriétaire de la clé publique, et divers autres renseignements sur son identité, les dates de validité de ce certificat et le tout est signé numériquement par une autorité de confiance. Cette signature numérique atteste que la clé publique est bien celle correspondant à la clé privée que son propriétaire est le seul à détenir. Si on change ne serait-ce qu'un seul espace, ou un seul bit dans le certificat, le logiciel qui va exploiter ce certificat, à la recherche de la clé publique, s'en apercevra tout de suite. Pourquoi ? Comment ? Ce sera expliqué dans un autre chapitre sur la signature numérique.

Reste un problème : Le chiffrement à clé publique qui manipule des clés longues et des algorithmes gourmands en puissance de calcul, est très lent, trop lent pour traiter de gros fichiers, beaucoup plus lent que le chiffrement symétrique. C'est donc le chiffrement symétrique qui va être utilisé pour chiffrer/déchiffrer. C'est traité dans un autre chapitre.

LE CONSEIL CRYPTO :

Vous pouvez, je dirais même vous devez, visualiser le certificat numérique d'un site web sur lequel vous allez, quand il vous le demande, déposer une information sensible, comme par exemple les renseignements sur votre carte de paiement, ou sur vos remboursements de santé.

Bien entendu quand un site web vous demande ce genre d'informations, ou d'autres données à caractère personnel, ne les déposez que si ce site vous semble être sécurisé, c'est-à-dire n'est pas en « http:// » mais en « https:// ». Un site web, dont l'adresse commence par https, vous garantit, en principe qu'il est bien le site que vous pensez

qu'il est (authentification du site par rapport à votre navigateur) et que toute transaction entre votre navigateur et ce site web sera intègre et confidentielle, parce qu'elle sera chiffrée. Votre navigateur connait un certain nombre de clés publiques. S'il ne connait pas celle du site que vous ouvrez pour la première fois, il vous envoie un message d'alerte et vous demande si vous considérez ce site comme sécurisé. Si vous acceptez, il ajoute le certificat numérique de ce site à son magasin de certificats.

Par exemple si vous allez sur le site https://www.google.fr avec le navigateur Mozilla Firefox (pas de publicité ici, logiciel libre, mais les autres navigateurs ont aussi ce cadenas) vous remarquez à droite de l'adresse du serveur web, un cadenas fermé. Cliquez dessus, puis sur la flèche à droite, puis sur « plus d'informations », puis sur l'icône « sécurité », et enfin sur « Afficher le certificat » et vous saurez tout sur son contenu, et jusqu'à la clé publique de ce site. L'important pour vous est « qui a émis/signé ce certificat ? ». Soit vous faites confiance en cette entité et vous poursuivez la transaction, soit vous ne lui faites pas confiance et vous arrêtez la transaction, soit vous ignorez le contenu de ce certificat et vous avez tort.

LES ECHANGES DES CLES DE CHIFFREMENT

COMBINER CHIFFREMENT SYMETRIQUE ET CHIFFREMENT A CLE PUBLIQUE

Nous avons écrit dans les chapitres précédents, qu'un chiffrement met en jeu des clés et un algorithme. On chiffre avec une clé à l'aide de l'algorithme, on déchiffre avec une clé et le même algorithme. Si la clé de chiffrement et celle de déchiffrement sont les mêmes, le chiffrement est dit « **symétrique** » et la clé est dite « **secrète** ». Si on chiffre avec une clé et on déchiffre avec une clé différente, le chiffrement est dit « **asymétrique** » ou encore « **à clé publique** » quand l'une des clés n'est pas un secret. Dans un chiffrement à clé publique, on chiffre avec **la clé publique du destinataire**, qui déchiffre avec une clé dite « **privée** » que seul lui possède.

Le chiffrement symétrique (exemple algorithme AES), basé sur des calculs simples, en particulier sur le « ou exclusif », est très rapide. Le chiffrement asymétrique (exemple RSA), basé sur des calculs lourds est très lent quand une grande quantité de données, ou des données très volumineuses, sont à chiffrer. Pour des questions de performance, c'est donc le chiffrement symétrique qui sera choisi pour chiffrer et déchiffrer les données numériques.

Le problème du chiffrement symétrique est que la clé secrète doit être échangée entre celui qui chiffre et celui qui déchiffre, mais cette clé doit rester secrète. Le problème de l'échange de la clé secrète est résolu par l'utilisation du chiffrement asymétrique. Ce chiffrement met en jeux deux clés, une clé privée que son propriétaire garde très confidentielle et une clé publique, mathématiquement liée à la clé privée, qui n'est pas confidentielle et qui donc peut, sans problème, être partagée avec tout le monde. On utilise la clé publique de son correspondant pour chiffrer, et son correspondant utilise sa clé privée, mathématiquement liée à sa clé publique, pour déchiffrer.

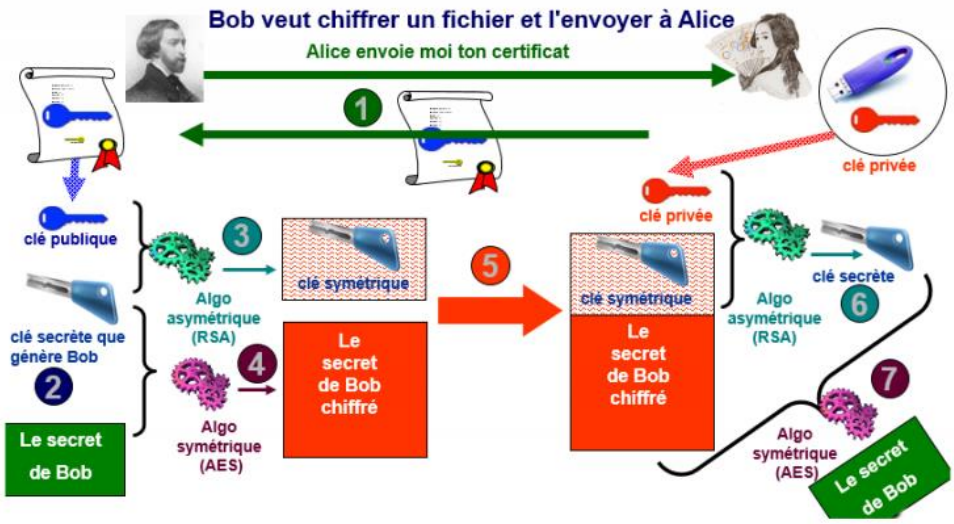
LES PHASES DES ECHANGES DES CLES

Introduisons le couple mythique de la cryptologie, Alice et Bob.

Bob veut échanger un message avec Alice en souhaitant que la confidentialité de ce message soit garantie. Autrement dit, seule Alice doit pouvoir le lire en clair. Bob va combiner le chiffrement symétrique et le chiffrement asymétrique. Les échanges de clés vont passer par différentes phases qui peuvent paraitre un peu complexes à première vue mais en fait, c'est très simple.

La complexité existe mais se trouve au niveau des algorithmes de chiffrement. Il n'est pas indispensable d'entrer dans le secret des algorithmes qui demandent des connaissances poussées en mathématiques pour comprendre les échanges de clés. Ici,

nous n'irons pas du tout dans les détails mathématiques, nous analyserons étapes par étapes comment et avec quoi Bob chiffre son message et Alice le déchiffre :



1- Bob, qui va émettre un message chiffré, demande d'abord à Alice **son certificat numérique** qui contient la **clé publique d'Alice** dont la propriété est attestée par l'autorité de confiance qui a signé numériquement ce certificat. Cette clé n'est pas confidentielle. Bob aurait pu aussi obtenir directement le certificat d'Alice dans un annuaire de certificats, généralement au format LDAP, ou déjà l'avoir obtenu si ce n'est pas la première fois qu'il chiffre des messages pour Alice. Si Bob a confiance en l'entité qui a signé le certificat d'Alice, si le certificat n'a pas été révoqué et s'il est dans ses dates de validité, Bob extrait, du certificat, la clé publique d'Alice. Alice possède **sa clé privée**, mathématiquement liée à sa clé publique pour être utilisée dans un **algorithme asymétrique**. Cette clé privée doit être gardée confidentielle par Alice.

2- Bob génère une clé secrète, généralement aujourd'hui de 128, 196 ou 256 bits pour être utilisée par l'algorithme symétrique AES. Cette clé sera valable pour la durée de la session. Elle est générée à partir d'une clé racine que Bob possède. Cette clé secrète de session doit évidemment être gardée confidentielle au cours de l'échange qui va suivre. Par sécurité, il est préférable que la clé racine, qui sert à générer les clés secrètes de sessions, se trouve sur un support amovible protégé par un code que seul Bob connait.

3- Bob chiffre sa clé secrète, avec la clé publique d'Alice, et **un algorithme de chiffrement asymétrique** (comme le **RSA**).

4- Bob chiffre son message avec sa clé secrète qu'il vient de générer et l'algorithme de chiffrement symétrique (comme l'AES). Le message de Bob ainsi chiffré est prêt à être envoyé. Mais Alice n'a pas encore la clé secrète générée par Bob.

5- Bob envoie à Alice

- Son message chiffré avec sa clé secrète
- Sa clé secrète chiffrée avec la clé publique d'Alice.

6- Alice a reçu ces deux informations chiffrées

- Le message chiffré avec la clé secrète de Bob
- La clé secrète de Bob chiffrée avec la clé publique d'Alice.

7- Alice a tout pour obtenir le message de Bob en clair :

Elle utilise sa clé privée pour déchiffrer par un algorithme asymétrique, la clé secrète de Bob. Rappelons que la clé secrète de Bob a été chiffrée avec la clé publique d'Alice. Rappelons aussi que ce qui est chiffré avec une **clé publique**, ne peut être déchiffré qu'avec **la clé privée** correspondante, et un algorithme asymétrique. Alice obtient donc en clair la clé secrète de Bob, qui a été transmise de manière sûre sur le réseau.

Elle déchiffre le message de Bob avec la clé secrète de Bob qu'elle vient d'obtenir en clair.

Compliqué ? Pas du tout, mais peut-être faut-il relire plus d'une fois les étapes qui précèdent pour s'en imprégner et avoir une vue complète, en prenant du recul. En fait le génie de cette méthode tient dans sa simplicité.

Et c'est ainsi, en combinant chiffrement symétrique (pour chiffrer/déchiffrer le message) et asymétrique (pour transmettre la clé) que fonctionne par exemple le PGP (Pretty Good Privacy) mis en avant par Philip Zimmermann au début des années 90, et aujourd'hui utilisé par la plupart des échanges chiffrés et le chiffrement sur disque.

L'APPORT DE LA PHYSIQUE QUANTIQUE

La physique quantique, notez que nous ne parlons pas de calculateurs quantiques, mais de contribution de la physique quantique au chiffrement symétrique, apporte de bonnes solutions à plusieurs problèmes posés par la génération et par l'échange des clés. Nous n'évoquons ici que les problèmes et leur résolution. Le comment sera traité dans un autre chapitre.

La clé secrète générée par Bob doit être aléatoire, pour ne pas qu'il soit possible de la déduire. Mais un ordinateur classique ne peut générer que des clés pseudo aléatoires. Dans de grandes séries de clés générées, celles-ci ont tendance à se répéter. La physique quantique permet de créer des clés purement aléatoires.

Quand existeront des calculateurs quantiques opérationnels, la factorisation d'un grand nombre se fera très rapidement (en particulier par l'algorithme de Shor), et donc le chiffrement à clés publiques sera compromis puisque, à partir d'une clé publique, on pourra calculer la clé privée correspondante en un temps acceptable.

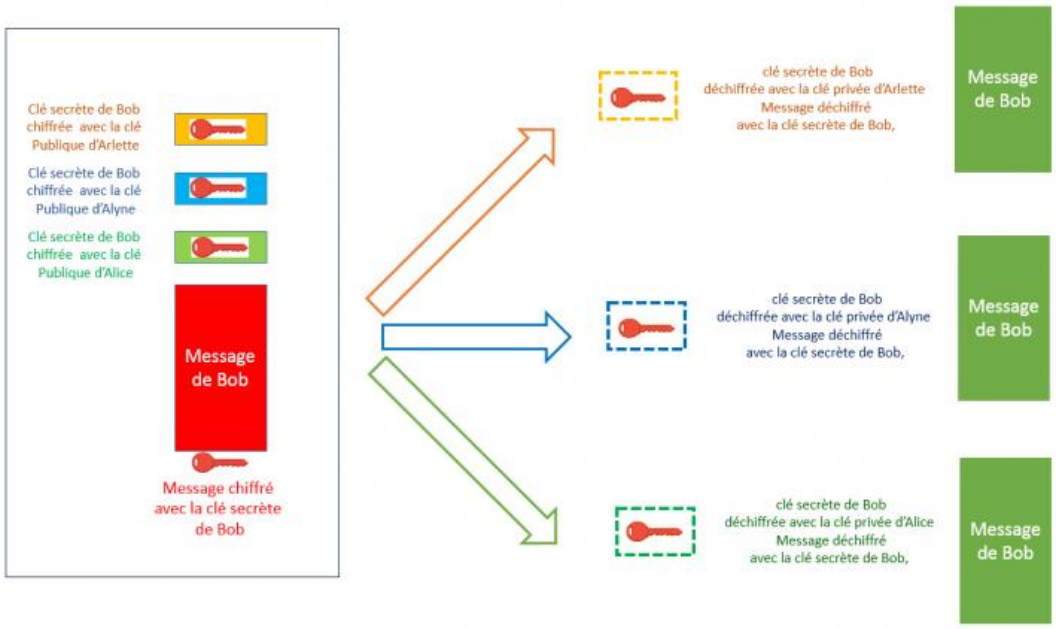
Mais la physique quantique permet également de transmettre de manière sûre une clé secrète. Si quelqu'un tente d'observer la clé, le principe de décohérence quantique joue et on s'aperçoit qu'un espion (ou une espionne comme Eve, l'Eyedropper) est dans la boucle. Ainsi le chiffrement asymétrique n'aura pas à être utilisé pour transmettre la clé secrète. En utilisant le principe d'intrication quantique, la clé secrète pourrait même ne pas avoir besoin d'être transmise. Elle pourra apparaitre simultanément à deux endroits, même très éloignés.

EN CONCLUSION

En combinant le chiffrement symétrique, type AES, pour chiffrer / déchiffrer les messages et le chiffrement asymétrique, type RSA, pour échanger la clé, on a combiné le meilleur des deux mondes, et c'est ainsi que se fait le chiffrement, en attendant que la physique quantique ne vienne bouleverser cet ordre établi.

LE CONSEIL CRYPTO :

Bob veut envoyer, par mail un message à Alice, Alyne et Arlette, en préservant sa confidentialité. Il pourrait certes envoyer trois fois son message en procédant comme on a dit plus haut, mais il ne veut l'envoyer qu'une seule fois aux trois destinatrices, comment peut-il procéder ?



Bob demande aux trois destinatrices de son message leurs certificats numériques. Après vérification des autorités qui ont signé les certificats, il en extrait les trois clés publiques (celle d'Alice, d'Alyne et d'Arlette). Il génère sa clé secrète. Il la chiffre avec la clé publique d'Alice, ça lui donne un premier paquet, puis il chiffre sa clé secrète avec la clé publique d'Alyne, deuxième paquet et enfin il obtient un troisième paquet en chiffrant sa clé secrète avec la clé publique d'Arlette.

Il chiffre son message avec sa clé secrète.

Il joint à son message chiffré les trois paquets, et il envoie le tout aux trois destinatrices.

Chacune d'elles déchiffre le paquet qui lui est destiné avec sa clé privée, et un algorithme asymétrique, et obtient donc la clé secrète générée par Bob. Elle utilise cette clé secrète, et un algorithme symétrique, pour déchiffrer le message de Bob.

Alice, Alyne et Arlette ont donc obtenu le message en clair que bob leur a transmis chiffré.

LE CALCUL D'EMPREINTE

Nous allons explorer maintenant une fonction qui n'est pas à proprement parler une fonction de chiffrement, et qui est à la base de la preuve d'**intégrité** d'un message ou d'un fichier et de la **signature électronique** : elle est également à la base de la **blockchain**. C'est sur une blockchain que reposent les bitcoins et d'autres crypto monnaies.

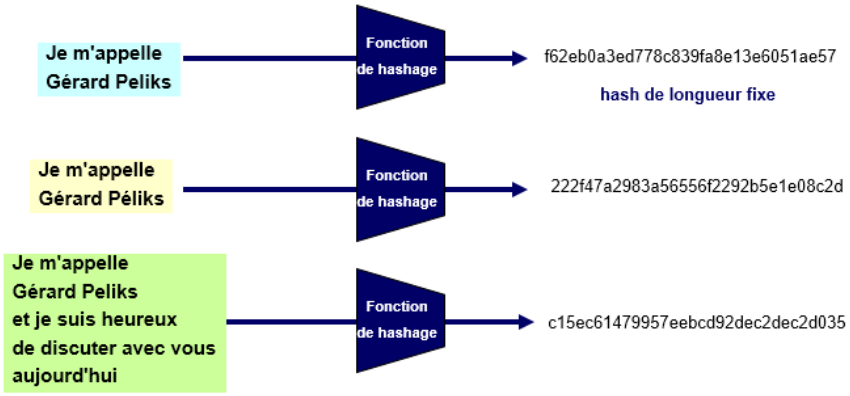
L'idée est, à partir d'algorithmes (programmes mathématiques), de créer en partant d'un texte de longueur quelconque, une chaîne de caractères de longueur fixe qui caractérise le texte. Jusque-là, ça va ?

Prenons un exemple :

Si le texte « *Adhérez à l'association Forum ATENA* » donne, après passage par la fonction à sens unique, la chaîne de caractères de longueur fixe « *f62eb0a3ed778c839fa8e13e6051ae57* », si l'on change ne serait-ce qu'un élément dans le texte initial, par exemple « *Adhérez à l'association Forum ATHENA* », le passage de ce dernier texte, par la même fonction à sens unique, donnera une chaîne de caractères toujours de la même longueur, mais complètement différente, par exemple « *222f47a2983a56556f2292b5e1e08c2d* ». La chaîne de caractères est donc caractéristique du contenu du texte.

Quel que soit le nombre de caractères du texte, le passage par la fonction à sens unique donnera une chaîne de caractères toujours de la même longueur. Par exemple le texte « *Adhérez à l'association Forum ATENA, car vous pourrez assister à la prochaine soirée networking ce mercredi à partir de 19h00, au Sénat* » après passage par la fonction à sens unique donnera la chaîne de caractères, toujours de longueur fixe : « *c15ec61479957eebcd92dec2dec2d035* ».

Tout cela est finalement très simple.



Explicitons quelques mots de vocabulaire maintenant :

La fonction à sens unique s'appelle « **une fonction de hachage** ». Citons parmi les algorithmes les plus utilisés qui implémentent les fonctions de hachage, le MD5, le SHA1, le SHA2 (SHA256 ou SHA512), le SHA3 très moderne, mais ne compliquons pas.

La **chaîne de caractères fixe** s'appelle un « **hash** » en langage anglo saxon, une « **empreinte** » en français courant et un « **condensat** » en bon français, mais on laisse tomber ce dernier mot qui est moins connu. Nous parlerons donc dans ce qui suit d'**empreinte**.

Donc un texte de longueur quelconque donne, après passage par une fonction de hachage, une empreinte de longueur fixe qui caractérise le texte.

Pourquoi la fonction de hachage est-elle dite « *fonction à sens unique* » ? Parce qu'à partir de la chaîne de caractères de longueur fixe, il n'est pas possible, ou du moins il est très difficile, de retrouver le texte de longueur quelconque, et d'ailleurs ce n'est pas le but recherché. Une fonction de hachage n'est pas une fonction de chiffrement.

Mais alors quel est son but ???

J'y viens. L'un des buts est de prouver, par exemple, **l'intégrité** d'un texte.

J'ai un texte, je calcule son empreinte par une fonction de hachage, mettons le SHA256. Je vous envoie mon texte et son empreinte.

Vous avez reçu mon texte et l'empreinte qui l'accompagne. Vous recalculez l'empreinte du texte reçu par la même fonction de hachage. Si l'empreinte recalculée est la même que l'empreinte reçue, il est prouvé que le texte que je vous ai envoyé, et que vous avez reçu avec son empreinte, n'a pas été modifié. Nous avons donc établi l'intégrité de ce texte.

Vous êtes d'accord ?

Et bien vous avez tort, nous n'avons rien prouvé du tout. Si quelqu'un intercepte mon message, le modifie, recalcule l'empreinte et vous envoie le tout, le message que vous recevrez sera différent du message que je vous aurai envoyé, bien que l'empreinte que vous aurez reçue avec le texte est la même que l'empreinte que vous aurez recalculée à partir du texte reçu. Donc l'intégrité du message n'est pas du tout prouvée. Ce qui est prouvé, c'est juste que le texte n'a pas été modifié depuis sa dernière modification avec recalcul de l'empreinte. Ce calcul a pu être fait par un pirate.

Mais alors, à quoi ça… ???

Attendez, pour vous expliquer le grand jeu de la **signature électronique**, qui sera la réponse, il faut avoir compris le mécanisme du **chiffrement à clé publique** (dit encore asymétrique), expliqué dans un autre chapitre.

LE CONSEIL CRYPTO :

Utilisez des mots de passe solides d'au moins 8 caractères, avec lettres minuscules (certaines accentuées), majuscules, chiffres et caractères spéciaux. Pourquoi ? Bien sûr, les mots de passe ne sont pas stockés en clair sur votre disque, pour des raisons évidentes de sécurité, mais leurs empreintes le sont. Chaque fois que vous entrez votre mot de passe, son empreinte est calculée et comparée à l'empreinte stockée. L'empreinte stockée n'est pas un secret, elle sert à la comparaison entre une empreinte recalculée et elle-même. Si les deux empreintes correspondent, vous avez accès à votre compte ou à votre ressource que votre mot de passe protège. Par force brute, en essayant toutes les combinaisons, si votre mot de passe est de six lettres minuscules, non accentuées, en essayant au plus 26x26x26x26x26x26 combinaisons possibles de mots de passe et en comparant leurs empreintes calculées aux empreintes stockées, avec un PC du commerce, on trouve la combinaison gagnante en 3 secondes. Avec un mot de passe de 8 caractères et conçu comme je vous l'ai dit au début du conseil, il faudra 3 mois. Si vous changez votre mot de passe de temps en temps, c'est plus sécurisé. Mais comment retenir de multiples mots de passe, qui sont en plus de temps en temps à changer ?

Mettez-les dans un « coffre-fort électronique » comme Keepass (logiciel libre et gratuit).

LA SIGNATURE NUMERIQUE

CALCUL D'EMPREINTE ET CHIFFREMENT A CLE PUBLIQUE

Quand un document papier est signé, il est établi que celui qui l'a signé l'a au moins lu, et que le document n'a pas été modifié, après sa signature.

La signature numérique, dite encore signature électronique, d'un document a pour fonction de garantir l'intégrité de ce document (il n'a pas été altéré depuis sa signature) et d'authentifier son signataire. Un document signé peut être en clair, mais il peut être aussi chiffré, ce qui garantit en plus sa confidentialité. Signature numérique et chiffrement d'un document sont deux fonctions indépendantes.

Pour comprendre le mécanisme de la signature numérique, il est indispensable de comprendre ce qu'est le calcul d'empreinte et le chiffrement à clé publique, dit encore chiffrement asymétrique.

En quelques mots, le calcul d'empreinte fait correspondre à un document, quelle que soit sa longueur, une chaine de caractères de longueur fixe. Cette chaine de caractères s'appelle l'empreinte ou, en bon français le condensat ou encore, en anglais, le hash. Elle caractérise ce document.

Le chiffrement à clé publique utilise deux clés, une clé privée et une clé publique. Ces deux clés sont mathématiquement liées de telle sorte que quand on chiffre avec l'une, on déchiffre avec l'autre. L'utilisateur garde sa clé privée et diffuse sa clé publique dans un certificat signé par une autorité de confiance. Connaissant une clé publique, il est très difficile de retrouver la clé privée correspondante.

LE MECANISME DE LA SIGNATURE NUMERIQUE POUR GARANTIR AUTHENTICITE ET INTEGRITE

L'utilisateur qui désire prouver qu'il est l'auteur d'un document et que ce document n'a pas été modifié le signe, par les actions suivantes :

- Il calcule l'empreinte de son document par une fonction de hachage, telle le SHA128 ou le SHA256.
- Il chiffre cette empreinte avec sa clé privée, que lui seul possède, par un algorithme de chiffrement à clé publique tel que le RSA ou les courbes elliptiques.
- Il joint cette empreinte chiffrée à son document. Le document est alors scellé.

Ceci permettra d'établir l'authenticité du signataire, et en même temps l'intégrité du document.

Celui qui lit le document vérifie qui l'a signé. Il peut vérifier aussi, dans le certificat contenant la clé publique du signataire, qui est l'autorité de confiance qui a signé le certificat. Toute la confiance en cette signature numérique repose sur la confiance qu'il a en cette autorité. S'il fait confiance en cette autorité, si le certificat se trouve bien dans les dates de validité qu'il contient, et si le certificat n'a pas été révoqué, il sait que la clé publique qu'il extrait du certificat correspond bien à la clé privée, mathématiquement liée à la clé publique du signataire dont l'identité se trouve aussi inscrite dans le certificat.

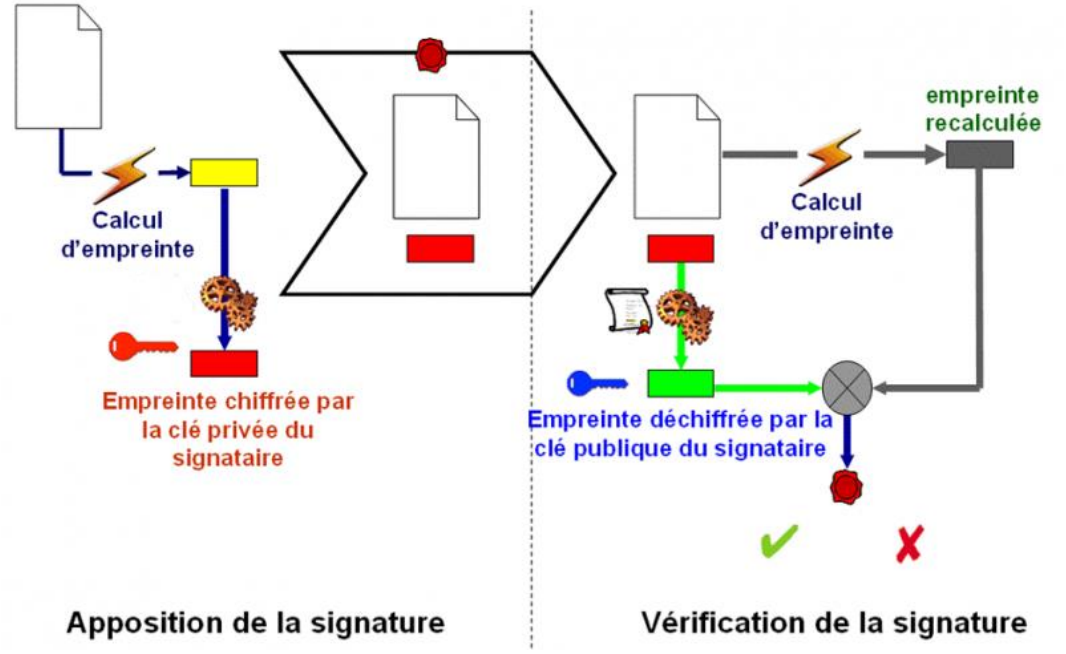
Il déchiffre l'empreinte de ce document à l'aide de la clé publique du signataire, par le même algorithme de chiffrement asymétrique que le signataire (qui a utilisé sa clé privée pour chiffrer l'empreinte). Cet algorithme est indiqué dans le certificat. Il obtient une empreinte déchiffrée. Il calcule ensuite l'empreinte du document, avec la même fonction de hachage que le signataire, cette fonction étant aussi indiquée dans le certificat. Il obtient l'empreinte recalculée.

Bien entendu ce n'est pas l'utilisateur qui effectue le déchiffrement de l'empreinte liée au document et qui recalcule l'empreinte du document mais l'application de signature numérique qu'il utilise. Cette application peut être la messagerie ou encore le navigateur web dans le cas d'une connexion sécurisée par SSL.

Si l'empreinte recalculée est la même que l'empreinte déchiffrée, seul le signataire dont les coordonnées sont inscrites dans le certificat d'où sa clé publique a été extraite, et attesté par l'autorité de confiance qui a signé le certificat, a pu avoir chiffré l'empreinte du document. Car seul le signataire possède sa clé privée. Ceci atteste de l'authenticité du signataire, garantie par l'autorité de confiance.

L'intégrité du document est aussi établie, car si le document avait été modifié depuis sa signature, l'empreinte recalculée n'aurait pas été la même que l'empreinte déchiffrée.

Ainsi sont établies l'authenticité de l'auteur et l'intégrité de son document, depuis sa signature. L'autorité de confiance a signé le certificat numérique par le même mécanisme que nous avons décrit plus haut, en utilisant donc aussi sa clé privée. L'empreinte du certificat, calculée et chiffrée par l'autorité de confiance, incluse dans le certificat, atteste de l'intégrité de ce certificat.



L'AUTORITE DE SIGNATURE DU CERTIFICAT, POUR ETABLIR LA CONFIANCE

Tous les certificats numériques n'ont pas la même valeur, suivant comment ils ont été obtenus, et par quelle autorité ils ont été signés. Il est parfois exigé que l'autorité de confiance qui a signé le certificat réside dans le même pays que celui qui vérifie la signature du document.

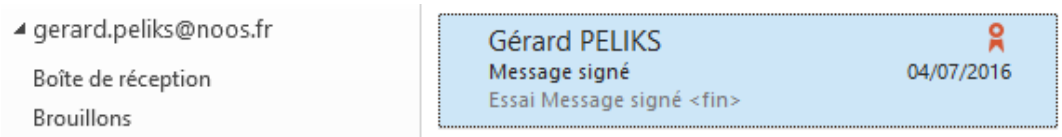
La signature numérique d'un document est indépendante du calculateur utilisé pour produire ou envoyer le document. Un navigateur web possède une liste d'autorités de confiance. Si le certificat d'un site web sécurisé (en https://) a été signé par une autorité de confiance connue par le navigateur, celui-ci ouvrira le site sans vous poser de questions. Dans le cas contraire il vous avertit qu'il ne connait pas l'autorité qui a signé le certificat et vous demande de la valider. Si vous acceptez, cette autorité est ajoutée à celles déjà connues par votre navigateur. Si le certificat d'un site web sécurisé est au-delà de sa date limite de validité, votre navigateur vous le signale.

LE CONSEIL CRYPTO :

Comment sait-on qu'un message, par exemple, reçu par messagerie, est signé ?

Tout dépend bien sûr du client de messagerie utilisé.

Dans la copie d'écran ci-dessous, un sceau indique que j'ai signé ce message, que je me suis envoyé. J'ai signé ici le message avec le client de messagerie Thunderbird et le module complémentaire Enigmail.



Avant d'exploiter la signature du message, il a fallu que le signataire du message, vous ait fait parvenir son certificat, pour que votre messagerie puisse déclencher les mécanismes de signature numérique décrits plus haut. Dans ce message signé, figure en attachement, un fichier « .asc ». Double cliquer sur ce fichier permet à Thunderbird d'importer le certificat contenant la clé publique de celui qui a signé le message.

Pour signer un message et l'envoyer, par Thunderbird avec le module complémentaire Enigmail, rien de plus simple. Dans votre client de messagerie, il y a une icône représentant un crayon. Cliquez dessus et vous avez signé votre message.

Il y a à côté du crayon une autre icône représentant un cadenas. Cliquez dessus et vous avez chiffré votre message. Tous les mécanismes de signature et de chiffrement sont pour vous transparents.

Nous abordons maintenant deux autres facettes de la cryptologie : La cryptologie quantique et le chiffrement homomorphe.

LA CRYPTOLOGIE QUANTIQUE, AUJOURD'HUI

DEUX PROBLEMES POSES AU CHIFFREMENT CLASSIQUE

Le chiffrement dit « classique » se fait en combinant le chiffrement symétrique pour chiffrer, et le chiffrement asymétrique pour échanger, en assurant sa confidentialité, la clé qui va servir à chiffrer, puis à déchiffrer, le message.

Alice veut correspondre avec Bob en garantissant la confidentialité de ses messages. Elle génère une **clé secrète**, **aléatoire** qui va servir pour la durée de la session. Avec cette clé secrète, Alice chiffre ses messages avec un **algorithme symétrique** comme l'AES. Bob, destinataire des messages d'Alice, doit les déchiffrer avec la même clé et le même algorithme symétrique. Pour transmettre à Bob cette clé secrète, et qui doit le rester, en toute confidentialité sur un réseau qui peut ne pas être sûr, Alice a besoin de la **clé publique** de Bob qu'il lui fournit dans son certificat numérique.

La clé publique de Bob est mathématiquement liée à une **clé privée** que Bob possède et ne donne à personne. Si on chiffre avec l'une des clés, publique ou privée, et un algorithme de chiffrement asymétrique comme le RSA, on déchiffre avec l'autre clé et le même algorithme. Bien entendu, à partir de la clé publique de Bob, qu'il donne à qui la lui demande dans son certificat numérique**, il ne doit pas être possible** à quiconque de reconstituer la clé privée de Bob.

Avec la clé publique de Bob, et un algorithme asymétrique, Alice chiffre la clé secrète aléatoire qu'elle a générée, et que Bob pourra déchiffrer avec sa clé privée, avec le même algorithme asymétrique utilisé par Alice. Bob reçoit le message chiffré avec la clé secrète d'Alice, et la clé secrète d'Alice qu'elle a chiffrée avec la clé publique de Bob. Il déchiffre cette clé secrète avec sa clé privée, et s'en sert pour déchiffrer le message d'Alice.

Mais il y a deux problèmes :

- Un ordinateur, machine déterministe, ne peut créer par logiciel **une clé purement aléatoire**. En effet, sans signal extérieur aléatoire, un ordinateur ne peut créer que des **clés pseudo aléatoires**. Dans les grandes séries de clés générées, celles-ci ont tendance à se répéter, ce qui peut les rendre prédictibles et donc affaiblir la robustesse du chiffrement symétrique.
- À partir d'une clé publique qui intervient dans un chiffrement asymétrique, il est très coûteux en temps, avec un calculateur classique, de reconstituer la clé privée, mathématiquement liée à la clé publique. Très coûteux en temps mais pas impossible, l'algorithme de Shor peut aider. La difficulté de reconstituer une clé privée, qui est un secret qui engage l'identité de son propriétaire, à partir d'une clé publique qui, comme son nom l'indique, est publique, repose sur un problème mathématique difficile à résoudre, comme la factorisation d'un très grand nombre, produit de deux nombres premiers, dans le cas de l'algorithme asymétrique RSA.

La **physique quantique** apporte une solution très sûre au problème de la difficulté pour un ordinateur de créer des clés purement aléatoires utilisées dans le chiffrement symétrique.

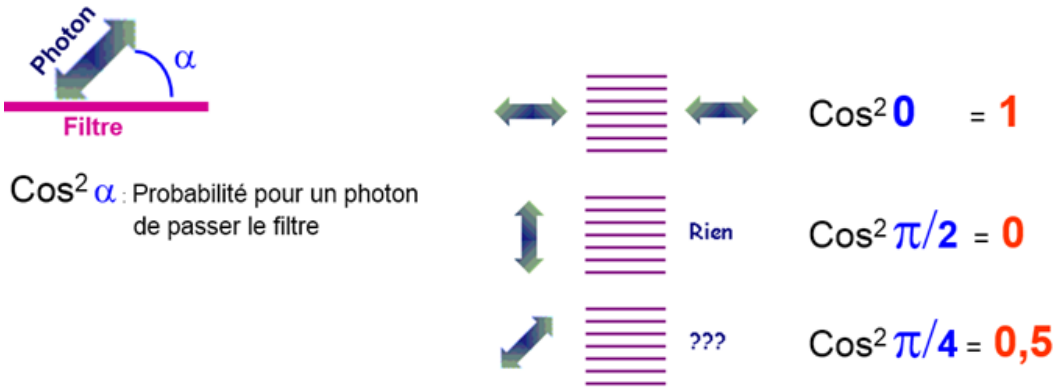
Un calculateur quantique, quand il sera pleinement opérationnel, permettra de résoudre dans un temps suffisamment court la factorisation d'un grand nombre utilisé dans un chiffrement asymétrique. Donc il permettra de retrouver les deux nombres premiers dont le produit constitue ce grand nombre, et, de là, la clé privée correspondant à cette clé publique. Le chiffrement asymétrique comme le RSA, solution pour transmettre la clé secrète de chiffrement symétrique, ne pourra alors plus être conseillé. Pareil pour le chiffrement asymétrique basé sur le « logarithme discret ». Mais la physique quantique apporte, dès aujourd'hui aussi, **une solution très sûre à l'échange des clés de chiffrement symétriques**.

ORIENTER LA POLARISATION D'UN PHOTON, PARTICULE ELEMENTAIRE, ONDE ET GRAIN D'ENERGIE

La physique quantique s'applique aux particules élémentaires de l'infiniment petit, comme les photons, et explique que la lumière peut être vue comme constituée de photons, qui sont des ondes ou des grains d'énergie, ou les deux à la fois. Nous ne développerons pas de théorie sur la physique quantique, hors propos dans ce chapitre. Retenons seulement qu'il est possible d'orienter la polarisation de chaque photon, objet quantique, et de les envoyer, un par un, sur un miroir incliné semi-réfléchissant.

Le principe de décohérence quantique explique qu'on ne peut observer directement la polarisation d'un photon sans que cette observation ne change son orientation. Mais on peut quand même déterminer si les photons ont traversé le miroir semi-réfléchissant ou au contraire ont été réfléchis.

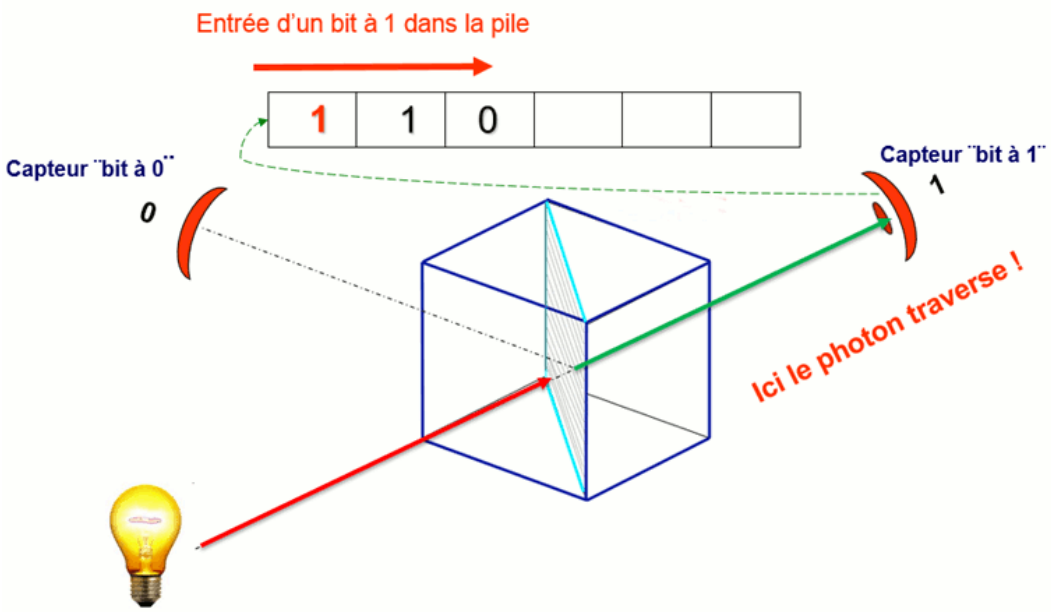
Si le photon dont la polarité a été orientée, rencontre le miroir semi-réfléchissant sous une incidence α par rapport aux stries du miroir, la probabilité qu'il passe à travers, sans être réfléchi, est exactement égale à $\cos^2 \alpha$.



- Si le photon et le miroir ont la même orientation, $\cos^2 0 = 1$ et la probabilité que le photon passe est de 100%. Donc le photon traverse le miroir.
- Si le photon et le miroir sont à angle droit (90 degré), $\cos^2 \pi/2 = 0$ et la probabilité que le photon passe est de 0%. Donc le photon est réfléchi par le miroir.
- Si le photon et le miroir sont à 45 degré, $\cos^2 \pi/4 = 1/2$ et le photon a exactement une chance sur deux de traverser le miroir, et une chance sur deux d'être réfléchi. Le résultat est donc mathématiquement imprédictible.

LA GENERATION D'UN NOMBRE PUREMENT ALEATOIRE RENDUE POSSIBLE PAR LA PHYSIQUE QUANTIQUE

Nous prenons un miroir semi réfléchissant incliné à 45 degré (π/4) par rapport à l'orientation de la polarisation de chaque photon qui le frappe. On envoie les photons un par un sur le miroir. Si le photon traverse, il parvient à un capteur, situé derrière le miroir, qui entre un bit à 1 dans une pile. Si le photon ne traverse pas, il est réfléchi et parvient à un autre capteur situé devant le miroir qui entre un bit à 0 dans la même pile.



Comme chaque photon a mathématiquement une chance sur deux de traverser le miroir, provoquant l'entrée d'un bit à 1, ou d'être réfléchi, provoquant l'entrée d'un bit à 0, photon après photon, un nombre purement aléatoire se constitue dans la pile. Ce nombre sera utilisé comme clé secrète qui interviendra dans le chiffrement symétrique.

Dans l'idéal, si la longueur de la clé est égale à la longueur du message qu'on veut chiffrer, et si cette clé n'est utilisée qu'une seule fois, on peut obtenir un chiffrement incassable (comme le chiffre de Vernam). Mais si on limite la longueur de la clé à 128, 192 ou 256 bits, on peut utiliser le chiffrement symétrique AES.

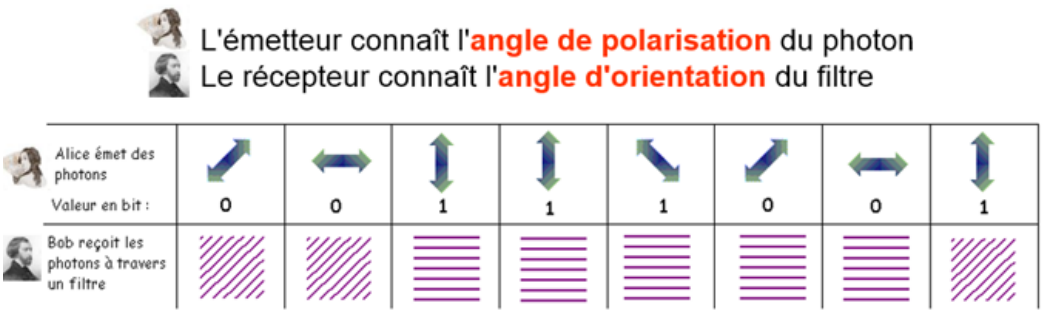
Ainsi est résolue avec élégance, grâce à la physique quantique, la génération de clés purement aléatoires, qui peuvent servir dans un chiffrement symétrique. Des générateurs de clés purement aléatoires existent dans le commerce, voir l'exemple pratique de l'université de Genève à la fin de ce chapitre.

Reste à trouver un moyen d'échanger cette clé entre Bob et Alice, puisque, quand un calculateur quantique sera opérationnel, le chiffrement asymétrique type RSA ne pourra plus assurer une sécurité suffisante dans le transfert de la clé symétrique.

LE TRANSFERT SUR DE LA CLE SECRETE, A TRAVERS UN RESEAU QUI PEUT ETRE PUBLIC

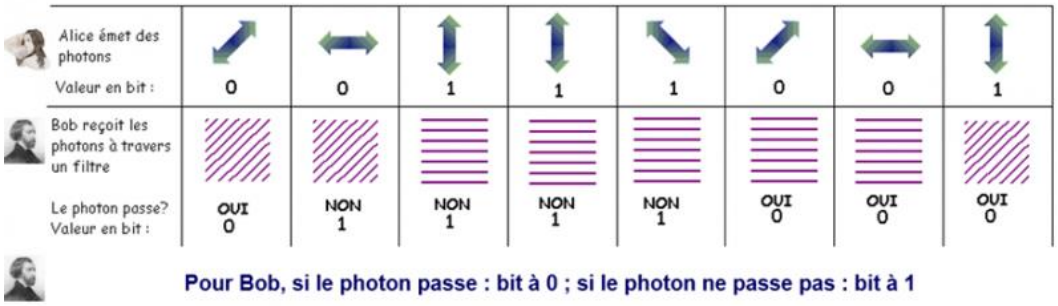
Le calculateur quantique rendra moins opérante la sécurité du chiffrement asymétrique pour acheminer la clé symétrique de chiffrement/déchiffrement, mais là encore, la physique quantique apporte, dès aujourd'hui, une solution.

Alice veut transmettre la clé secrète, qu'elle vient de générer, à Bob, à travers une fibre optique qui peut ne pas être protégée. Elle sait orienter la polarisation des photons et les envoyer un par un. Elle oriente chaque photon en « rectiligne » (0 ou 90 degrés), ou en « diagonal » (45 ou 135 degrés) par rapport à la verticale. Alice convient que, pour la clé qu'elle veut transmettre à Bob, pour un bit de la clé à 0, elle oriente la polarisation du photon à 45 **ou** à 90 degrés. Pour un bit à 1, elle l'oriente à 0 **ou** à 135 degrés.

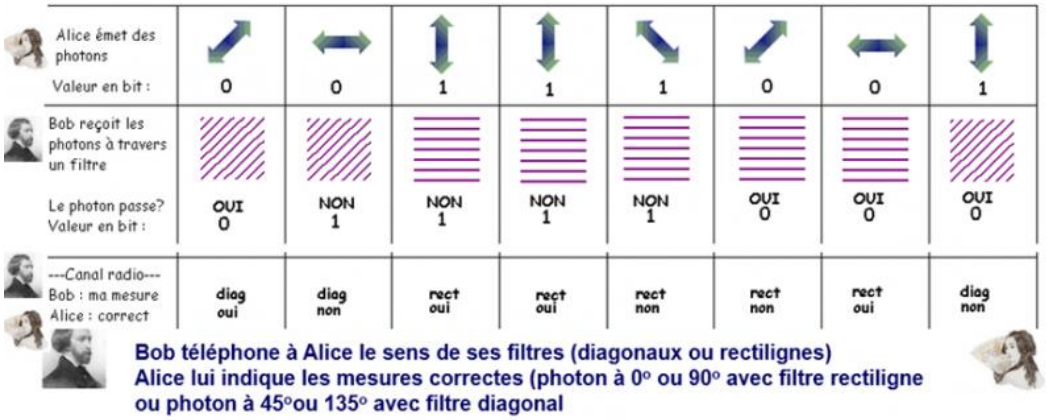


Bob place un miroir sur le trajet des photons. Il oriente son miroir en rectiligne (0 ou 90 degrés) ou en diagonal (45 ou 135 degrés) par rapport à la verticale, mais n'a aucune idée de l'orientation de la polarisation de chacun des photons qu'Alice lui envoie. Alice ne connait pas comment Bob oriente son miroir pour chaque photon reçu. Bob peut juste savoir si le photon a traversé ou pas son miroir.

Si le photon passe, il note un bit à 0. Si le photon ne passe pas, il note un bit à 1.



Par un autre canal, par exemple avec son smartphone, Bob téléphone à Alice pour lui dire s'il avait, pour chaque photon reçu, orienté son miroir en rectiligne (0 ou 90 degrés), ou en diagonal (45 ou 135 degrés), par rapport à la verticale.



Donc Bob dit à Alice, par téléphone : **« rectiligne** » ou « **diagonal** ».

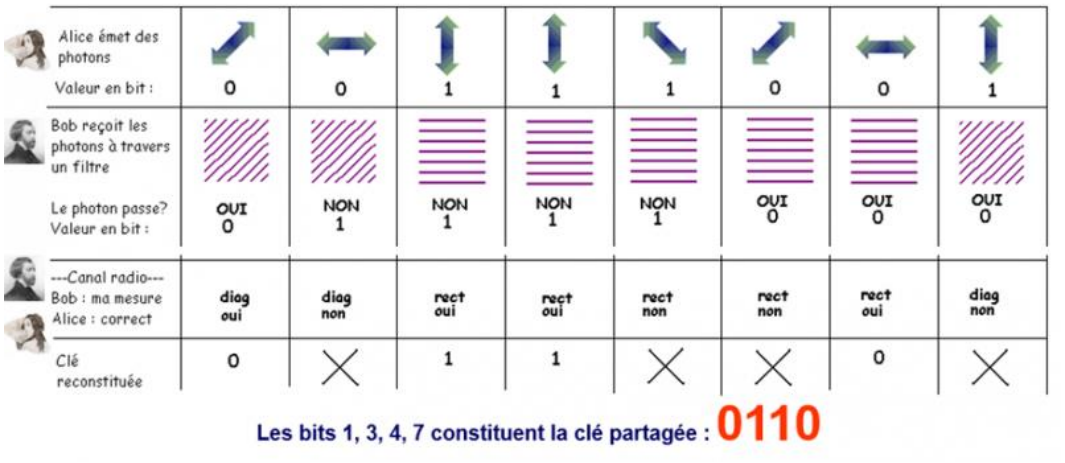
Si Bob a dit au téléphone « **rectiligne** » Alice répond à Bob « **oui** » si elle avait orienté son photon à 0 ou à 90 degrés, et « **non** » si elle avait orienté son photon à 45 ou à 135 degrés.

Si Bob a dit au téléphone « **diagonal** », Alice répond à Bob « **oui** » si elle avait orienté son photon à 45 ou 135 degrés, et « **non** » si elle avait orienté son photon à 0 ou 90 degrés.

Quand Alice a répondu « **oui** » à Bob, si la polarisation du photon faisait un **angle de 0 degré** avec l'orientation du miroir, il y avait 100% de chance que le photon ait traversé le miroir. **Bob considère que le bit de la clé est à 0**. Si la polarisation du photon faisait un angle de 90 degré avec l'orientation du miroir. Il y avait 100% de chance que le photon n'ait pas traversé le miroir. **Bob considère que le bit de la clé est à 1**.

Quand Alice a répondu « **non** » à Bob, il n'y avait que 50% de chance que le photon ait traversé le miroir car la polarisation du photon faisait un angle de 45 ou 135 degrés avec l'orientation du miroir.

Alice et Bob laissent tomber les bits issus des configurations où la probabilité que le photon traverse ou ne traverse pas le miroir était de 50% (ceux pour lesquels Alice a répondu « non ») et ne gardent que les valeurs déterministes (celles pour lesquelles Alice a répondu « oui ». Alice et Bob se partagent ainsi la clé qui va servir au chiffrement symétrique, comme l'AES (dans cet exemple : 0110).



POURQUOI CETTE METHODE DE TRANSFERT DE CLE EST-ELLE SURE ?

Si Ève, l'espionne passive (**E**yedropper), se place au milieu des échanges entre Bob et Alice, pour observer la polarisation d'un photon, l'orientation du photon émis par Alice est modifiée, principe de décohérence quantique. Ève ne connaissant pas l'orientation initiale du photon choisie par Alice ne saura pas avec quelle orientation elle devra le renvoyer à Bob après avoir observé ce photon. La probabilité que Bob et Alice s'aperçoivent que la clé a été observée est ainsi très forte.

Pour être persuadés que la méthode choisie a donné les bons résultats, Bob et Alice choisissent de sacrifier quelques bits de la clé. Bob annonce par exemple à Alice, par téléphone : « ***Le premier bit de notre clé était 0*** ». Alice approuve et ils enlèvent de la clé ce premier bit révélé. La clé qu'ils ont ainsi échangée est, dans cet exemple, 110.

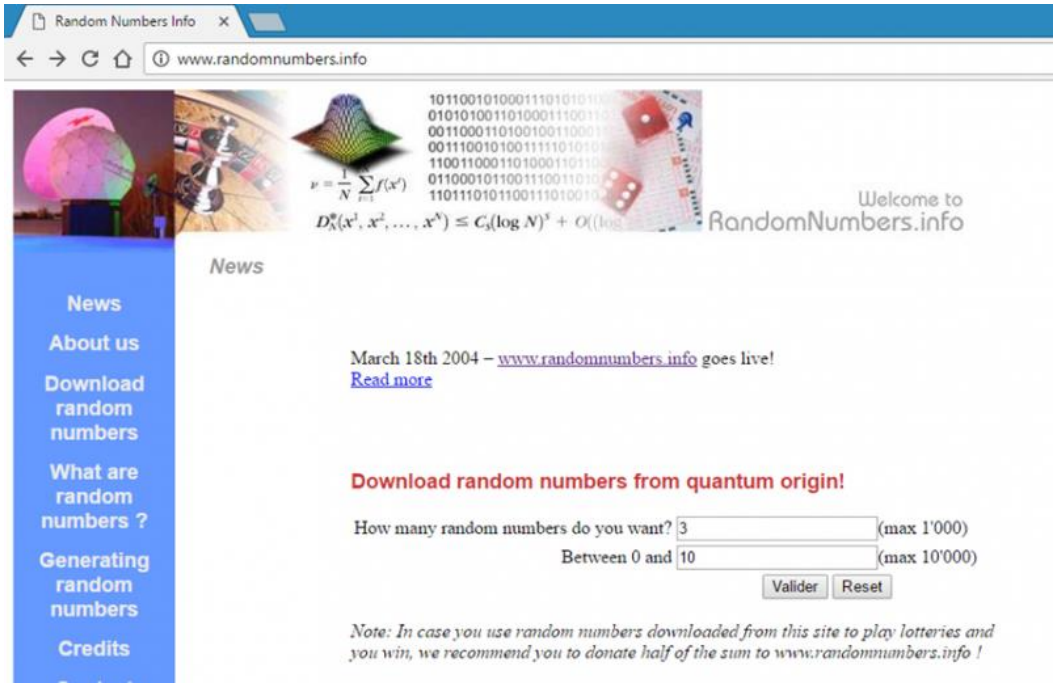
Ainsi, avant même que le calculateur quantique soit opérationnel et condamne la sécurité du transfert de clés secrètes chiffrées par chiffrement asymétrique, la physique quantique apporte une réponse à ce transfert, et aussi une réponse à la production de clés symétriques vraiment aléatoires.

Grâce à une autre faculté de la physique quantique, Bob et Alice n'auront, un jour, même plus à s'échanger une clé symétrique à travers un média physique comme une fibre optique. Alice génèrera une clé et celle-ci sera automatiquement générée chez Bob.

Magie ? Non principe d'**intrication quantique**. Les Chinois disent avoir déjà une solution entre deux entités éloignées de plusieurs centaines de kilomètres.

LE CONSEIL CRYPTO

À partir du web http://www.randomnumbers.info/ de l'université de Genève, vous pouvez utiliser une application, basée sur la physique quantique, pour générer des nombres purement aléatoires.



Vous choisissez la quantité de nombres que vous souhaitez générer et leur valeur maximale. Dans cet exemple, on demande la génération de 3 nombres aléatoires compris entre 0 et 10.

Les éléments de la suite des trois nombres proposés, par exemple (3, 8 et 5) sont aléatoires. Vous pouvez vous en servir pour offrir des lots à 3 personnes dans une liste de personnes numérotée de 0 à 10, ici la 3e, 8e et 5e personne de la liste numérotée, sachant que des personnes peuvent être tirées au sort plusieurs fois, puisque la même valeur peut sortir plusieurs fois.

LE CHIFFREMENT HOMOMORPHE POUR UN CLOUD SECURISE

Le chiffrement homomorphe permet de travailler sur des données chiffrées sans avoir à les déchiffrer. Comment cette technologie s'applique-t-elle à la sécurité du Cloud ? Quel est son état de l'art ?

Le cloud public est une solution merveilleuse pour stocker les données. Pour un coût de services maitrisé, les entreprises ont la possibilité de disposer de tout l'espace qui leur est nécessaire, sans avoir à investir sur des serveurs et des disques supplémentaires, quand un besoin de plus d'espace de stockage apparait.

Comme la sécurité n'est pas, dans la plupart des cas, le métier des entreprises utilisatrices des services du Cloud, celles-ci peuvent s'assurer, par contrat, que leurs données sont bien en sécurité dans le cloud de leur prestataire. Voilà pour la **disponibilité des données** et leur **protection périmétrique.** Quand les entreprises ne maîtrisent pas la sécurité et la sûreté de leurs données numériques, celles-ci font face à de multiples menaces et rares sont les entreprises en mesure de les contrer efficacement. Les prestataires de cloud sont par contre censés bien maîtriser la cybersécurité et censés avoir les compétences dans ce domaine.

Mais qu'en est-il de la **confidentialité** et de l'**intégrité** des données confiées dans un cloud extérieur ?

Rappelons que la confidentialité d'une information est l'assurance qu'elle ne pourra être lue que par des personnes autorisées à en prendre connaissance, alors que l'intégrité est l'assurance qu'elle ne peut être écrite ou modifiée que par les personnes également autorisées à le faire.

Une solution serait de n'utiliser l'espace d'un cloud public que pour héberger les données non sensibles. Mais alors on se prive de l'avantage de l'espace quasi infini que propose

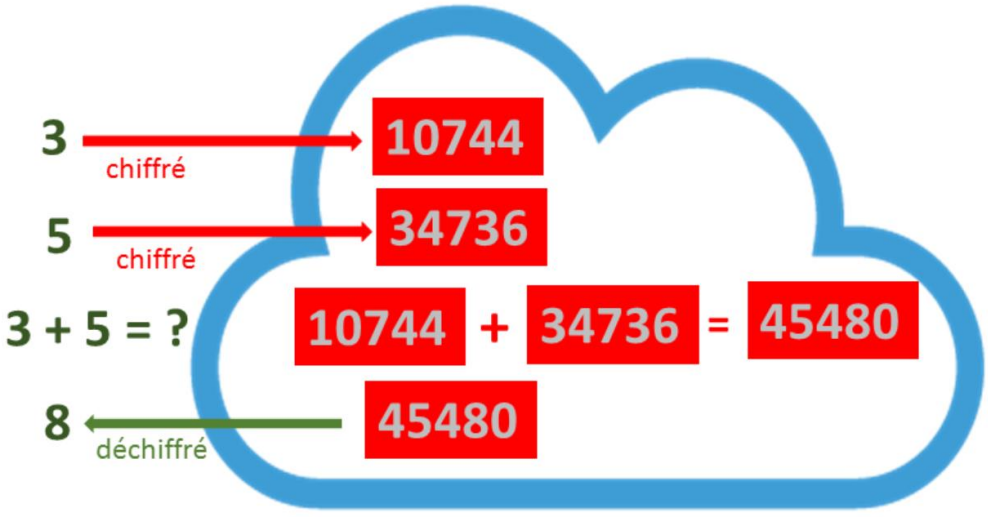
le cloud pour les héberger. Chiffrer les données sensibles et les confier dans un cloud public est aussi une solution, mais qui gère les clés ? L'idéal est bien sûr, pour les entreprises qui confient leurs données dans un Cloud public, de gérer elles-mêmes les clés de chiffrement. Confier la gestion des clés de chiffrement à son prestataire de cloud trouve ses limites dans la confiance que les entreprises clientes accordent à leur prestataire. Gérer les clés de chiffrement en interne dans l'entreprise est une tâche complexe pour qui la sécurité des données numériques n'est pas le métier. Confier la gestion des clés à un autre prestataire différent de celui qui héberge les données chiffrées semble être une meilleure solution. La confidentialité et l'intégrité des données sensibles sont ainsi assurées.

Si les données sont stockées en clair chez le prestataire, alors le client peut en disposer pour effectuer des traitements mais elles sont accessibles à toute personne mal intentionnée disposant d'un accès privilégié chez le prestataire. Si les données sont stockées chiffrées, il est alors difficile d'en disposer pour effectuer des traitements.

Le problème qui se pose est donc : « *comment effectuer des traitements sur les données chiffrées ?* ». Il est bien évident que, par exemple pour une addition entre deux nombres qui sont chiffrés, la somme des deux nombres chiffrés ne donne pas, lors du déchiffrement, le résultat attendu. Il est bien sûr possible de rapatrier en interne les données à traiter, les déchiffrer pour effectuer les traitements, chiffrer les résultats et les remettre éventuellement dans le Cloud. Cette solution n'est, de toute évidence, pas vraiment jouable.

Alors le Cloud est-il condamné à ne rester qu'un espace de stockage, sans permettre un espace de calcul ? Les données à manipuler ne pourraient-elles pas rester dans le Cloud, chiffrées, et les traitements s'effectuer sur les données chiffrées en donnant le bon résultat lors du déchiffrement chez l'utilisateur ?

Il existe une solution très élégante déjà opérationnelle pour certains traitements, et que les cryptologues font avancer dans les centres de recherche pour prendre en compte tous les traitements possibles, c'est le **chiffrement homomorphe**.



Avec ce type de chiffrement, le Cloud n'est plus seulement un espace de stockage sécurisé mais devient également un espace de calcul et de consultation sécurisé. Il va vraiment servir, non seulement à héberger l'information sensible, mais aussi à l'utiliser … sans la sortir du Cloud. Seuls les résultats après traitements seront sortis pour être déchiffrés et exploités.

Dans le schéma ci-dessus, on veut obtenir le résultat de l'addition de deux nombres confiés chiffrés au Cloud, « **3** » et « **5** ». Mettons que le résultat homomorphe chiffré de 3 est « 10744 » et que le résultat homomorphe chiffré de 5 est « 34736 ».

Dans le Cloud s'opère l'addition homomorphe 10744 + 34736, qui donne 45480. Le déchiffrement homomorphe de 45480 donne … « 8 », ce qui est le résultat attendu.

Ainsi le Cloud serait devenu non seulement un espace de stockage mais aussi un espace de calculs et de traitements ?

C'est du moins ce qu'on souhaiterait en attendre, mais aujourd'hui le chiffrement homomorphe ne fonctionne que pour certaines opérations. Il ne permet pas, par

exemple, de consulter une base de données chiffrée pour obtenir le résultat souhaité en clair. Si un chiffrement dit « **pleinement homomorphe** » existait dès aujourd'hui, si tout traitement pouvait être réalisé sur les données chiffrées confiées dans un Cloud public, leur confidentialité et de leur intégrité seraient garanties. Mais on n'en est pas encore là, et les opérations qui peuvent déjà fonctionner posent quelques problèmes de performance, car les clés de chiffrement ont encore quelques problèmes de (grande) longueur, mais les recherches vont bon train pour offrir cette faculté inestimable.

Remarquons, sans verser trop dans la technique, et en simplifiant, que l'algorithme de chiffrement utilisé par le RSA, qui est à la base du chiffrement à clé publique, est, par nature, homomorphe pour la multiplication. En effet, le produit de deux nombres chiffrés est égal au chiffré du produit des deux nombres. Ce résultat, une fois déchiffré, est le même que si on fait la multiplication des deux nombres en clair. Un chiffrement homomorphe qui fonctionnerait pour l'addition **ET** pour la multiplication est appelé « chiffrement doublement homomorphe ». On s'en approche aujourd'hui, mais avec des problèmes de largeur des éléments chiffrés et de bruits numériques engendrés par les traitements. La difficulté du chiffrement homomorphe est de maintenir le "bruit numérique", que les opérations engendrent, au-dessous d'un seuil raisonnable sinon les algorithmes divergent et tout devient indéchiffrable. Nous n'étudierons pas ces problèmes complexes ici, mais nous pouvons espérer que les mathématiciens trouveront une solution élégante aux problèmes posés par le chiffrement doublement homomorphe.

LE CHIFFREMENT « CHERCHABLE »

Le chiffrement homomorphe ne doit pas être confondu avec le chiffrement cherchable qui permet de spécifier une procédure de déchiffrement à un résultat de calcul dans le domaine chiffré. Ce dernier type de chiffrement offre une solution pour consulter une base de données chiffrée, obtenir un résultat qui, déchiffré, donne le résultat attendu.

APPLICATION PRATIQUE : LE VOTE PAR INTERNET

Comme application pratique, voyons comment le chiffrement homomorphe fournit une solution au vote par Internet. Nous ne parlons pas ici des machines de vote électronique, mais de l'électeur qui vote à partir de son navigateur et de sa connexion internet.

Avec l'utilisation des algorithmes de El Gamal, le produit homomorphe des bulletins de votes chiffrés est égal à la somme homomorphe chiffrée des bulletins de votes. Les choix des votants ne sont jamais déchiffrés. A la clôture du scrutin, on effectue une multiplication homomorphique de tous les bulletins de votes. On obtient la somme chiffrée et on la déchiffre. Cette somme est donc le résultat des votes qui est obtenu immédiatement. Oui, le chiffrement homomorphe de El Gamal (entre autres cryptologues qui ont fait avancer cette technologie) permet cela.

Les bulletins sont chiffrés par la clé publique de l'urne, le déchiffrement de la somme des bulletins se fait par la clé privée de l'urne. Cette clé privée peut être répartie en plusieurs morceaux détenus par le président du bureau de vote et ses assesseurs. A l'ouverture du scrutin, le président et ses assesseurs reconstituent la clé de déchiffrement et obtiennent quasi immédiatement le résultat attendu.

Que le produit homomorphe des bulletins de votes chiffrés soit égal à la somme homomorphe chiffrée des bulletins de votes est une belle application de ce type de chiffrement.

Pour ceux qui aiment les formules mathématiques, si *Ek(mn)* est le bulletin de vote mn chiffré avec la clé publique k de l'urne :

$$E_k(m_1) \times E_k(m_2) \times \ldots \times E_k(m_n) = E_k(m_1 + m_2 + \ldots + m_n)$$

Cette méthode est élégante dans sa simplicité d'utilisation. Les bulletins dans l'urne ne sont jamais déchiffrés pourtant on connaît le résultat de la somme des votes qui est d'ailleurs le seul renseignement qui est intéressant et non confidentiel après la fermeture du scrutin.

Cette méthode a déjà été utilisée pour les élections des représentants des Français résidant à l'étranger. Elle peut être utilisée aussi pour les élections des représentants du

personnel ou dans les conseils d'administration des entreprises. Mais pour les élections présidentielles, sénatoriales ou législative, elle n'est pas autorisée en France. Nous ne parlons ici que du fondement cryptologique d'une application pratique d'un chiffrement homomorphe qui fonctionne. Le vote par Internet qui n'impose pas de passer par un isoloir, et qui ne nécessite pas la cérémonie républicaine du dépouillement des votes, est-il à recommander ? C'est un débat intéressant dans notre démocratie mais dans lequel nous ne prendrons pas part ici.

Mais si le chiffrement est indispensable pour protéger une donnée, il est une autre méthode qui consiste à cacher un message en clair dans un autre message en clair ; c'est la stéganographie.

CHIFFRER ? NON, PLUTOT DISSIMULER : LA STEGANOGRAPHIE

La **stéganographie** est une méthode vieille comme le monde pour assurer la **confidentialité** d'une information sensible. La cryptologie (*du grec κρυπτός : cacher*) consiste à brouiller une information pour la stocker ou la transmettre au travers d'un contenant qui lui n'a pas à être caché. La stéganographie (*στεγανός : couvrir*) consiste au contraire à stocker ou transmettre une information en clair, mais intégrée dans un contenant qui n'est pas caché. Ce qui est caché, et qui assure la confidentialité, c'est que ce contenant contient l'information qu'on souhaite n'être lue que par celui qui en a l'autorisation, mais bien malin celui qui peut le savoir si on ne le lui a pas dit, surtout si le contenant semble anodin. En somme, pour ne pas découvrir le contenu d'un message caché, il suffit de ne faire savoir que son contenant contient ce message, qu'à son correspondant.

Il y a 2500 ans, quand les Perses ont levé une flotte pour envahir la Grèce, un Grec vivant à Babylone, ayant eu vent de l'affaire eut l'idée de graver un message d'alerte à destination d'Athènes, sur une planchette en bois qu'il recouvrit de cire. Cette planchette traversant sans encombre tous les contrôles fut remise aux Grecs qui grattèrent la cire et surent ainsi que les trières perses allaient emporter l'armée d'invasion de la Grèce en passant par le détroit de Salamine. A l'heure dite, les Grecs les attendaient dans le détroit et lancèrent des embarcations enflammées sur la flotte d'invasion perse, causant la panique parmi les assaillants et un immense télescopage des navires perses. Et ce fut, grâce à ce renseignement fourni par ce procédé de stéganographie, la victoire navale de la bataille de Salamine. La Stéganographie avait ainsi acquis ses lettres de noblesse. Mais le procédé avait dû être utilisé bien avant, peut-être même, qui sait, dans les peintures rupestres de la grotte de Lascaux.

Un exemple littéraire célèbre de stéganographie se trouve dans la correspondance prêtée à un échange entre Alfred de Musset et George Sand. Le poème très romantique,

très fleur bleue d'Alfred de Musset cache un poème beaucoup plus terre à terre, c'est le moins qu'on puisse dire, quand on ne lit qu'un vers sur deux. Et le premier mot de chaque vers que George Sand lui a fait en réponse le renseigne sur le moment des retrouvailles. Encore fallait-il connaître la méthode pour retrouver ces messages cachés...

Voir en : http://5ko.free.fr/fr/sand.html

Bon, cet exemple célèbre d'acrostiche littéraire n'est sans doute pas d'Alfred de Musset ni de Georges Sand, mais qu'importe ici la vérité pourvu qu'on ait l'ivresse ! :-)

De l'encre invisible aux micro points cachant une information qui n'est lisible qu'à l'aide d'un microscope électronique, les techniques de stéganographie n'ont cessé de se perfectionner. Et bien sûr avec elles se sont développées les techniques de contre-mesures. Par exemple si un texte est écrit en utilisant de l'encre invisible (ça marche avec le jus de citron) qui ne se révèle qu'à une certaine température, il suffit de chauffer toutes les feuilles pour savoir si elles ne recèlent pas un message caché (quand le message ne se révèle pas spontanément parce qu'il fait trop chaud). Les micro points, eux, peuvent être décelés en lumière rasante, parce qu'ils sont, en général, plus brillants que le texte qui les contient.

Le numérique allait donner une impulsion aux techniques de stéganographie. Prenons comme exemple un message sensible caché dans une image.

UN MESSAGE CACHE DANS UNE IMAGE

Une image numérique est composée de petits carrés élémentaires, les pixels (picture éléments). Chaque pixel coloré dans une image en couleur est constitué de trois octets composant les couleurs fondamentales, rouge, vert et bleu. La combinaison des nuances de ces trois couleurs donne la couleur du pixel. Un pixel est composé ainsi d'un octet de rouge, d'un octet de vert et d'un octet de bleu. Un octet, élément de 8 bits, peut prendre 256 valeurs.

Si par exemple l'octet de bleu d'un des pixels se termine par un bit à 1 dans la couleur naturelle du pixel, et que le message que l'on veut cacher dans l'image demande qu'on le change par un bit à 0, on aura changé 1/256^{e} de la teinte en bleu du pixel. Un être humain ne peut le détecter sur l'image. On va utiliser chacun des octets des pixels constituant l'image couleur, pour cacher le message sensible. Seuls ceux qui savent que l'image a été modifiée pourront donc s'intéresser aux bits de poids faible de chacun des trois octets de chaque pixel et reconstituer le message caché. Cette technique a pour nom LSB (Least Significant Bit).

Donc si on réserve pour dissimuler le message le bit de poids faible de chaque octet composant les pixels d'une image, ces pixels parfois changent, parfois pas, on peut disposer du 8eme de la taille de l'image pour cacher son message. Voir, à la fin,

l'application pratique avec l'utilisation du logiciel libre *SteganograFree*. Si on a besoin de plus de place pour placer son message sensible, on peut prendre plusieurs bits de poids faibles de chaque octet (le dernier et l'avant dernier par exemple, mais l'image qui contient le message sera un peu plus dégradée. Mais du moment que ce n'est pas perceptible à l'œil humain …

STEGANOGRAPHIE ET CRYPTOLOGIE, DEUX SOLUTIONS COMPLEMENTAIRES

La stéganographie corrige un gros problème de la cryptographie. Si un fichier est chiffré, c'est qu'il présente en principe une certaine importance puisqu'on a pris la peine de le chiffrer et les cryptanalystes vont essayer de le décrypter. Mais si un fichier n'est pas chiffré, seulement dissimulé, même en clair dans un élément en clair, comment connaître son existence ? En somme, si la cryptographie repose sur le fait que le message ne sera pas compris, la stéganographie repose sur le fait que le message ne sera pas trouvé. De plus, il n'est pas interdit de chiffrer également le message sensible, si la sensibilité du fichier, de plus caché dans son contenant, exige deux précautions plutôt qu'une. A l'écriture secrète de la cryptologie s'ajoute ainsi l'écriture discrète de la stéganographie.

Et de même que par ce procédé on peut dissimuler un texte dans une image, on peut également dissimuler une image ou tout autre fichier dans une image. On peut aussi cacher un son dans un son. Dans un fichier .wav par exemple on peut utiliser les fréquences non audibles par l'oreille humaine pour dissimuler un secret. C'est la **stéganophonie**.

LES CONTRE-MESURES

De même que la cryptanalyse a pour but de décrypter un message chiffré sans connaître la clé de chiffrement, la stéganalyse a pour but de permettre de découvrir une information cachée dans son contenant, ou au moins de la détruire.

Si on possède l'image originale et l'image modifiée, et qu'on se doute de quelque chose, il est évidemment possible de découvrir ce que cache l'image modifiée. Si on convertit une image BMP qui contient une information cachée, en image JPEG, la compression opérée détruit le message caché.

Le mieux serait d'interdire tout échange, mais ce n'est pas réaliste donc la stéganographie est une technologie d'avenir.

WATERMARKING OU TATOUAGE

Le watermarking est une technologie qui assure non pas la confidentialité d'une image mais qui peut établir les droits d'auteur en dissimulant un copyright dans l'image. C'est une des applications de la stéganographie. Le watermarking ne doit pas altérer la qualité visible de l'image, tout en restant invisible. Il doit résister aux altérations de l'image, comme par exemple les compressions, et doit empêcher sa suppression. Autant de contraintes qui font du watermarking un espace de recherche actuel très fécond.

APPLICATION PRATIQUE : CACHER UN MESSAGE DANS UN LOGO

Il n'est évidemment pas question de changer chaque bit de poids faible de chaque pixel à la main. Des outils conviviaux sont disponibles, comme dans l'exemple ci-dessous le logiciel libre SteganograFree.

Le but est ici de cacher le message « *Vous ne pouvez pas vous douter que le logo LinkedIn cache un message secret* » dans le logo LinkedIn. On charge le logo qui, dans le cas de SteganograFree, doit être une image au format BMP, on écrit le message, et on protège le tout par un mot de passe.

Pour retrouver le message, on charge le logo et on entre le mot de passe. On obtient le message secret. Simple et très efficace !

A PROPOS DE L'AUTEUR

Gérard Peliks *gerard.peliks (at) forumatena.org*

Travaille depuis plus de 20 ans dans le domaine de la sécurité de l'Information. Président de l'association CyberEdu (créée par l'ANSSI), président de l'atelier sécurité et VP de Forum ATENA, Ingénieur diplômé, il a travaillé pour Airbus Defence & Space Cybersecurity. Lieutenant-colonel de gendarmerie dans la Réserve Citoyenne de Cyberdéfense (DGGN) et membre du Conseil d'Administration de l'Association des Réservistes du Chiffre et de la Sécurité de l'Information (ARCSI), il coorganise, sur une base mensuelle, les « Lundi de l'IE » du Cercle d'Intelligence économique du Medef Ile-de-France.

Chargé de cours sur la cybercriminalité/cybersécurité dans des mastères d'écoles d'ingénieurs, en particulier à l'Institut Mines Télécom et à l'Institut Léonard de Vinci, son activité principale aujourd'hui est de porter l'esprit de cybersécurité / Cyberdéfense auprès du citoyen en organisant des présentations pédagogiques de tous niveaux et en écrivant des articles de vulgarisation sur les dangers du cyberespace et des contre-mesures pour en diminuer les risques.