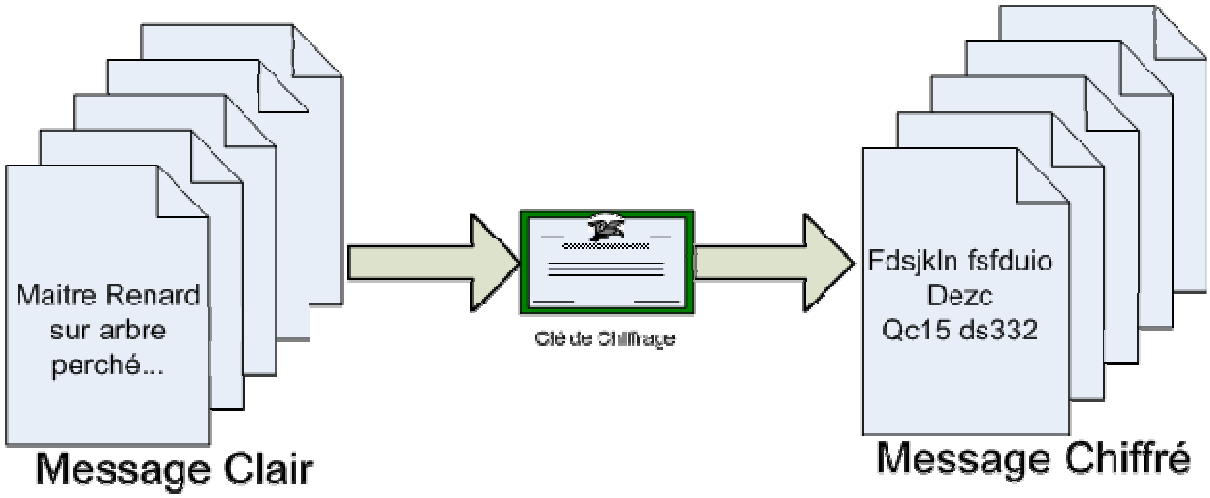
Avertissements :
Le contenu de ce document est sous licence GPL. Le document est librement diffusable dans le contexte de cette licence. Toute modification est encouragée et doit être signalée à olivier[chez]thebaud.com
Les documents ou applications diffusées sur thebaud.com sont en l'état et sans aucune garantie ; l'auteur ne peut être tenu pour responsable d'une mauvaise utilisation (au sens légal comme au sens fonctionnel). Il appartient à l'utilisateur de prendre toutes les précautions d'usage avant tout test ou mise en exploitation des technologies présentées.

Objet :	**Explications sur la cryptographie**	Date :	**22/02/2008**
		Version :	**1.0**

1 - Chiffrement ou Cryptage, Déchiffrement ou Décryptage ?

La confusion est fréquente.

- Le **chiffrement** consiste à transformer une donnée lisible (ou clair) en une donnée illisible (ou incompréhensible par un humain logiciel, ou cryptogramme) sans posséder la clé qui permet de déchiffrer la donnée.
- Le **déchiffrement** est logiquement l'opération inverse du chiffrement.
- **Cryptage** est un anglicisme ambigu issu de *Encryption* qui signifie en français… Chiffrement.
- **Décryptage** désigne l'acte de rendre clair un message chiffré sans en posséder la clé.



2 - Définitions complémentaires utiles

Cryptogramme : message chiffré.
Crypto-Analyse : technique qui vise à étudier les messages chiffrés ou Cryptogrammes en vue de les rendre lisible (ou de les décrypter)
Cryptographie : science qui vise à créer des cryptogrammes à partir d'algorithmes de chiffrage.

3 – A quoi çà sert ?

Tous ces mots ont pour vocation , en vrac à :

- empêcher la diffusion d'information sensible ou confidentielle = *confidentialité*
- mettre en évidence si un document a été modifié (une commande de client, un virement bancaire,…) = *intégrité*
- faire en sorte que la compromission d'une donnée coûte trop chère à l'attaquant au regard du contenu de la donnée = *confidentialité & intégrité*
- protéger des mots de passe = *confidentialité & intégrité*
- rendre illisible des bandes de sauvegardes, si elles devaient êtres perdues ou volées (?) = *confidentialité*
- identifier une personne ou un système = *authentification*
- empêcher la répudiation : c'est faire en sorte qu'un correspondant récepteur recevant une donnée (mail, fichier,…) peut vérifier qu'il provient bien du destinataire annoncé ; de même qu'un correspondant émetteur envoyant une donnée sera assuré que le récepteur a bien reçu le document envoyé = *non-répudiation*

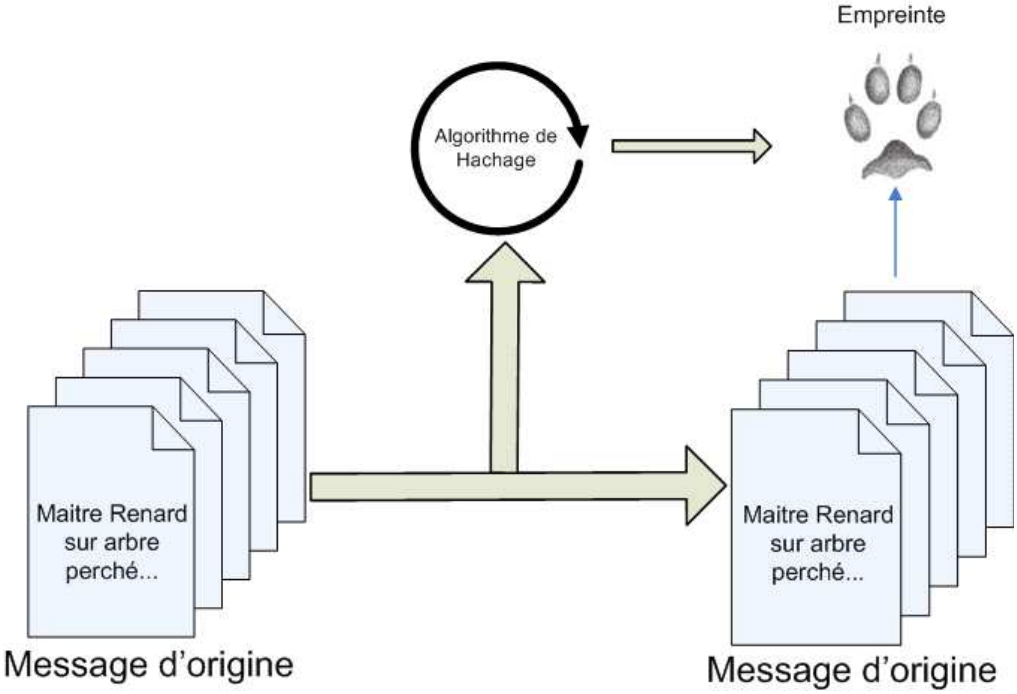
4 - Hachage ou Signature ?

Dans les deux cas, il s'agit de générer une empreinte quasiment unique d'une donnée (mail, fichier, …) afin de s'assurer que le contenu n'a pas été modifié sinon l'empreinte serait différente.
L'empreinte peut être appelée aussi condensat, hash, ou encore somme de contrôle.
On retrouve ce mécanisme de prise d'empreinte dans la gestion de mot de passe associé aux mécanismes d'authentification, dans certains protocoles de communication, l'échange de mails, l'échange de fichiers (notamment les exécutables),…
Il y a quelques différences importantes entre ces deux techniques :

- Le **Hachage** a pour vocation de vérifier que le contenu n'ait pas été modifié (intégrité). Le hachage produira toujours la même empreinte numérique (sauf à utiliser une variable complémentaire telle qu'une clé de sel telle que date/heure).
- La **Signature** a pour vocation d'identifier l'auteur d'un document (identité) et de s'assurer que le document n'ait pas été modifié (intégrité). La signature ou vérification de signature nécessite de posséder une clé.

5 - Hachage - Fonctionnement

Un algorithme va digérer un message ou document, puis en sortir une suite numérique d'une certaine longueur, qui sera l'empreinte du document.
Un algorithme donné produira toujours la même longueur de l'empreinte numérique. La fonction de Hachage ne permet de reproduire le message d'origine (en principe), ce n'est donc pas une fonction de cryptage.



Les algorithmes de hachage sont légions mais ceux utilisés sont souvent :

MD4 : Message Digest 4 (RFC 1320) qui génère une empreinte de 128 bits. Cet algo est désormais abandonné par une faiblesse de conception et

tant la probabilité d'avoir le même résultat pour deux messages différents est important.

MD5 : Message Digest 5 (RFC 1321) améliore MD4, l'empreinte reste sur 128 bits.. Mais cet algorithme est désormais considéré comme non sûr pour usage cryptographique : une équipe de Chinois a pu démontrer pouvoir reproduire à partir d'une empreinte e1 (calculée à partir d'un message X), un nouveau message (Y) capable de produire une empreinte e2 identique à e1. Ce qu'on appelle une collision complète (dans un contexte peu sécurisé puisque le second message n'a pas été trouvé de manière aléatoire…)
Il n'empêche qu'il est encore fortement utilisé pour vérifier l'empreinte de fichiers téléchargés sur Internet, pour le cryptage de mot de passe sur les sites Web (fonctions PHP disponibles)

SHA-0 : Secure Hash Algorithm 0, cette première version mise au point en 1993 fut vitre abandonnée à cause de deux failles permettant des collisions, failles associées à la présence de la NSA.

SHA-1 : Secure Hash Algorithm 1 est sorti en 1995 sous la coupelle de la NSA, il produit une empreinte de 160 bits mais reste parmi les algorithmes peu sûrs bien qu'il faille un puissance de calcul importante (à ce jour) pour permettre une collision qui ne sera trouvée à partir d'un message aléatoire.
SHA-2 (aussi appelé SHA-224, SHA-256, SHA-384, SHA-512) : Secure Hash Algorithm xxx bits. Algorithme dérivé de SHA1, publié par la NSA en 2000, les versions 256 et 512 sont répandus et dits sûrs : une étude de 2003 a montré que l'algorithme n'avait pas la fragilité des algorithmes rencontrés sur MD5 et SH1. A suivre.

Whirlpool : conçu dans le cadre d'un projet européen, cet algorithme génère des empreintes de 512 bits. La longueur de l'empreinte et le fait que l'algorithme travaille sur des registres 64 bits en font un système assez consommateur pour le processeur et la mémoire (notamment sur environnements embarqués et basés sur des architectures 32 bits), mais reste exceptionnellement robuste, normalisé (ISO 10118-3 :2004 pour sa version finale) et *libre de droits*.
Réf :
http://paginas.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html

D'autres algorithmes de hachage peuvent être rencontrés mais présentent pour la plupart 1 – une implémentation peu rencontrée, 2- des failles (collisions) mises en évidence. Citons notamment FFTH, RIPEMD, LANMAN Hash, Tiger.

6 – Chiffrement / Déchiffrement

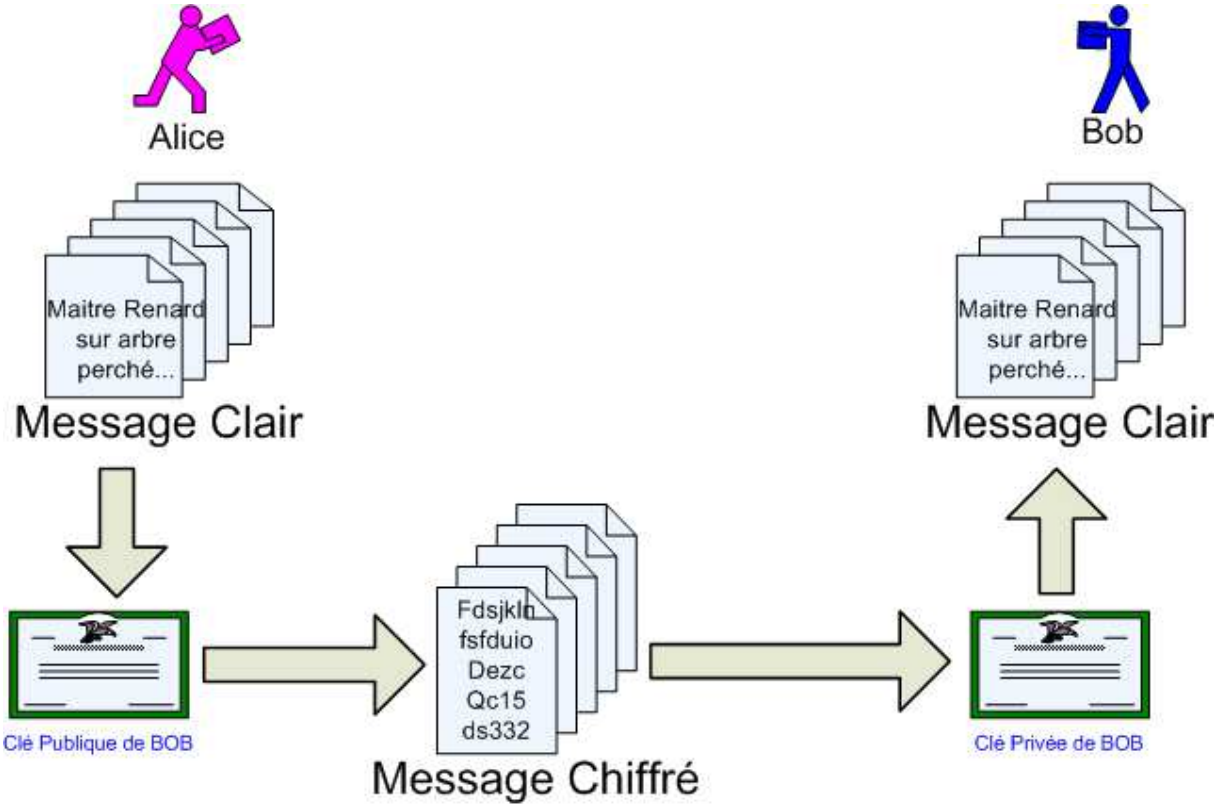
Il existe plusieurs grands principes de chiffrement (seuls les deux premiers seront expliqués) :

1- basé sur des **clés asymétriques**, la clé de chiffrement est différente de la clé de déchiffrement. C'est pratique lorsqu' il s'agit protéger un document et de faire en sorte que seules les personnes qui en possèdent la clé « complémentaire » pourront le déchiffrer. Sachant que cette clé ne permet pas de rechiffrer le document.
2- basé sur des **clés symétriques**, la même clé permet de chiffrer et déchiffrer. Utile lors de chiffrement UN vers UN (comme par exemple partiellement SSL entre un serveur Web et un navigateur Web, PGP, …).
3- Puis aussi …. par substitution, par transposition, polyalphabétique, par dissimulation, et par stéganographie.

7 – Chiffrement Asymétrique ou cryptographie à clé publique

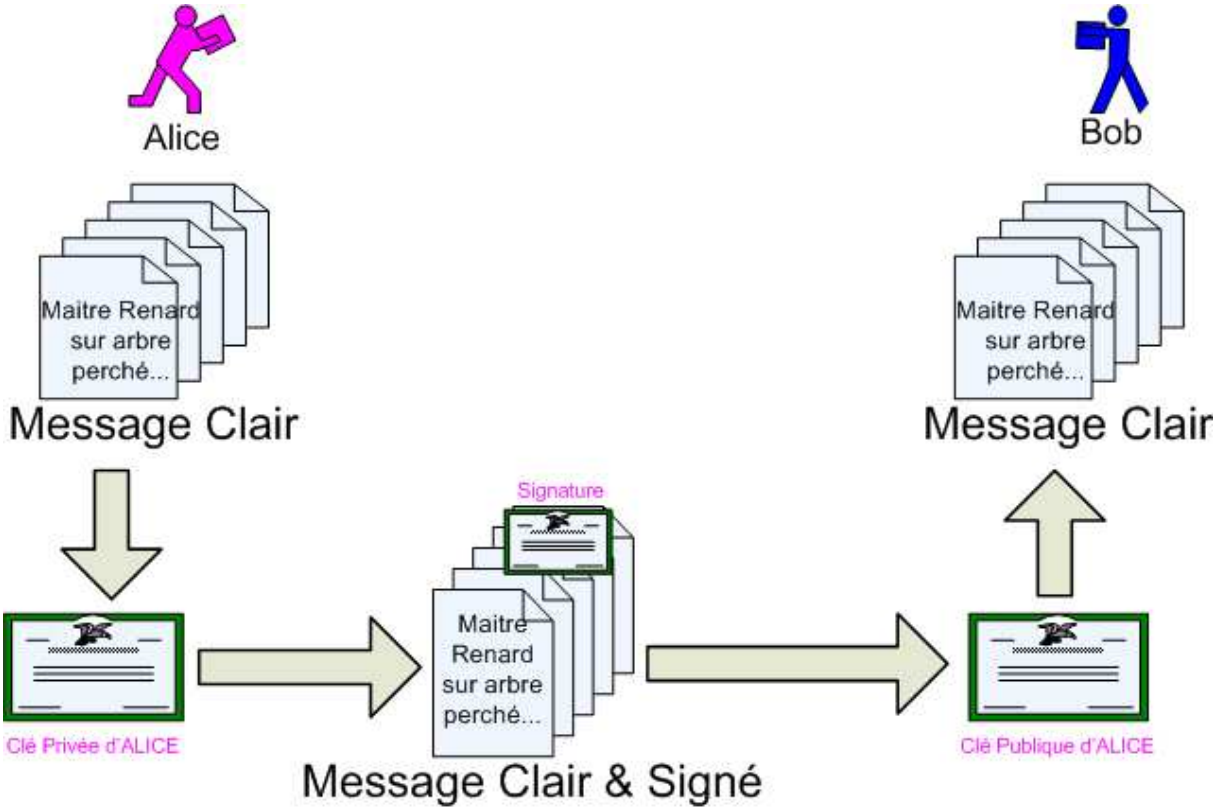
Le fonctionnement repose sur une paire de clés liées mathématiquement. On considère l'une d'entre elle comme privée, et l'autre publique (qui pourra donc être diffusée). Le contenu de la première clé ne peut être retrouvé à partir de la seconde clé. Les opérations se font à sens unique, on ne peut revenir en arrière (principe de la trappe). Ce retour en arrière est toutefois possible en possédant la clé privée.

Traditionnellement les exemples de chiffrement font appels à Alice et Bob. Allons-y pour un exemple de chiffrement de message dans un but de rendre **confidentiel** le message :



1 – Alice veut envoyer un message confidentiel à Bob
2 – Bob génère une paire de clé Publique / Privée
3 – Bob fournit à Alice sa clé publique
4 – Alice chiffre le message à partie de la clé publique de Bob
5 – Alice envoie le message chiffré à Bob
6 – Bob déchiffre le message d'Alice avec la seule clé en mesure de le faire : sa clé privée.

Une autre utilisation d'un système de chiffrement asymétrique est dans le but de garantie l'origine du message ou de **signé** ce message.
Autrement dit, Alice souhaite communiquer à Bob un message (dont le contenu n'est pas nécessairement confidentiel) ; le challenge est qu'Alice veut s'assurer que le message que recevra Bob sera bien celui qui a été envoyé, et qu'il n'aura pas soumis de modification sur le chemin.



1- Alice veut envoyer un message à Bob et se garantir de l'intégrité du message
2- Alice génère une paire de clé Publique /Privée
3- Alice fournit à Bob sa clé publique
4- Alice génère une signature à partir du message et de sa clé privée
5- Alice envoie le message signé à Bob
6- Bob utilise la clé publique d'Alice pour vérifier que la signature correspond au contenu exact du message.

Bien sûr, deux correspondants souhaitant s'échanger des données confidentielles et signées pourront mixer les deux techniques à condition de s'échanger mutuellement leur clé publique.

Un avantage de ce système est la diffusion facilité des clés publiques sans compromission et également la robustesse des algorithmes basés sur des problèmes mathématiques complexes (factorisation , logarithme). De ce fait, un chiffrement à clés asymétriques est considéré comme 1000 fois plus lent qu'un chiffrement à clé symétriques.

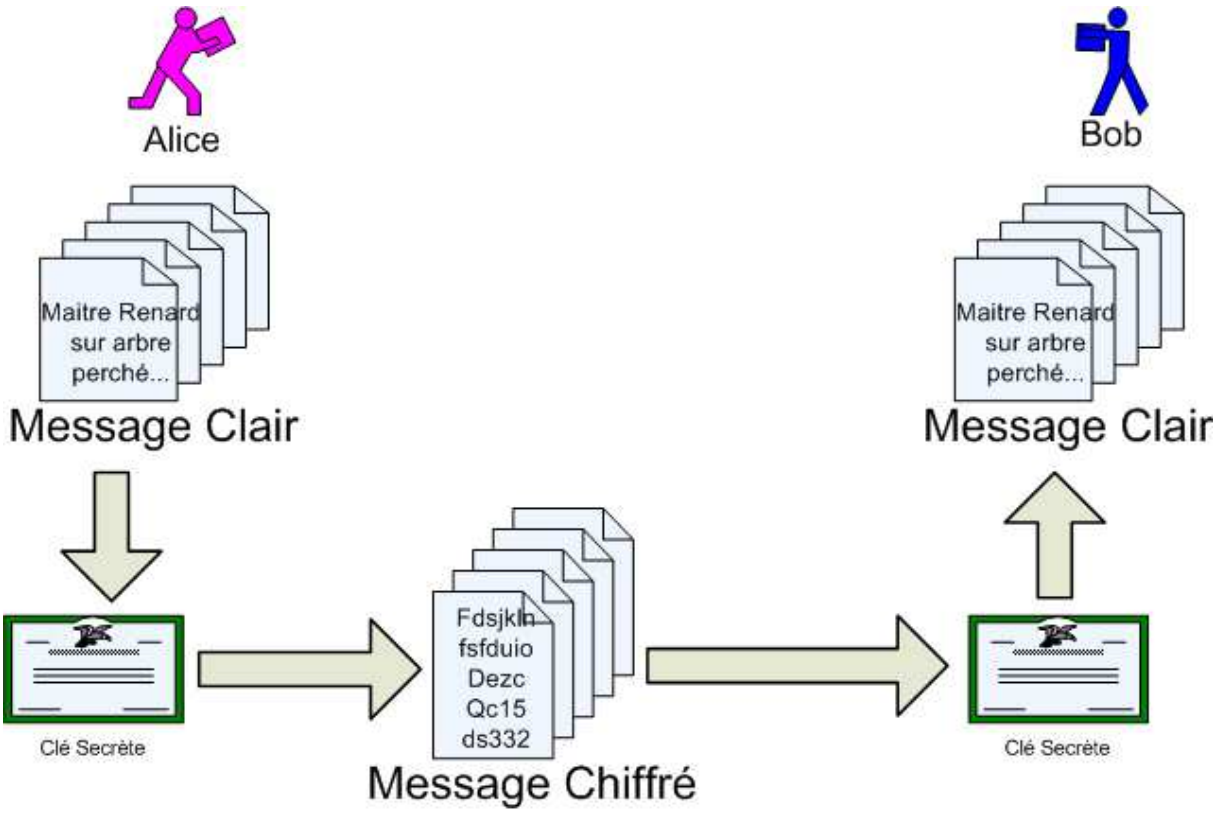
Des algorithmes asymétriques utilisés :

RSA : Rivest Shamir Adleman, du nom des 3 personnes qui l'ont créé (1977) et breveté en 1983. C'est le plus répandu et son brevet a désormais expiré (2000). Bien que cet algorithme puisse être « cassé » pour des clés de 256 bits avec ordinateur classique, 512 bits avec un réseau de processeurs, l'algorithme reste sûr pour des clefs RSA de 1024 bits minimum, voire 2048, 4096.

EL Gamal : du nom de son créateur, c'est un autre algorithme basé sur le protocole Diffie & Helmannqu'on retrouve notamment dans les dernières versions de PGP ; algorithme qui n'est pas sous brevet. A noter que le message chiffré est deux fois plus long que le message d'origine : ce qui peut engendrer quelques complications en terme de stockage ou de communication. Certaines implémentations d'ElGamal génèrent parfois des risques quant à l'exposition possible de la clé pricée.

8 – Chiffrement Symétrique ou algorithme à clé secrète

Son principe est très simple :



Pour que Alice et Bob puissent s'échanger un message confidentiel, donc chiffré, ils doivent tous deux posséder la même clé unique qui servira tant à chiffrer qu'à déchiffrer.
La faiblesse du système apparaît clairement dans l'échange de la clé, opération pourtant nécessaire au chiffrement/déchiffrement du message.
Le fonction d'un algorithme à clé secrète n'est pas exploitable dans le cas de non-répudiation ou de preuve d'origine.
Les algorithmes symétriques sont cependant parmi performants, capables de traiter un volume de données important en très peu de temps. Ils sont également robustes ou du moins dépendant de la taille de la clé : plus la clé sera importante, plus le nombre de combinaisons possibles pour trouver la bonne sera important.
C'est donc le délai nécessaire à la découverte de la clé parmi x combinaison qui définira la robustesse du système cryptographique.

Quelques algorithmes de chiffrement à clé secrète

DES : Data Encryption Standard, méthode de chiffrement utilisant des clés à 56 bits standardisée en 1976, à la demande de la NSA. Il était particulièrement utilisé dans les systèmes Unix.

TripleDES ou 3DES : c'est l'application successive de trois passes dans l'algorithme DES avec des clés en principes différentes (mais c'est parfois la même qui soit utilisée selon les implémentations…). La longueur de la clé est de 168 bits (3*56 bits). 3DES est en passe de disparaître également, parce que lent et dont le niveau de sécurité est peu performant. On le rencontre encore toutefois sur de nombreux équipements matériels.

Pour remplacer ces algorithmes vulnérables, une « compétition » a été initiée par le NIST afin de trouver le nouveau Standard de Chiffrement Avancé ; 5 algorithme candidats ont été présentés avec les votes suivants pour les 3 premiers Rijndael à 86 voix,Serpent à 59 voix, Twofish à 31 voix.

AES : Advanced Encryption Standard, adopté en 2001 est un algorithme développé par *Rijndael*, qui consomme peu de mémoire, facile à implémenter et utilise des clés de 128, 192 ou 256 bits. C'est devenu le remplaçant de DES et 3DES. Les données sont « passées » de 10, 12 ou 14 fois et entre chaque tour, elles subissent un masque XOR, les données XORées sont alors réintroduites, etc… AES est désormais utilisé dans la sécurisation Wifi, outils de compression, de cryptage de volumes,.. C'est le seul algorithme autorisé à porter le nom de AES.

BlowFish : conçu 1993 et entièrement libre, cet algorithme utilise des clés de 32 à 448 bits. Environ 5 fois plus rapide que 3DES, il reste encore très fiable (selon la taille de la clé). On le rencontre dans Ssh, PGP, TrueCrypt,…

TwoFish : reprend certaines caractéristiques de BlowFish mais avec des clés de 128 à 256 bits. Il était l'un des concurrent possibles à AES, mais n'a pas été retenu, pourtant plus rapide, il serait résistant et reste une bonne solution de 'secours', mais très peu utilisé.

Serpent : comme TwoFish, il a été présenté à l'évaluation pour devenir le AES mais n'a pas été retenu (trop proche de DES ?). Il n'existe pas d'attaque connue contre cet algorithme. Les clés générées peuvent être de 128, 192 ou 256 bits.

9 - Infrastructure à Clé Publique ou PKI (Public Key Infrastructure)

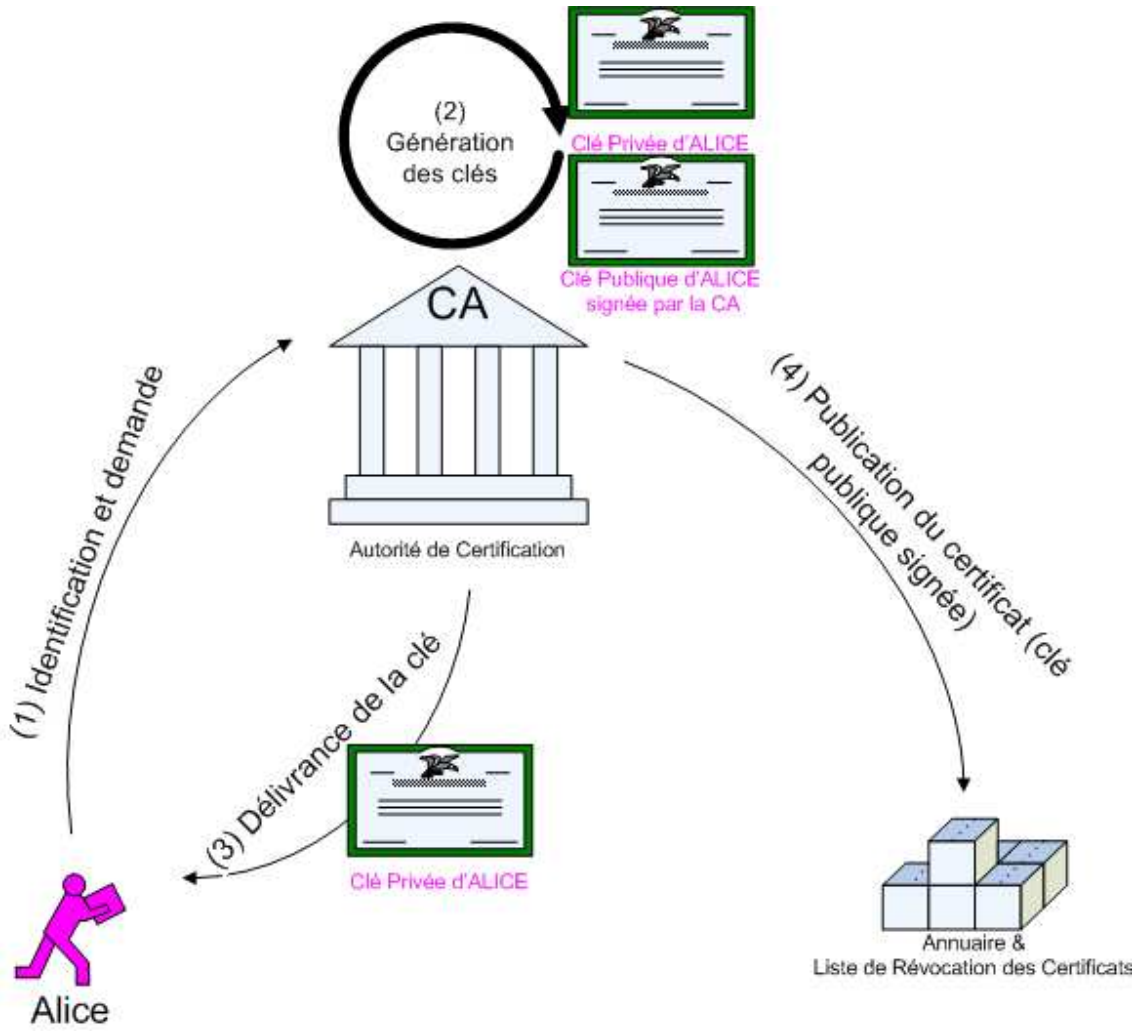
L'une des problèmatiques dans la gestion des clés de chiffrement est la diffusion de ces clés, la garantie de la provenance d'une clé (privée comme publique), la gestion de la durée de validité des clés, la mise en place d'un système de normalisation pour l'échange des clés , et la reconnaissance des participants cryptographiques par une tierce partie. C'est ce que permet de faire une Infrastructure à Clé Publique à titre externe (réseau internet, inter entreprise, …) comme à titre intra entreprise :

- assurer la non-répudication
- générer les paires de clés privée/publiques
- garantir la provenance d'un clé publique en y apposant sa propre signature (= certificat)
- définir/renouveller la durée de vie du couple de clés.

Voyons comment cela fonctionne-t-il :

Etape 1

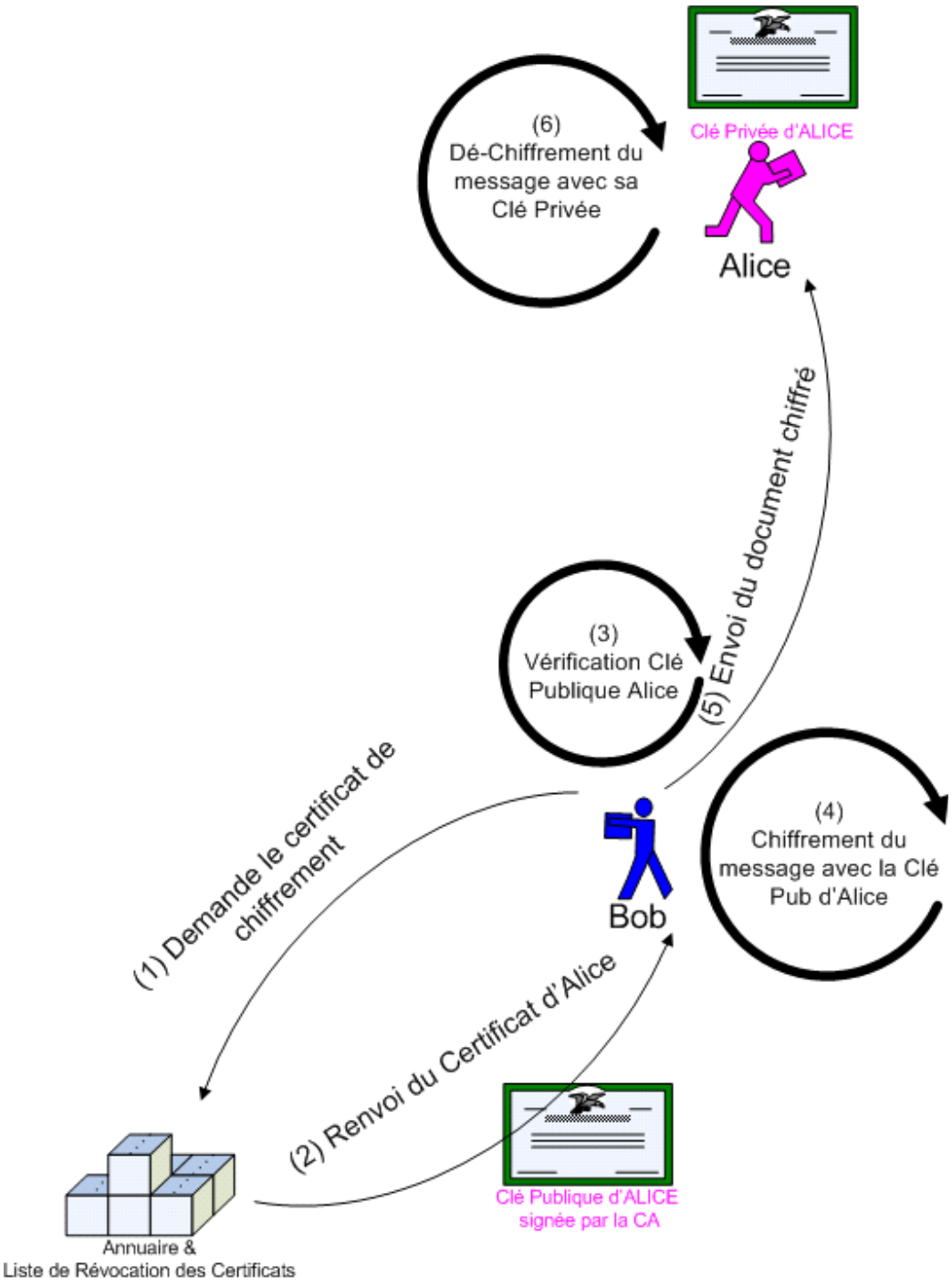
Bob souhaite chiffrer un message pour l'envoyer à Alice, ils vont faire appel à une PKI pour cette opération



1- Alice présente son identité auprès de l'Autorité de Certification (en fait cela devrait être l'Autorité d'Enregistrement mais dans la pratique, ces rôles sont cumulés par l'AC)
2- L'Autorité de Certificat ou CA génère une paire de clé (avec un clé publique signée par la propre clé privée de l'Autorité de Certification)
3- L'Autorité de Certification délivre la clé privée au demandeur
4- La CA publie le certificat dans une liste de révocations de certificats (CRL)

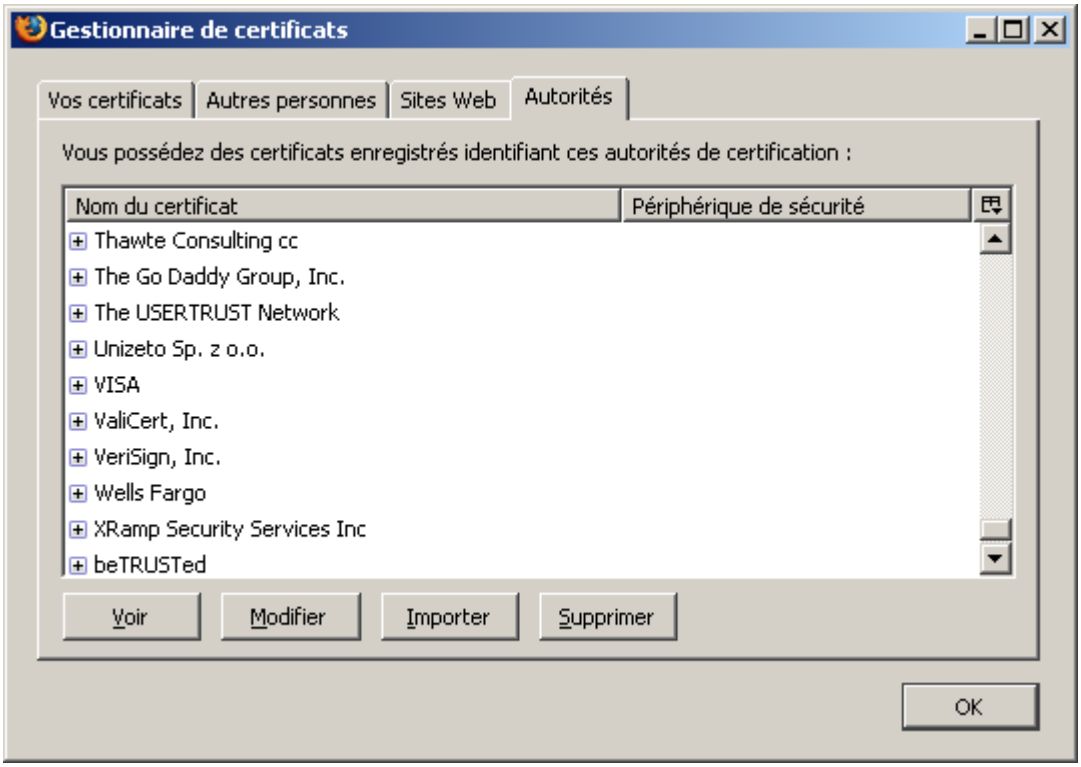
Etape 2

Bob souhaite chiffrer le document pour Alice. Il va contacter l'Annuaire (ou Alice) pour obtenir le certificat publié d'Alice.

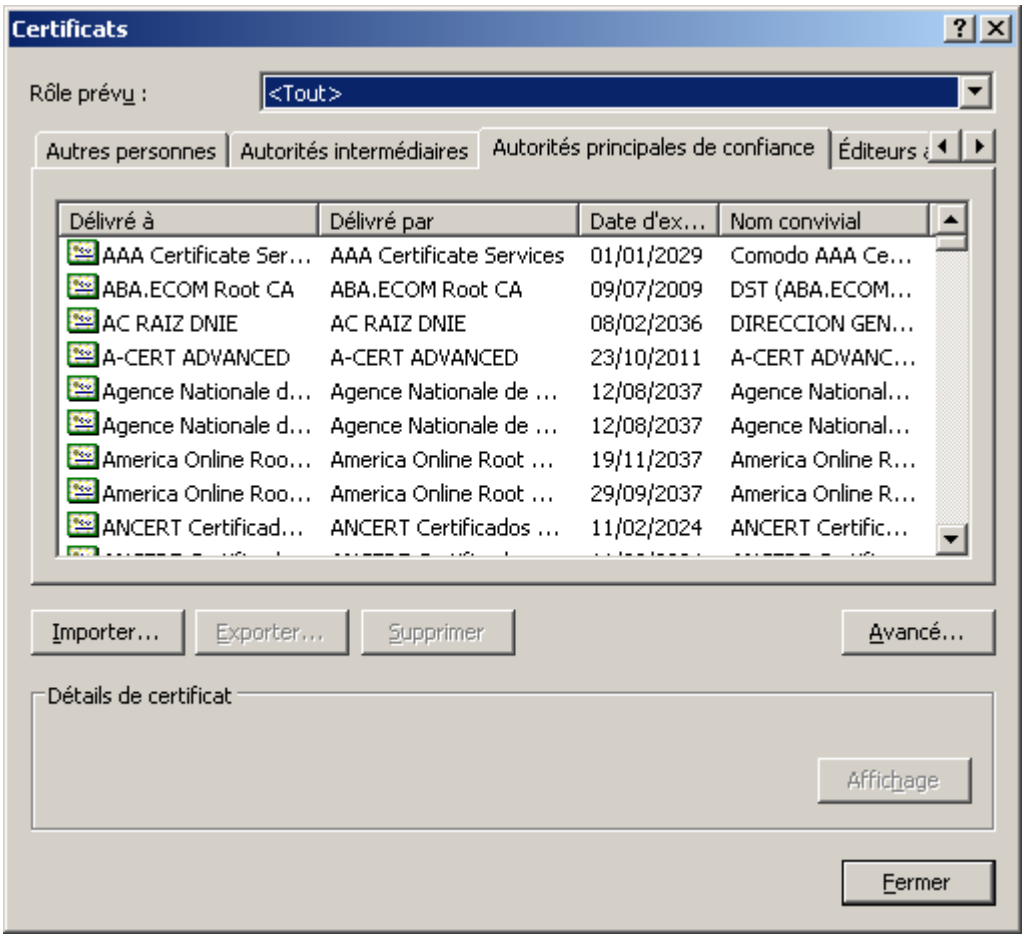


1 & 2 – Bob contacte l'Annuaire (ou Alice) pour obtenir le certificat idoine (qui contient la clé publique d'Alice signée par l'Autorité de Certification, la CA).
3 – Bob doit pouvoir vérifier que le certificat obtenu est bien celui d'Alice. Il va devoir vérifier la signature de la CA, donc devoir posséder la clé publique de la CA. C'est couramment chose immédiatement possible : un certain nombre de CA externes (ou Autorités de Certification Principales) sont déjà installées avec les OS et affichables facilement depuis son navigateur web

➔ FireFox 2



➔ IE 6



Ces certificats sont donc délivrés par des CA connues et répandues donc nativement installées avec les outils classiques de communication Internet.
Bob va donc retrouver LE certificat qui correspond à l'Autorité de Certification qui a signé la clé publique d'Alice et valider cette clé publique.

4 – Bob chiffre le message avec la clé publique d'Alice contenu dans le certificat

5 – Bob envoi son document à Alice, seule Alice pourra le déchiffrer.

6 – Alice utilise sa clé privée pour déchiffrer le message.

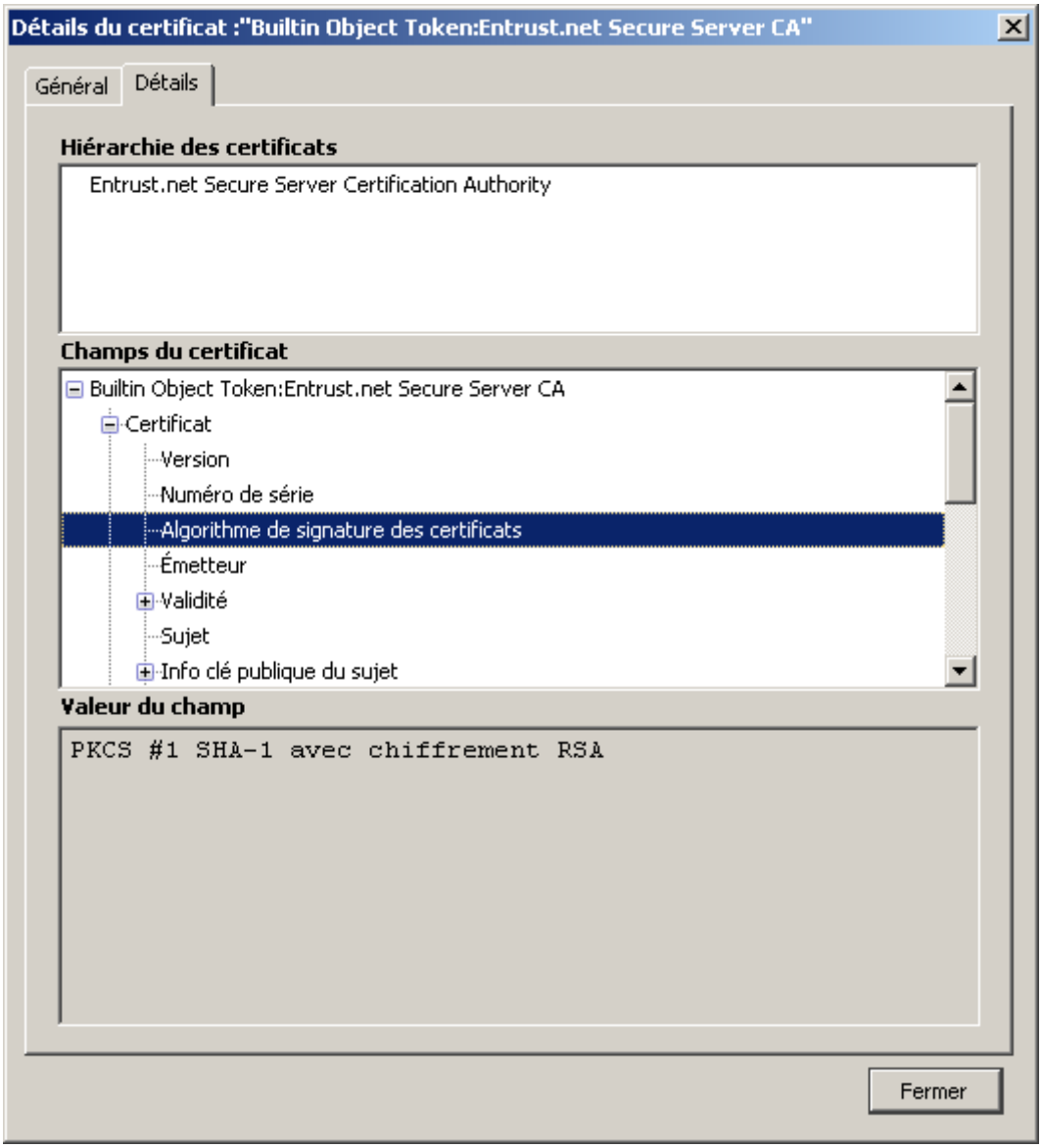
A titre d'exemple, plusieurs PKI sont aujourd'hui rencontrées. A commencer par celle proposée dans un environnement Active Directory 2000/2003, OpenPKI (http://www.openssl.org), OpenCA (http://www.openca.org)

10 - Quelques normes liées aux PKI

L'échange de certificats est normalisé par la norme X509 (v3) et contient :

- la version de la norme utilisée (v1, v2 ou v3)
- un numéro de série
- la signtaure de l'alogrithme utilisé
- Le nom de l'organisme délivrant
- La période de validité du certificat
- L'objet du certificat
- La clé publique
- La signature par la CA

Vous retrouvez toutes ces informations en clair lorsuqe vous affichez le contenu du certificat depuis votre navigateur :



IL existe par ailleurs un certains nombre de standards (ou spécifications) qui interviennent dans les systèmes d'échanges cryptographiques à clé publique, les **PKCS** (Public Key Cryptography Standards) déposés par le laboratoire du RSA.
Citons notamment :

PKCS#7 : spécification du format d'échange de messages cryptographiques

PKCS#10 : syntaxe de demande de certification composé d'un nom distinctifn d'une clé publique, d'attributs complémentaires et le tout signé par le demandeur.

PKCS#12 : définit un format de stockage du certificat utilisateur, une clé privée ou une clé publique et éventuellement d'autres certificats. Pour protéger la clé privée, l'accès au fichier est protégé par mot de passe. Fichier au format .p12 ou .pfx

Excellente référence ici : http://glasnost.entrouvert.org/rubrics/45.html