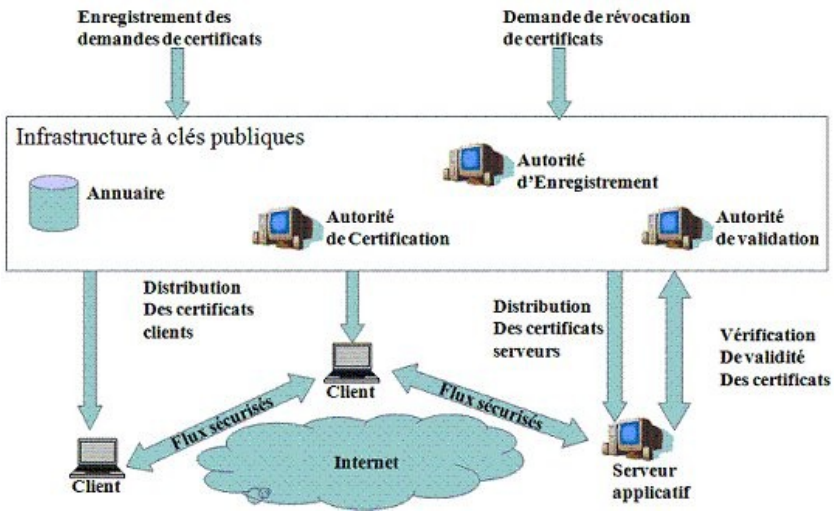
Systèmes Intélligents pour le Transport

Introduction à la sécurité informatique

Version 1



YACINE CHALLAL & HATEM BETTAHAR

15/10/2008

Table des matières

Objectifs

Se sensibiliser aux risques liés aux attaques sur les systèmes d'information.

Se familiariser avec les concepts de la sécurité informatique.

Connaître les différents services de sécurité.

Savoir utiliser des mécanismes cryptographiques pour garantir différents services de sécurité.

Analyser un protocole cryptographique

Introduction

Ce cours introduit les différents services de sécurité et les mécanismes cryptographques permettant de les réaliser. Ainsi, on présentera les notions de confidentialité et chiffrement, contrôle d'intégrité, authentification, signature digitale et non-répudiation. On présentera également l'importante notion de certificats numériques et d'infrastructures à clés publiques.

Des ateliers sur machines permettrons de pratiquer ces différentes notions.

Introduction à la sécurité des échanges

I

Dans cette partie on présentera les différents défis de la sécurité informatique, les différents types d'attaques et leurs motivations, les services de sécurité, et des statistiques sur les pertes engendrées par des attaques sur les systèmes d'information d'entreprise.

A. Réseaux informatiques : Risques et enjeux

Confiance et Internet

Dans la vie courante la plupart des transactions reposent sur une « confiance » acquise par une relation en face à face ou un contact physique . Dans le cybermonde cette relation de proximité est rompue. Comment établir une relation de confiance indispensable à la réalisation de transactions à distance entre personnes qui ne se connaissent pas ? Ce cours a pour but de répondre à cette question.

Fondamental : Part de responsabilité des usagers

Thucydite dit : « Ce ne sont pas les murs qui protègent la citadelle, mais l'esprit de ses habitants ». Ceci s'applique également aux systèmes d'information où les statistiques indiquent que 40% des attaques sont causées par les usagers du SI eux mêmes.

Rappel : Environnement de l'entreprise

Une vingtaine d'années auparavant, les systèmes d'information d'entreprises étaient plutôt centralisés, basés sur des échange papiers, sans accès distants. Aujourd'hui les SI d'entreprises sont plutôt distribués sur plusieurs sites: on retrouve notamment un siège principales et des succursales, des filiales, des télé-travailleurs, des commerciaux, ... L'accès distants devient alors indispensable pour supporter cette décentralisation et la mondialisation des échanges. Ceci devient plus important avec les nouvelles technologies sans fils (Haut débit sur GSM, UMTS, WiMAX, etc.) et la forte pénétration d'Internet dans nos sociétés ;dans quelques années le nombre d'internautes atteindra les 3.000.000.000 de personnes.

Attention : Risques liés aux réseaux

Malgré les bienfaits des réseaux informatiques, ceux-ci présentent d'énormes

risques. Parmi ceux-là on peut citer :

- Interception de messages
  - Prise de connaissance des mots de passe
  - Vol d'information
  - Perte d'intégrité du système et du réseau
- Intrusion des systèmes
  - Vol ou compromission des informations
  - Destruction des informations
  - Virus
  - Détournement de biens
- Perte d'accessibilité au système ou au réseau
- Faux clients, marchands escrocs

Malgré la panoplie de technologies utilisées pour sécuriser les SI des entreprises, les attaques sur les SI existent toujours comme illustré dans la figure suivante tirée d'une enquête du FBI/CSI en 2006.

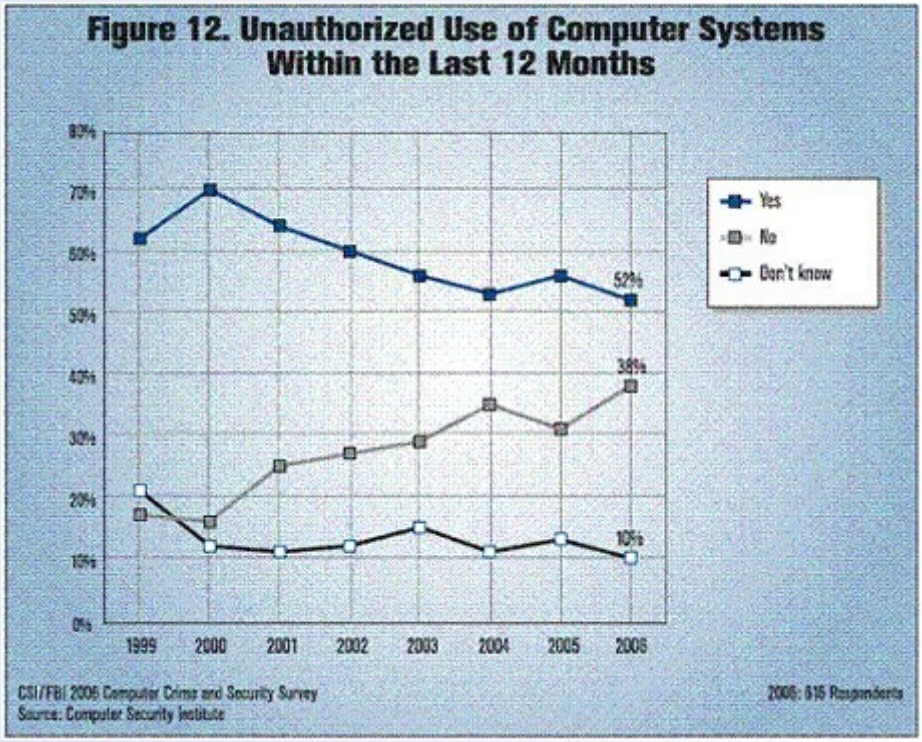


Image 1 : Accès frauduleux aux SI

La figure suivante illustre les différents types d'attaques notées en 2006 :

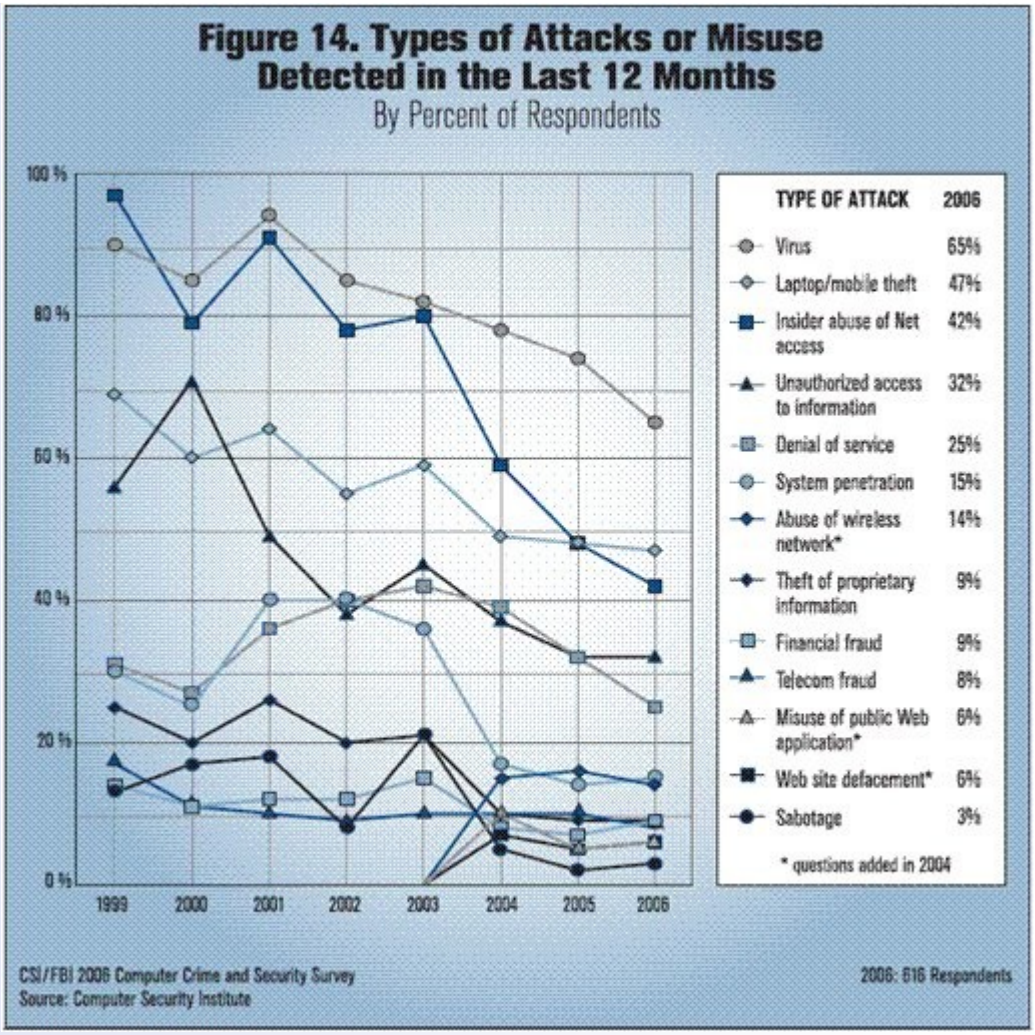


Image 2 : Types d'attaques sur les SI en 2006

Motivations d'un attaquant

Un attaquant n'est pas forcément un "hacker" chevronné. Ca peut être n'importe quelle personne avec des motivations aussi banales que les suivantes :

- Le gain financier
  - Récupération de num de cartes bancaires, ...
- Vengeance
  - Site www.aljazeera.net lors de la couverture de la guerre d'irak
- Besoin de reconnaissance
  - Attaque contre le site du cerist avec un message sur les restrictions d'accès à Internet à Cuba.
- Curiosité
  - Attaques d'étudiants du MIT sur le premier ordinateur IBM 704 au MIT en 1959.
- Recherche d'émotions fortes
- Ignorance
  - Envoie de mots de passes par email, ...

Pertes phénoménales !!!

Les pertes financières dues aux attaques informatiques sont phénoménales. D'après une enquête réalisée par le FBI/CSI :

74% des pertes financières des entreprise sujets de l'enquête sont duent aux :

- attaques de virus (plus de 15 millions de dollars de perte)
- accès non autorisés aux systèmes d'information (plus de 10 millions de dollars de perte)
- vols d'équipement mobile (plus de 6 millions de dollars)
- vols de la propriété intellectuelle (plus de 6 millions de dollars)

52% des organisations sondées ont déclaré avoir été attaqué les 12 derniers mois (2006) :

- 24% d'entre elles ont reporté plus de 6 attaques
- 48% ont reporté 1 à 5 attaques

La figure suivante illustre la répartition des pertes sur leurs causes :

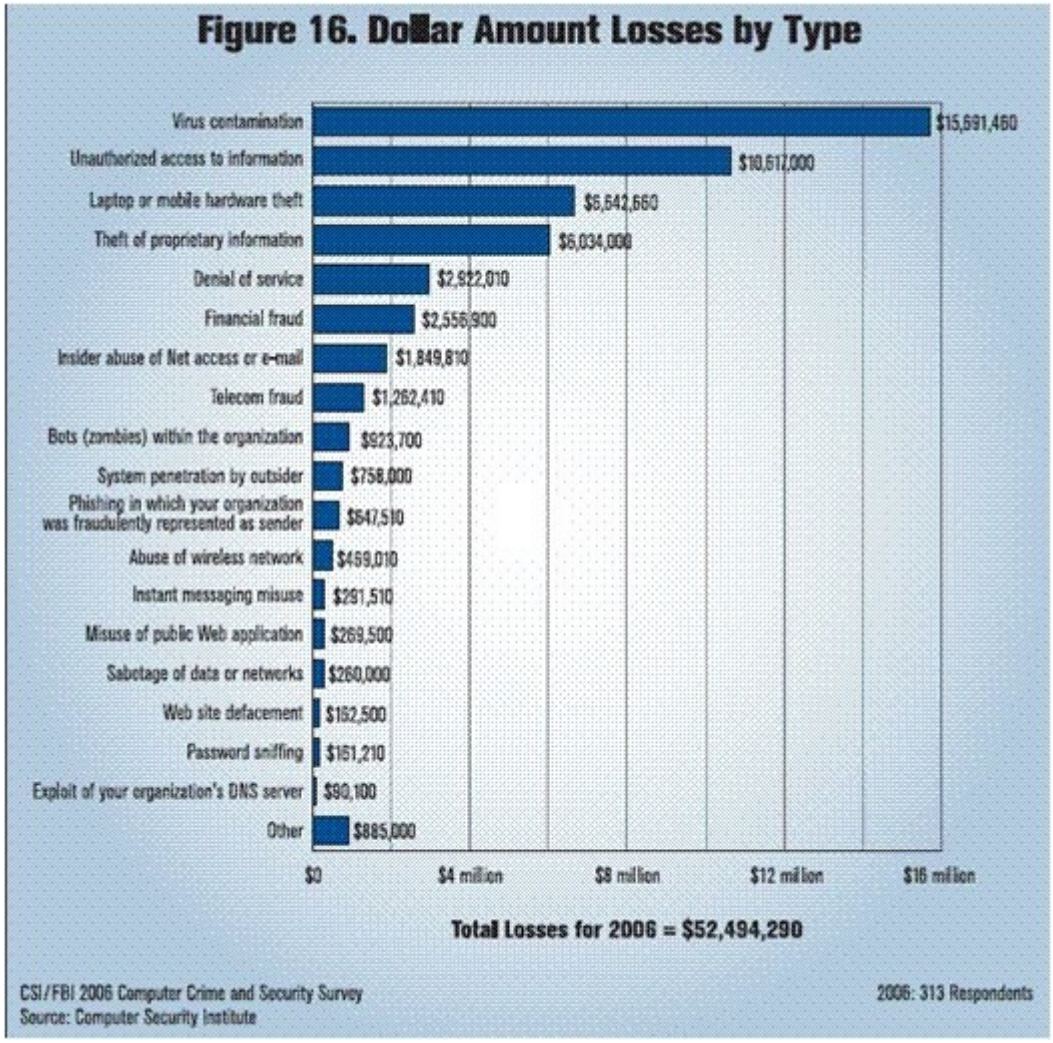


Image 3 : Pertes financères

Complément : Rehaussement des budgets réservés à la sécurité des SI

La conséquence immédiate de ces pertes qui sont majoritairement liées à la sécurité des SI, est l'augmentation importante des budgets alloués à sécuriser les SI et à former le personnel sur la sécurisation des SI et des échanges d'information. Selon la même enquête :

- 34% des organisations allouent pas moins de 5% du budget informatique à la sécurité informatique

  - En 2006, les compagnies de revenues inférieurs à 10 millions de dollars ont dépensé en moyenne 1349 dollars par employé pour la sécurité informatique- un rehaussement de 210% par rapport à l'année 2005
- plus de 80% des institutions conduisent un audit de sécurité informatique
- la majorité des institutions jugent la formation en sécurité informatique comme importante et stratégique
  - 61% de ces organisations refusent de sous-traiter leurs fonctions de sécurité informatique

La figure suivante illustre le pourcentage du budget IT alloué à la sécurité :

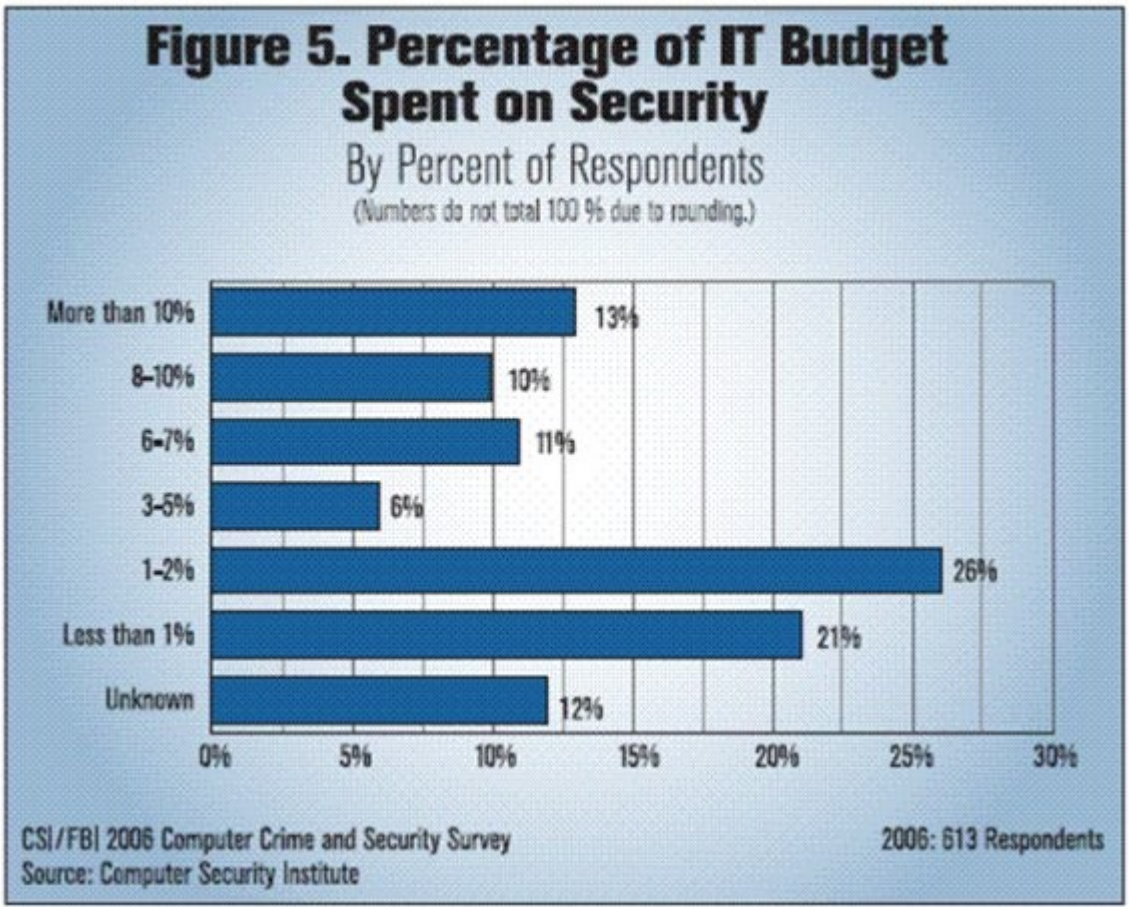


Image 4 : Part de la sécurité dans le budget IT

Remarque : Rapport du FBI/CSI

Le FBI/CSI (Computer Security Institute) publie chaque année un rapport sur la sécurité informatique. Voici le rapport du FBI/CSI pour 2007 :

Exemple : Menaces Informatiques et Pratiques de Sécurité en France

Selon un le rapport 2008 du CLUSIF (Club de la Sécurité de l'Information Français) :

- Plus de 70% des entreprises françaises ont une forte dépendance à l'informatique
- Le budget moyen alloué pour la sécurité du SI dépasse 114K€ dans 21% des cas.
- 28% des entreprises du secteur des services, banques et assurances ont augmenté leur budget sécurité du SI de plus de 10% en 2008

- 53% des Responsables de Sécurité du SI (RSSI) dénoncent le manque de personnel qualifié comme frein majeur à la conduite de leur mission.
- Plus de 30% des entreprises n'ont pas une Politique de Sécurité de l'Information (PSI), et 45% de celles qui en ont ne respectent pas une norme de sécurité.
- Le rattachement de la RSSI à la DG passe de 39% en 2006 à 45% des cas en 2008.

Selon le même rapport du CLUSIF, beaucoup de technologies de contrôle d'accès sont méconnues et/ou non utilisées en entreprises françaises comme illustré dans la figure suivante :

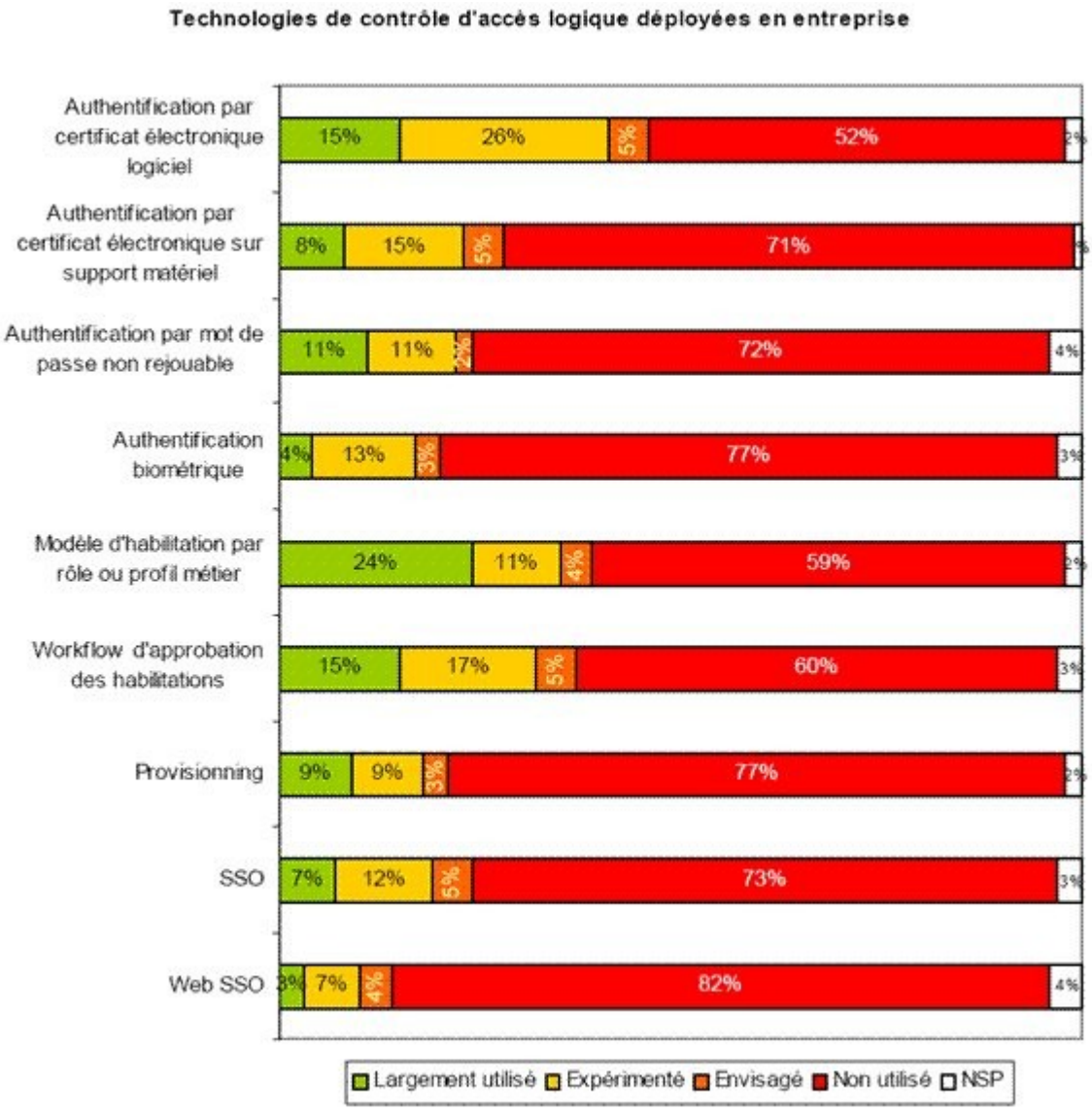


Image 5 : Technologies de contrôle d'accès non utilisées en France

Le rapport note aussi que 56% des RSSI ont noté au moins un incident de sécurité

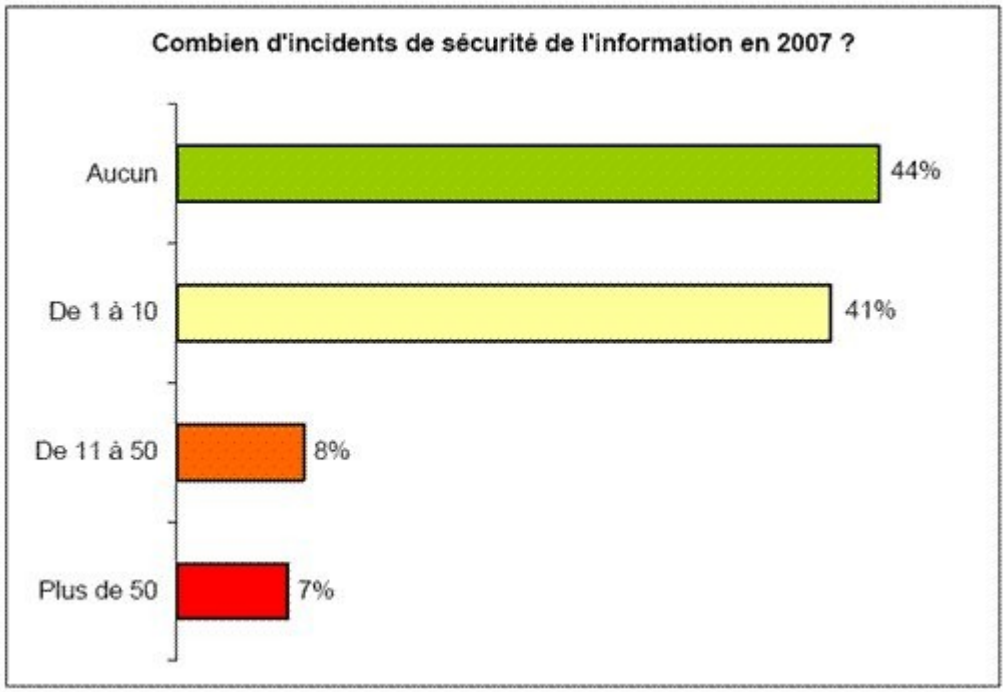


Image 6 : Incidents de sécurité informatique en France

Le rapport du CLUSIF 2008 est très riche et mérite une lecture. Le voici :

B. Définitions des Services de Sécurité

Voici quelques définitions informelles à retenir concernant les services de sécurité les plus importants

Définition : Authentification

Permet de vérifier l'identité revendiquée par une entité, ou l'origine d'un message, ou d'une donnée .

Définition : Confidentialité

Permet de se protéger contre la consultation abusive des données par des entités tierces indésirables

Définition : Contrôle d'intégrité

Permet de vérifier qu'une données n'a pas été modifiée par une entité tierce (accidentellement ou intentionnellement)

Définition : Contrôle d'accès

Permet de vérifier que toute entité n'accède qu'aux services et informations pour lesquelles elle est autorisée

Définition : Non répudiation

Permet de se protéger contre la contestation d'envoi et de réception de données lors d'une communication

Introduction à la cryptographie

II

Cette partie du cours introduira les mécanismes de base de la cryptographie moderne qui permettent de réaliser quatre services de sécurité fondamentaux :

1. La confidentialité
2. L'intégrité des données
3. L'authentification de l'origine de données
4. La non-répudiation de l'origine

Pour chacun de ces services nous rappellerons la définition puis nous introduirons le mécanisme cryptographique permettant de le réaliser.

A. La cryptographie

Définition : La cryptographie

Le mot « Cryptographie » est composé des mots grecques :

- CRYPTO = caché
- GRAPHY = écrire

C'est donc l'art de l'écriture secrète.

C'est une science permettant de préserver la confidentialité des échanges.

Définition : Cryptanalyse

La cryptanalyse est l'art de décrypter des messages chiffrés.

Objectifs

Parmi les objectifs de la cryptographie :

- Garantir la confidentialité
- Vérifier l'intégrité des données

- Gérer l'authentification
- Assurer la non-répudiation

B. La confidentialité

1. Confidentialité et chiffrement

Définition : Confidentialité

La confidentialité est la propriété qui assure que l'information est rendu inintelligible aux individus, entités, et processus non autorisés.

Définition : Chiffrement / déchiffrement

Le chiffrement est une transformation cryptographique qui transforme un message clair en un message inintelligible (dit message chiffré), afin de cacher la signification du message original aux tierces entités non autorisées à l'utiliser ou le lire. Le déchiffrement est l'opération qui permet de restaurer le message original à partir du message chiffré.

Clé de chiffrement

Dans la cryptographie moderne, l'habilité de maintenir un message chiffré secret, repose non pas sur l'algorithme de chiffrement (qui est largement connu), mais sur une information secrète dite CLE qui doit être utilisée avec l'algorithme pour produire le message chiffré.

Selon que la clé utilisée pour le chiffrement et le déchiffrement est la même ou pas, on parle de système cryptographique symétrique ou asymétrique.

Fondamental : Chiffrement symétrique

Dans le chiffrement symétrique, une même clé est partagée entre l'émetteur et le récepteur. Cette clé dite symétrique est utilisée par l'émetteur pour chiffrer le message te par le récepteur pour le déchiffrer en utilisant un algorithme de chiffrement symétrique.

Image 7 : Chiffrement Symétrique

Exemple : Algorithmes de chiffrement symétriques

Il existe deux types d'algorithmes de chiffrement symétrique :

1. Chiffrement par bloc : division du texte clair en blocs fixe, puis chiffrement bloc par bloc
   - DES: IBM, Standard NIST 1976

- 3DES: W. Diffie, M. Hellman, W. Tuchmann 1978.
- IDEA: Xuejia Lai et James Massey en 1992
- Blowfish: Bruce Schneier en 1993
- AES (Rijndael): Joan Daemen et Vincent Rijmen 2000

2. Chiffrement par flux : le bloc a une dimension unitaire (1 bit, 1 octet, ...), ou une taille relativement petite
   - RC4: Ron Rivest 1987
   - SEAL: Don Coppersmith et Phillip Rogaway pour IBM 1993.

Fondamental : Chiffrement asymétrique

Dans un système asymétrique, le récepteur génère une paire de clés asymétrique : une clé publique qui est diffusée à tout le monde et une clé privée maintenue secrète chez le récepteur. La particularité de cette paire de clé est que tout message chiffrée avec la clé publique ne peut être déchiffré qu'avec la clé privée correspondante. D'où la confidentialité des messages chiffré avec la clé publique d'un récepteur. Bien évidemment la clé privée correspondante ne peut être calculée à partir de la clé publique correspondante.

Image 8 : Chiffrement Asymétrique

Exemple : Algorithmes de chiffrement asymétrique

RSA: Rivest, Shamir et Adleman 1978

Diffie et Hellman 1976

2. Introduction à DES

Aperçus générale

DES (Data Encryption Standard) est l'un des algorithmes pionnier de chiffrement symétrique. Il est basé sur un ensemble de permutations et substitutions comme présenté ci-dessous.

C'est un algorithme qui opère sur des blocs de 64 bits, et utilise une clé de 56 bits qui était suffisante à l'époque. Il a été définis officiellement dans FIPS46-3. Il est constitué d'une permutation initiale, un calcul médian en fonction de la clé et une permutation finale.

La permutation initiale

La figure ci-dessous illustre la permutation initiale sur un bloc de 64 bits sur lequel opère DES

Permutation initiale

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Image 9 : Permutation initiale de DES

Algorithme DES

Cette figure résume les 16 itérations de l'algorithme DES.

La fonction f intervenant à chaque phase sera présentée dans le paragraphe suivant.

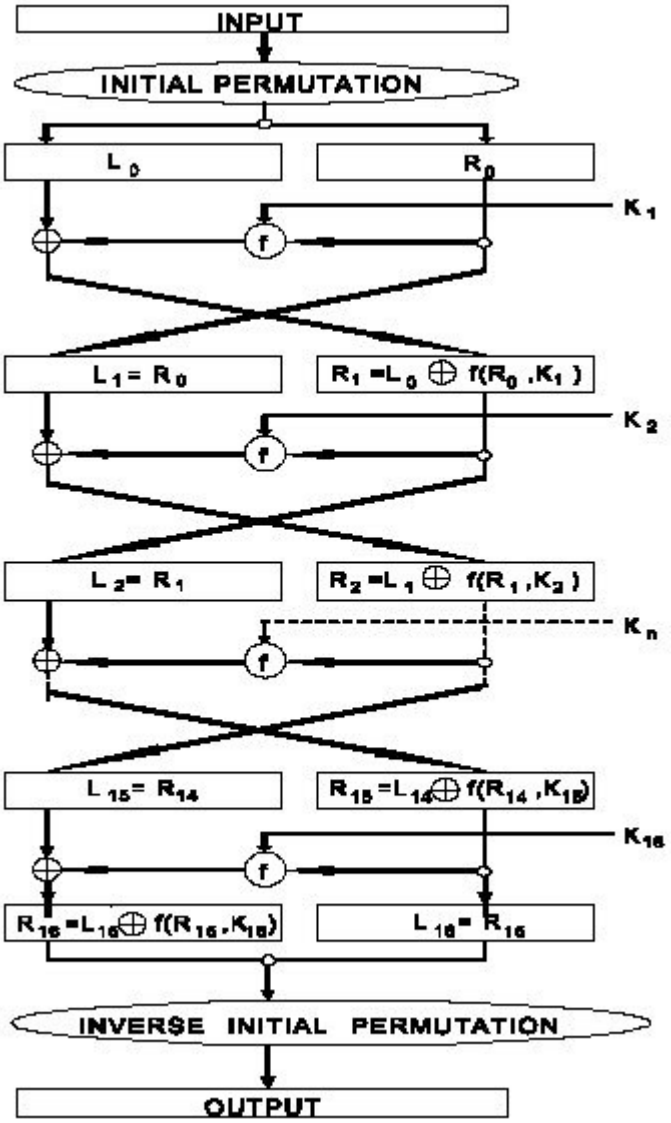
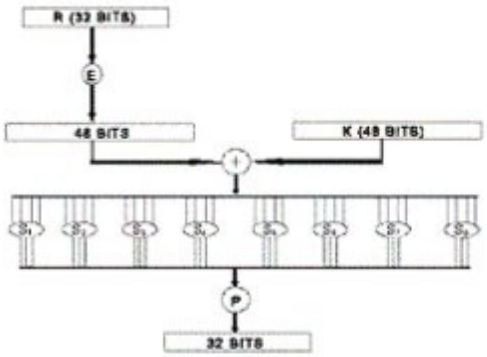


Image 10 : Algorithme DES

La fonction f de DES

La figure suivante illustre la fonction f et ses différentes sous fonctions, ainsi qu'un exemple d'une S-box

f



E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

S_1

Ligne No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Image 11 : La fonction f

Attention : Sécurité de DES

DES fut raisonnablement sûr à l'époque de son invention. RSA Security a lancé le DES Challenge qui a permis de mettre fin à la robustesse de DES à la cryptanalyse :

- DES Challenge I 1997: DESCHALL a cassé la clé DES en 96 j
- DES Challenge II-1 1998: Distributed.net a résussi à casser la clé DES en 41j
- DES Challenge II-2 1998: EFF Deep Crack a cassé la clé DES en 56h
- DES Challenge III 1999: Deep Crack et Distributed.net ont cassé la clé DES en 22h15

En 2000 AES deviens le standard à la place de DES

3. Les modes d'opération du chiffrement symétrique

Dans le chiffrement symétrique l'algorithme opère sur un bloc. Pour chiffrer un ensemble de blocs constituants le message à chiffrer il est nécessaire de définir une stratégie d'opération sur la succession des blocs à chiffrer.

Il existe quatre modes définis dans FIPS 81 (1980)

- Electronic Code Book (ECB),
- Cipher Block Chaining (CBC),
- Cipher FeedBack (CFB) et
- Output FeedBack (OFB).

Syntaxe : Notation

Nous adoptons la notation suivante dans la description des quatre modes d'opération :

- T[n]: n-ième bloc du texte clair

- C[n]: n-ième bloc du texte chiffré
- E(m): fonction de chiffrement
- D(m): fonction de déchiffrement
- IV : Initialization Vector
- ^ : XOR

Electronic Code Book (ECB)

Chiffrement :C[n] = E(T[n])
Déchiffrement :T[n] = D(C[n])
Le même texte clair et clé de chiffrement donnent le même texte chiffré.

CBC : Cipher Block Chaining

Chiffrement :

- C[0] = E(T[0] ^ IV)
- C[n] = E(T[n] ^ C[n-1]) , si (n > 0)

Déchiffrement :

- T[0] = D(C[n]) ^ IV
- T[n] = D(C[n]) ^ C[n-1] , si (n > 0)

IV est envoyé en clair avec le message chiffré

CFB : Cipher Feedback

I[n]: bloc temporaire
Chiffrement :

- I[0] = VI
- I[n] = C[n-1] , si (n > 0)
- C[n] = T[n] ^ E(I[n])

Déchiffrement :

- I[0] = VI
- I[n] = C[n-1], si (n > 0)
- T[n] = C[n] ^ E(I[n])

Offre une sécurité plus élevée

OFB : Output Feedback

I[n]=nième bloc temporaire
R[n]=nième bloc temporaire second
Chiffrement :

- I[0] = VI
- I[n] = R[n-1] , si (n > 0)
- R[n] = E(I[n])
- C[n] = T[n] ^ R[n]

Déchiffrement :

- I[0] = VI
- I[n] = R[n-1] , si (n > 0)
- R[n] = E(I[n])
- T[n] = C[n] ^ R[n]

4. RSA : Rivest Shamir et Adleman 1978

Fondamental : Principe de RSA

RSA esr fondé sur la difficulté de factoriser des grands nombres qui sont le produit de deux grands nombres premiers.

Cryptographiquement parlant, on peut dire que multiplier deux grands nombres premiers est une fonction à sens unique: Il est facile de multiplier deux nombres pour obtenir un produit, mais difficile de factoriser ce produit et de retrouver les deux grands nombres premiers.

Algorithme RSA

Initialisation

- Choisir deux nombres premiers, p et q, les deux étant plus grands que 10^{100}.
- Calculer n = p · q (n est le modulus)
- Choisir e aléatoire tel que e et ((p - 1) · (q - 1)) n'aient aucun facteur commun excepté 1
- Trouver d tel que : ed = 1 mod((p - 1)(q - 1)).
- Clé publique : (n,e).
- Clé privée : (n,d) ou (p,q,d) si on désire garder p et q.

Chiffrement/Déchiffrement

- L'expéditeur crée le texte chiffré c à partir du message m : $c = m^e \bmod(n)$, où (n,e) est la clé publique du destinataire
- Le destinataire reçoit c et effectue le déchiffrement : $m = c^d \bmod(n)$, où (n,d) est la clé privée du destinataire.

Attention : Sécurité de RSA

Ce qui est connus est la clé publique (e,n)

Pour déchiffrer un message m, il faut connaître d tel que ed=1 mod(p-1)(q-1)

Pour calculer d il faut donc connaître p et q

Or on sait que n=pq et on connaît n

Il faut donc factoriser n en ses facteurs premiers p et q

Or personne n'a pus le faire en un temps raisonnable.

5. Échange de clé Diffie-Hellman

Objectif de l'algorithme

Deux entités voudraient se mettre d'accord sur un secret (en échangenat des messages publiques) afin de s'échanger des messages confidentiels

Fondamental

L'algorithme de Diffie Hellman a été fondé sur la difficulté du calcul du logarithme discret

Algorithme Diffie-Hellman

La figure suivante illustre les différentes étapes à suivre pour se mettre d'accord sur un secret commun en échangeant des messages publiques :

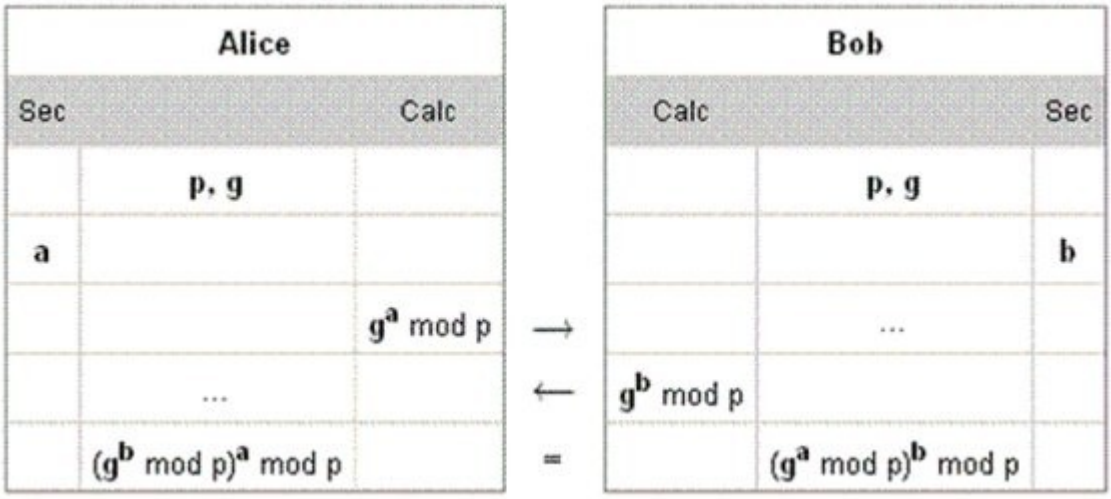


Image 12 : Algorithme Diffie-Hellman

Attention : Sécurité de Diffie-Hellman

Au final, Alice et Bob partagent le secret g^{ab} mod p, mais une tierce partie ayant intercepté les messages échanger entre Alice et Bob ne pourra calculer ce secret. En effet, cette tierce partie connaît g^a mod p, et g^b mod p, mais pour calculer g^{ab} mod p, il faut calculer a à partir de g^a mod p, ou b à partir de g^b mod p. Or personne ne sait comment calculer le logarithme discret.

C. Intégrité de données

Définition : Intégrité de donnée

C'est la propriété qui permet de vérifier qu'une données n'a pas été modifiée par une entité tierce (accidentellement ou intentionnellement).

Une fonction de hachage est typiquement utilisée pour vérifier l'intégrité de données.

Définition : Fonction de hashage cryptographique

Une fonction de hashage associe à une chaîne binaire (de longueur variable) une chaîne de longueur fixe [Menezes et al.]

Une fonction de hashage cryptographique a les propriétés suivantes :

- Étant donné m, il est facile de calculer h(m)
- Étant donné h, il est difficile de calculer m tel que h(m)=h
- Étant donné m, il est difficile de trouver un autre message, m', tel que h(m)=h(m').

Méthode : Comment utiliser une fonction de hashage pour contrôler l'intégrité de données.

La figure ci-contre illustre comment utiliser une fonction de hashage pour vérifier l'intégrité d'un document numérique.

Initialement le code de hashage du document numérique est calculé et stocké dans un endroit sûre. Ultérieurement ce code est recalculé et comparé à celui qui a été stocké.

Si les deux valeurs sont égales alors le document n'a pas été modifié. Sinon, le document a subit une modification.

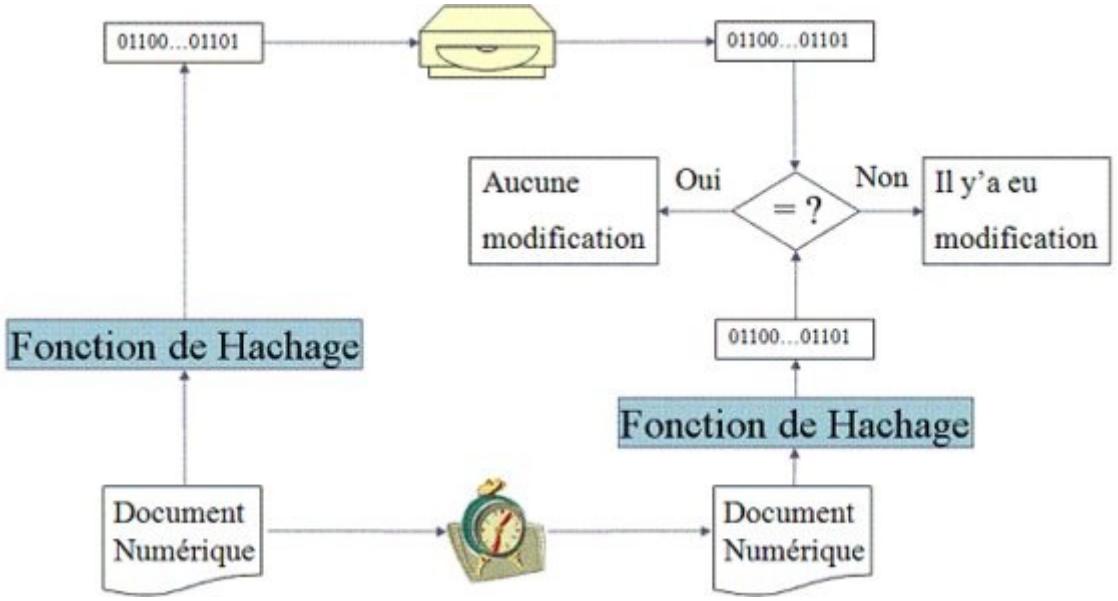


Image 13 : Intégrité de données

Exemple

Il existe plusieurs fonctions de hashages ; En voici quelques unes :

- MD2 (Message Digest 2) : Opère sur des bloques de 16 octets, manipule des mots de 8 bits Output 128 bits.
- MD4 (Message Digest 4) : Manipule des mots de 32 bits, plus performant sur des processeurs 32 bits.
- MD5 (Message Digest 5) : Une passe de plus / MD4, plus sûre
- SHA-1 (Secure Hash Algorithm) : Proposé par le NIST Input message 2^{64} octets (au max), output 160 bits.

D. Authentification de l'origine de données

Définition : Authentification de l'origine

C'est la propriété qui permet de vérifier que la source de données est bien l'identité prétendue.

Définition : Message Authentication Code (MAC)

C'est est un mécanisme cryptographique qui permet de vérifier l'authenticité de l'origine des données et leur intégrité en même temps.

Un MAC est une famille de fonctions h_k paramétrée par une clé secrète k avec les propriétés suivantes :

- Étant donné une clé k et un message m, $h_k(m)$ est facile à calculer,
- Étant donné zéro ou plusieurs paires $(m_j,h_k(m_j))$, il est rès difficile de calculer n'importe quelle paire $(m,h_k(m))$ pour n'importe quel message m.

Méthode : Comment utiliser un MAC pour garantir l'authentification d el'origine

Pour garantir d'authenticité de l'origine, l'émetteur et le récepteur doivent partager une clé symétrique.

Cette clé sera utilisé par l'émetteur pour calculer un MAC sur le message à envoyer. Ce MAC (code de hashage) est la preuve d'authenticité qui accompagnera le message.

Le récepteur utilisera la même clé secrète pour calculer le MAC de nouveau sur le message reçus. Le MAC nouvellement calculé sera comparé au MAC accompagnant le message. Si les deux valeurs sont égales alors le message et l'origine sont authentiques. Sinon, soit le message ou l'origine n'est pas authentique.

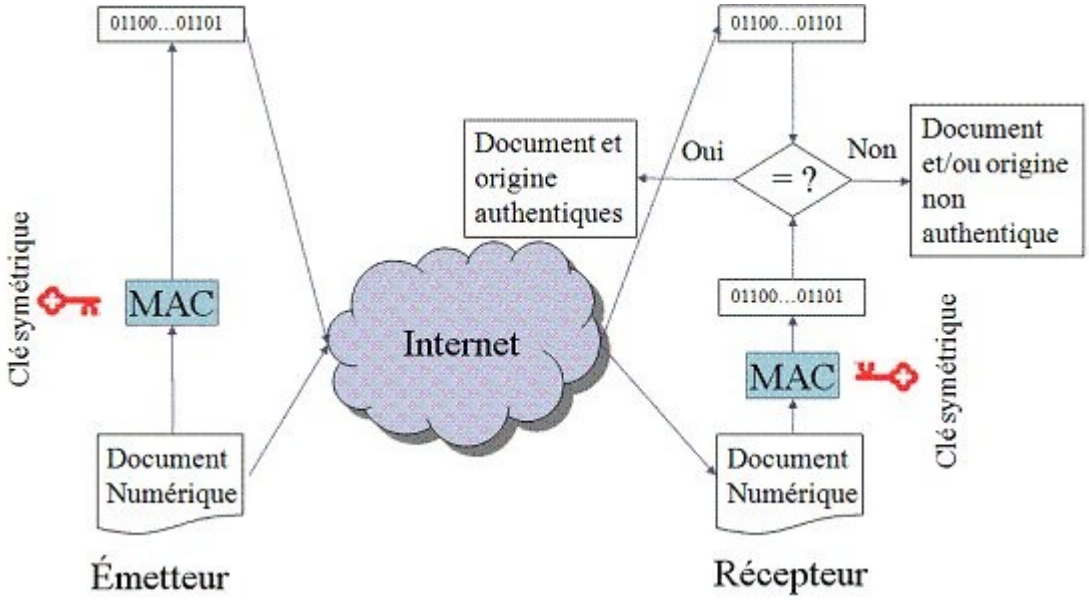


Image 14 : Authentification de l'origine

Exemple : Exemple de MAC

HMAC : Mihir Bellare, Ran Canetti, et Hugo Krawczyk 1996 FIPS PUB 198, RFC 2104

- HMAC-MD5
- HMAC-SHA-1

$$HMAC_K(m) = h\Big((K \oplus opad) \,||\, h\big((K \oplus ipad) \,||\, m\big)\Big)$$

opad= 0x5c5c5c...5c5c
Ipad= 0x363636...3636

Image 15 : Exemple de MAC : HMAC

E. Non-répudiation de l'origine

Définition : Non-répudiation de l'origine

La non répudiation de l'origine assure que l'émetteur du message ne pourra pas nier avoir émis le message dans le futur.

La signature digitale

La signature digitale est un mécanisme cryptographique qui permet d'assurer la non répudiation de l'origine.

Ce mécanisme repose sur un système cryptographique asymétrique

La signature est calculée en utilisant la clé privé de l'émetteur

La signature est vérifiée en utilisant la clé publique de l'émetteur

Méthode : Comment utiliser la signature digitale pour assurer la non-répudiation de l'origine ?

L'émetteur du message génère sa paire de clés (publique, privée). Il diffuse sa clé publique te maintient sa clé privée secrète. Pour signer un document l'émetteur commence par calculer le code hashage du document puis signe ce code de hashage avec sa clé privée. Le résultat de cette dernière opération (chiffrement avec clé privée dans le cas de RSA) est la signature digitale qui accompagnera le document. Quand le récepteur reçoit le message et la signature digitale, il recalcule le code de hashage, déchiffre la signature avec la clé publique de l'émetteur et compare les deux codes de hshages. Si les deux codes sont similaires alors la signature est valide.

L'émetteur ne pourra pas nier dans le futur avoir émis le message puisque y a que lui qui peut générer la signature digitale avec sa clé privée secrète.

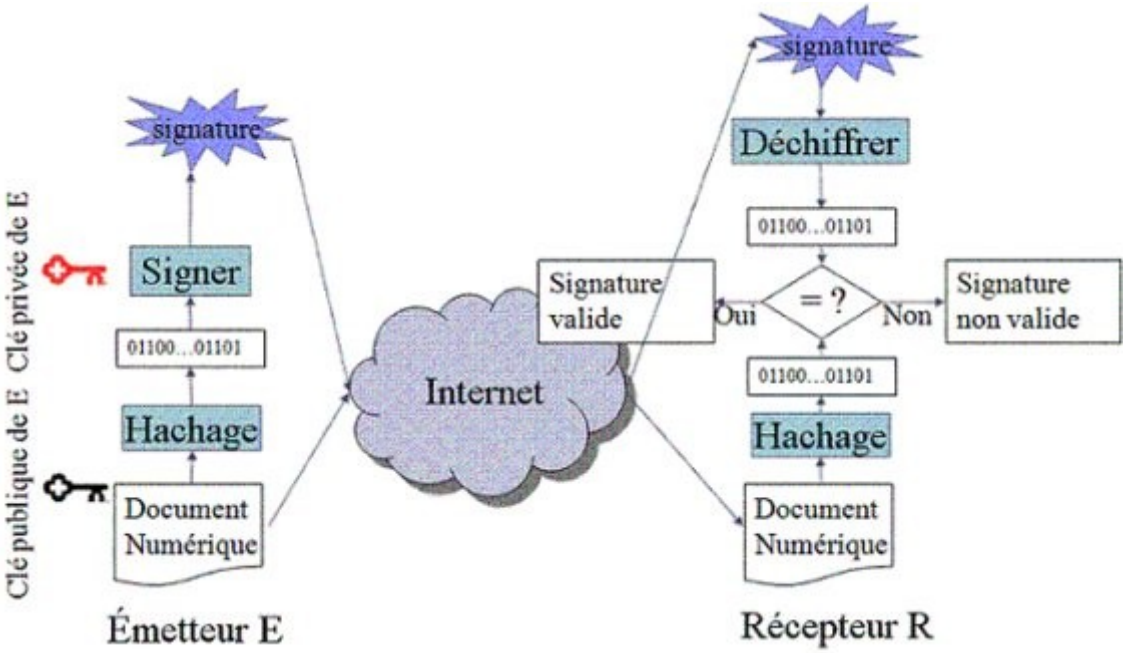


Image 16 : Signature digitale et non-répudiation

Exemple : Signature digitale avec RSA

Initialisation

- Choisir deux nombres premiers, p et q, les deux étant plus grands que 10^{100}
- Calculer n = p · q (n est le modulus)
- Choisir e aléatoire tel que e et ((p - 1) · (q - 1)) n'aient aucun facteur

commun excepté 1
- Trouver d tel que : ed = 1 mod((p - 1)(q - 1)).
- Clé publique : (n,e).
- Clé privée : (n,d) ou (p,q,d) si on désire garder p et q.

Signature digitale
- L'expéditeur crée la signature s à partir du message m : $s = m^d \bmod(n)$, où (n,d) est la clé privée de l'expéditeur.
- Le destinataire reçoit s et m et effectue la vérification de m : $m = s^e \bmod(n)$, où (n,e) est la clé publique de l'expéditeur.

RSA est connu pour être très lent, ayant une clé très longue.

F. Historiques

Quelques fais marquants de l'histoire de la cryptographie

- 50 av. JC. : Julius Cesar utilise une simple substitution de l'alphabet pour les communications gouvernementales
- 1918 : Gilbert Vernam, mathématicien américain, inventa le one-time pad, l'algorithme de chiffrement le plus sûr jusqu'à aujoud'hui, mais impratiquable
- 1923 : Dr. Albert Scherbius, hollondais résidant en Allemagne, met au point la machine Enigma qui sert à encoder des messages. Le prix très cher en fait un échec.
- 1925 : La marine de guerre allemande reprend le projet Enigma en le confiant au Chiffrierstelle, le service de chiffrement
- 1937 : Enigma M3 est adoptée par le Wehrmacht, l'armée allemande
- 1939 : début de la seconde guerre mondiale, où des milliers de scientifiques britanniques, polonais et français travaillaient pour solutionner Enigma, et les milliers de messages chiffrés. L'équipe de Alan Turing trouva la solution
- 1976 : IBM publie un algorithme de chiffrement basé sur Lucifer. Il devient le DES (Data Encryption Standard)
- 1976 : Whitfield Diffie et Martin Hellman introduisent l'idée d'un système à clé publique
- 1978 : l'algorithme de chiffrement à clé publique RSA est publié par Rivest, Shamir et Adleman
- 1978 : Le RC4 est développé par Ronald Rivest pour la RSA Security et sera gardé secret jusqu'en
- 1994, où l'algorithme est rendu public anonymement dans une liste de distribution de Cypherpunks
- 1991 : Phil Zimmermann rend disponible sa première versio de PGP
- 1992 : IDEA est inventé en Suisse par Xuejia Lai et James Massey
- 1992 : MD5 est développé par Ronald L. Rivest
- 1994 : Ron Rivest, déjà auteur de RC2 et RC4, publie RC5
- 2000 : Rijndael devient l'AES, le standard du chiffrement avancé

G. La libraire OpenSSL

Le projet OpenSSL

OpenSSL (http://www.openssl.org) est une libraire qui compte 60.000 lignes de code (langage C). Elle est utilisées par de nombreuses applications; openssh, apache+mod_ssl,... Elle est fondé sur la bibliothèque cryptographique SSLeay d'Eric Toung et Tim Hudson. L'objectif initial de cette librairie était la mise en œuvre des protocoles SSL et servir comme bibliothèque cryptographique.

La bibliothèque OpenSSL

La librairie OpenSSL en décline en deux formes :

1. Interface de programmation en C
   - Bibliothèque SSL/TLS (libssl.a)
     - Mise en œuvre des protocoles SSLv2, SSLv3, TLSv1
   - Bibliothèque cryptographique
     - Cryptographie clé publique et certificats X509: RSA, DSA, DH
     - Chiffrement: DES, 3DES, Blowfish, RC2, IDEA, RC4, + modes ECB, CBC,CFB,OFb pour les algorithmes par blocs
     - Hachage: MD2, MD4, MD5, SHA1, MDC2, RIPEMD160
2. Suite d'applications en ligne de commande openssl(1)
   - Un ensemble de commandes permettant de réaliser les différentes opérations cryptographiques

III

PKI : Infrastructures à clés publiques

A. Systèmes asymétriques : atouts et limites

La gestion des clés est plus facile avec les systèmes asymétriques

Dans ce scénario, si le réseau est composé de n noeud alors il faudra gérer n.(n-1)/2 clé, ce qui ne s'adapte pas au facteur d'échelle. Avec 500 noeud, on arrive déjà à plus de 12 millions de clés à gérer.

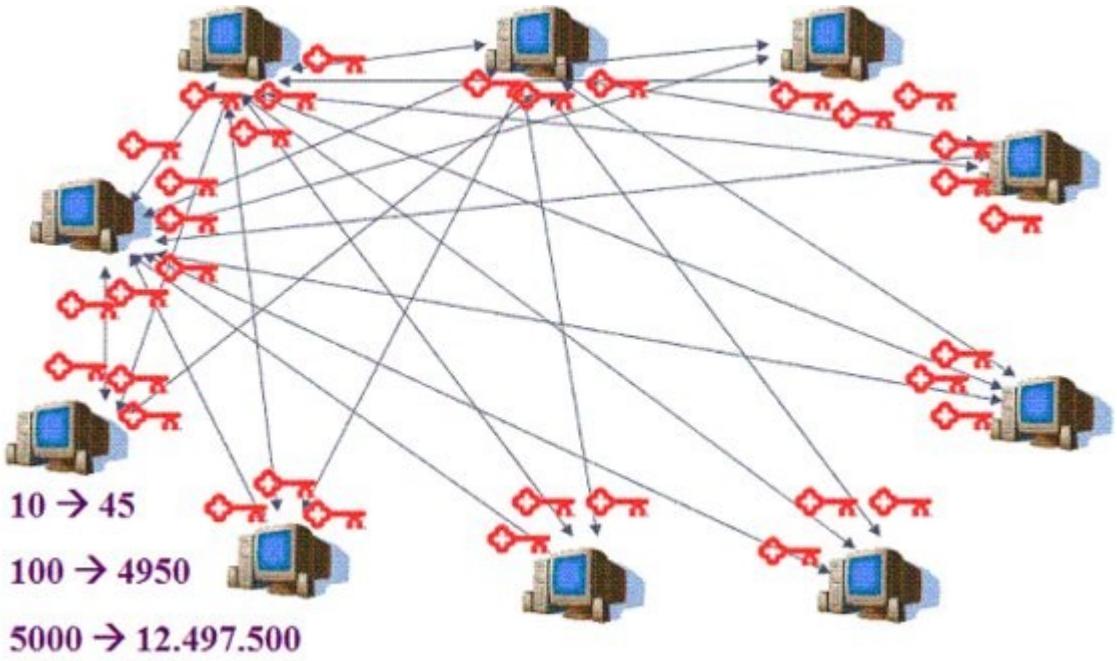


Image 17 : Limites de la gestion de clés symétriques

Par contre, avec un système asymétrique chaque utilisateur aura besoin d'une paire de clés. Donc on aura à gérer seulement 2.n clés au lieu des n.(n-1)/2 clés dans le cas symétrique.

Utilité d'un système asymétrique dans l'authentification

Nous avions déjà vus que l'usage d'un système asymétrique est indispensable pour garantir la non-répudiation de l'origine.

L'usage d'un système asymétrique est également utile pour l'identification comme expliqué plus bas.

Dans ce scénrio, Alice veut s'assurer de l'identité de Bob. Elle ne connaît de Bob que sa clé publique.

Pour s'assurer de l'identité de Bob, Alice lui envoie un défi (un nombre aléatoire).

Pour prouver son identité à Alice, Bob signe le défi avec sa clé privée et envoie sa signature sur le défi à Alice.

Pour vérifier l'identité de Bob, il suffit que Alice vérifie la signature de Bob avec sa clé publique comme illustré sur la figure ci-contre.

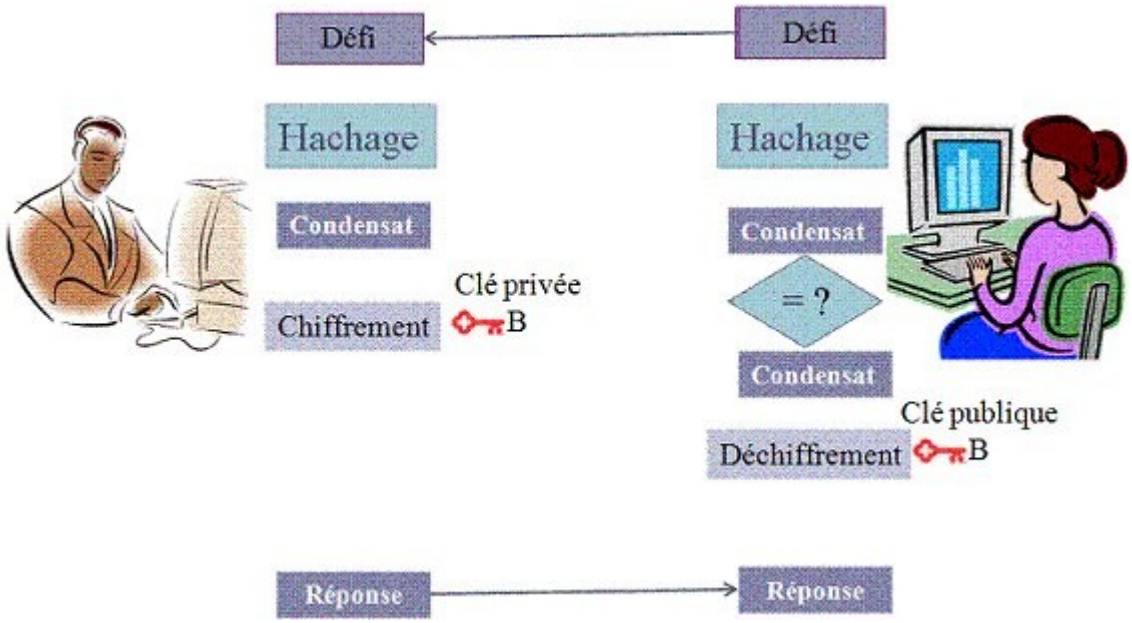


Image 18 : Identification avec un système asymétrique

Attention : Comment garantir qu'une clé publique correspond bien à l'entité avec qui on communique ?

Jusque là, nous avions toujours supposé que la clé publique est distribuée d'une manière sécurisée. Si cette hypothèse n'est pas vérifiée, un schéma asymétrique peut subir une attaque de type "Man in the Middle". Une telle attaque est illustrée dans le scénario ci-après.

Image 19 : Man in the middle

Méthode : Certification numérique

La solution au problème dit "man in the middle" est l'usage d'un certificat numérique qui assure la liaison entre l'identité et la clé publique correspondante dans un document numérique signé par une tierce partie de confiance dite autorité de certification.

B. La certification numérique

Définition : Certificat numérique

- Un certificat à clé publique est un certificat numérique qui lie l'identité d'un système à une clé publique, et éventuellement à d'autres informations;
- C'est une structure de donnée signée numériquement qui atteste sur l'identité du possesseur de la clé privée correspondante à une clé publique.
- Un certificat est signé numériquement par une autorité de certification à qui font confiance tous les usagers et dont la clé publique est connue par tous d'une manière sécurisée. Ainsi, afin de publier sa clé publique, son possesseur doit fournir un certificat de sa clé publique signé par l'autorité de certification. Après vérification de la signature apposée sur le certificat en utilisant la clé publique de l'autorité de certification, le récepteur peut déchiffrer et vérifier les signatures de son interlocuteur dont l'identité et la clé publique sont inclus dans le certificat.

Exemple : Structure d'un certificat X.509

- Version
- Numéro de série
- Algorithme de signature du certificat
- Signataire du certificat
- Validité (dates limite)
  - Pas avant
  - Pas après
- Détenteur du certificat
- Informations sur la clé publique

  - Algorithme de la clé publique
  - Clé publique
- Identifiant unique du signataire (Facultatif)
- Identifiant unique du détenteur du certificat (Facultatif)
- Extensions (Facultatif)
  - Liste des extensions...

La figure suivante illustre un tel certificat inclus dans un navigateur web

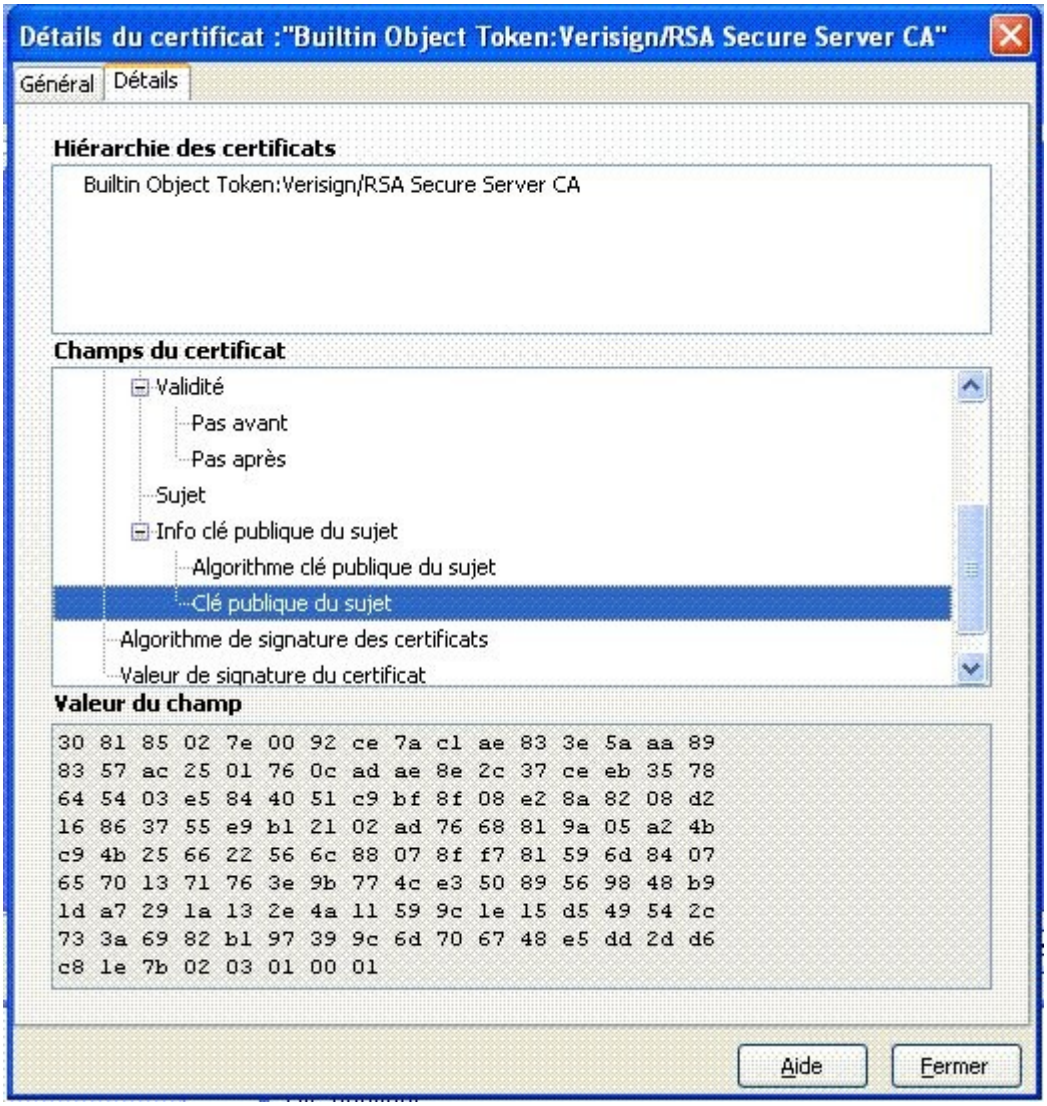


Image 20 : Certificat Numérique

Définition : Autorité de certification

Une autorité de certification est toute entité qui délivre des certificats de clé publique

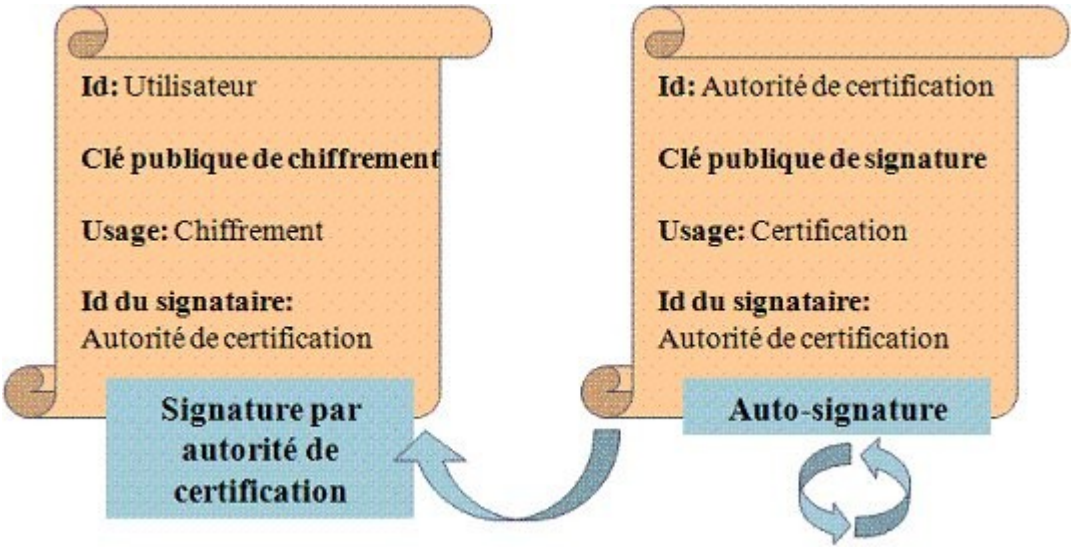


Image 21 : Autorité de certification

Remarque : Auto signature

Une autorité de certification auto-signe son certificat numérique. ceci ne posant pas de problème puisque la clé publique d'une autorité de certification est censée connue d'une manière sécurisée (remise en main propre pas exemple).

Autorité de certification et confiance

L'autorité de certification certifie la correspondance Clé publique – Identité pour l'ensemble d'une population. Ceci mène à faire régner la confiance par transitivité :

- A fait confiance à l'Autorité de Certification
- L'Autorité de Certification délivre un certificat à B
- A est assuré de l'identité de B

C. PKI : Infrastructure à clés publiques

Définition : Infrastructure à clés publiques

« Ensemble de composants, fonctions et procédures dédié à la gestion de clés et de certificats utilisés par des services de sécurité basés sur la cryptographie à clé publique ». [Politique de certification type: Ministère de l'Economie, des Finances et de l'Industrie, Fr]

Méthode : Cycle de vie d'un certificat

La figure suivante illustre le cycle de vie d'un certificat de sa délivrance et à sa destruction

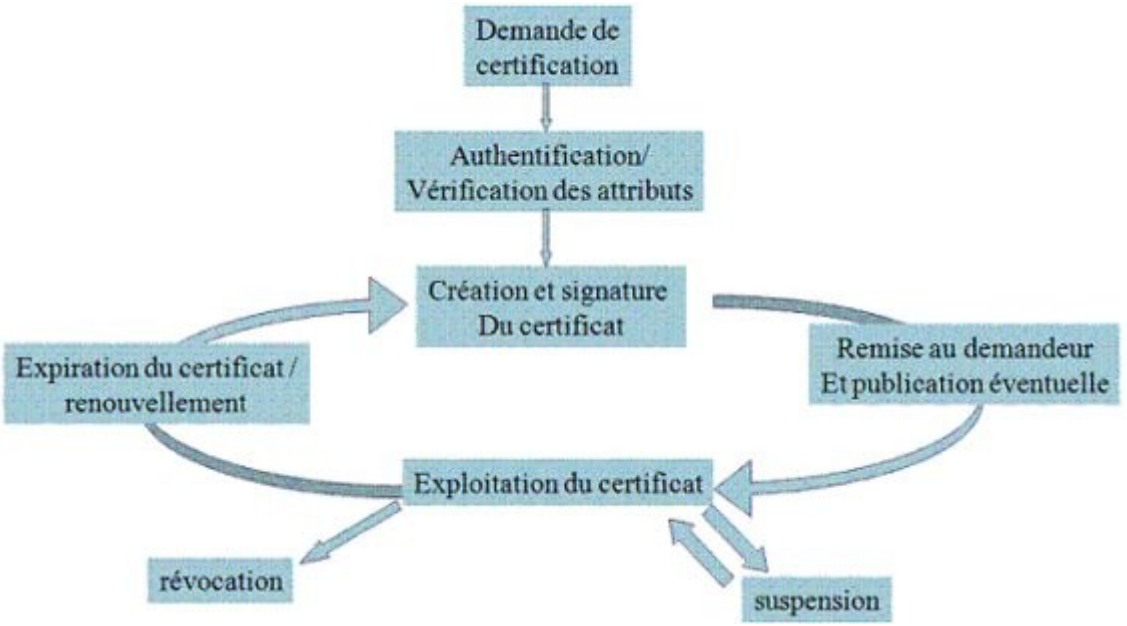


Image 22 : Cycle de vie d'un certificat numérique

Fonctions d'une PKI

Enregistrer et vérifier les demandes de certificats

- Autorité d'enregistrement

Créer et distribuer des certificats

- Autorité de certification

Vérification de validité de certificats

- Autorité de validation

Gérer à tout moment l'état des certificats et prendre en compte leur révocation

- Dépôt de listes de certificats révoqués – CRL (Certificate Revocation List)

Publier les certificats dans un dépôt

- Dépôt de certificats (Annuaire)

Modèles de confiance dans les PKI

Modèle monopoliste

- Une CA pour tout le monde

Modèle monopoliste avec Autorités d'enregistrement

- Une CA avec plusieurs RAs pour la vérification des identités, ...

Délégation de pouvoir de certification

- Une CA délègue le pouvoir de certification à d'autres entités qui deviennent CA à leur tour, en leur fournissant un certificat qui certifie leur capacité d'être CA.

Modèle oligarchique

- Déploiement des produits (comme les navigateur web) avec plusieurs entités de confiance qui sont des CA. Le navigateur fera confiance à tout certificat signé par l'une de ces CA dans sa liste

Modèle anarchique

- Chaque utilisateur établit la liste des entités à qui il fait confiance

Validation de certificat

Pour pouvoir se fier au contenu d'un certificat, il est nécessaire de réaliser les vérifications suivantes:

Vérification	Commentaire
Signature de l'AC	L'application doit vérifier que le certificat est intègre et authentique
Chemin de certification	L'application doit vérifier qu'il existe une chaîne de certificats valide permettant de remonter à une AC de confiance
Période de validité	L'application doit vérifier que le certificat présenté n'est pas expiré
Statut du certificat	L'application doit vérifier que le certificat n'est pas révoqué (ni suspendu)

Image 23 : Validation d'un certificat numérique

Il existe par ailleurs différents moyens et techniques standards pour offrir ce service

- Vérification du statut du certificat par récupération régulière de CRL
- Vérification du statut du certificat en ligne : OCSP (On-line Certificate Status Protocol)
- Vérification complète du certificat en ligne : SCVP (Simple Certificate Validation Protocol)

Image 24 : Protocoles de vérification de certificats

Attention : Révocation de certificats

Un certificat peut être révoqué. La révocation intervient quand la fin de validité réelle précède la fin de validité prévue. La révocation peut avoir plusieurs motifs :

- Compromission réelle ou suspectée de la clé privée
- Modification d'un au moins de attributs certifiés
- Perte de la clé privée (effacement d'un disque dur, perte ou détérioration d'une carte à puce, oubli du code PIN, ...)

- Évolution de l'état de l'art cryptographique (la cryptanalyse de la clé privée entre dans le domaine du possible)
- Perte de confiance vis-à-vis d'un acteur ou d'un composant de la PKI

Le demandeur doit être habilité et authentifié

- Le propriétaire du certificat
- Son supérieur hiérarchique
- Le service de gestion du personnel ...

La méthode de révocation dépend de la méthode de validation

- Utilisation d'annuaire « positif » ==> La révocation consiste à enlever le certificat révoqué de l'annuaire
- Utilisation d'un annuaire « négatif » ou CRL ==> La révocation consiste à inscrire le certificat dans une liste de révocation de certificat

Remarque : La gestion des CRL

La gestion des CRL peut devenir complexe et lourde :

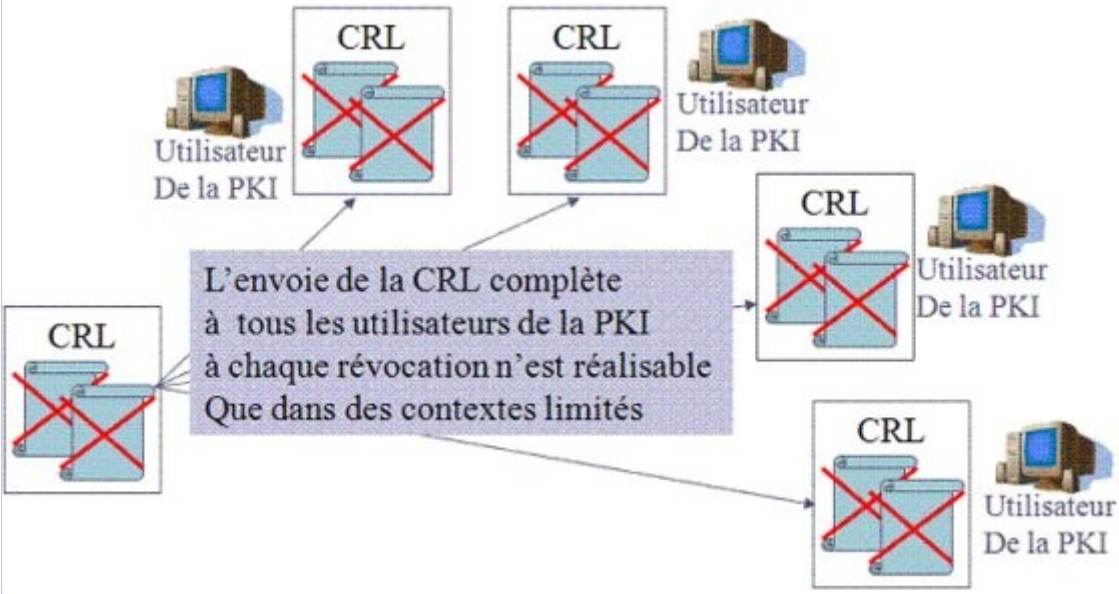


Image 25 : Gestion des CRL

Les delta CRL ne contiennent que les changements depuis la dernière diffusion :

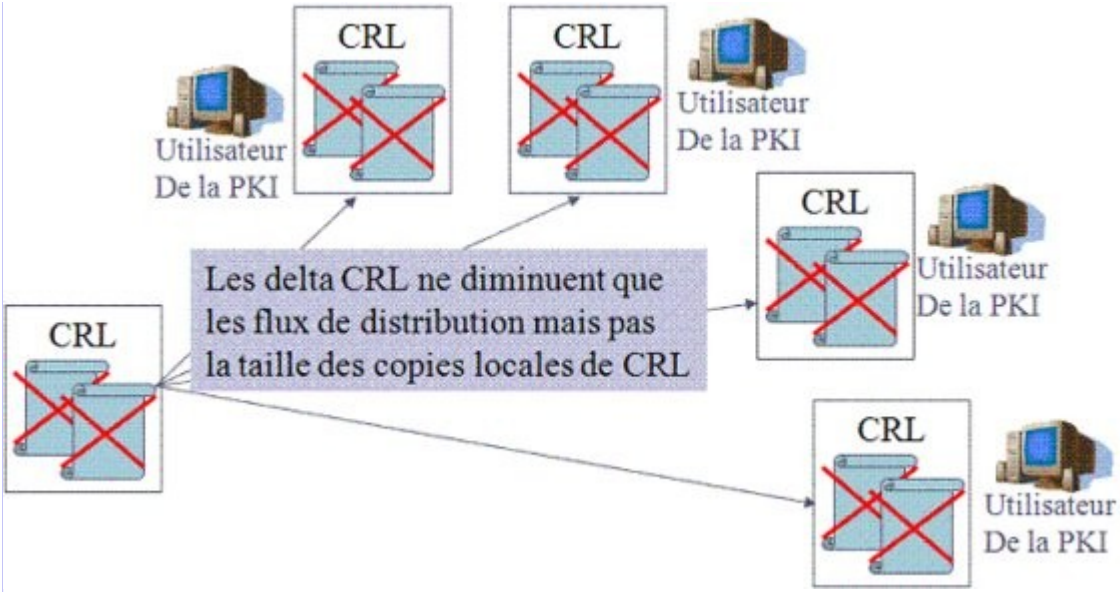


Image 26 : Gestion des delta CRL

Fondamental : Schéma fonctionnel simplifié d'une PKI

En résumé, voici les différents composants d'une PKI :

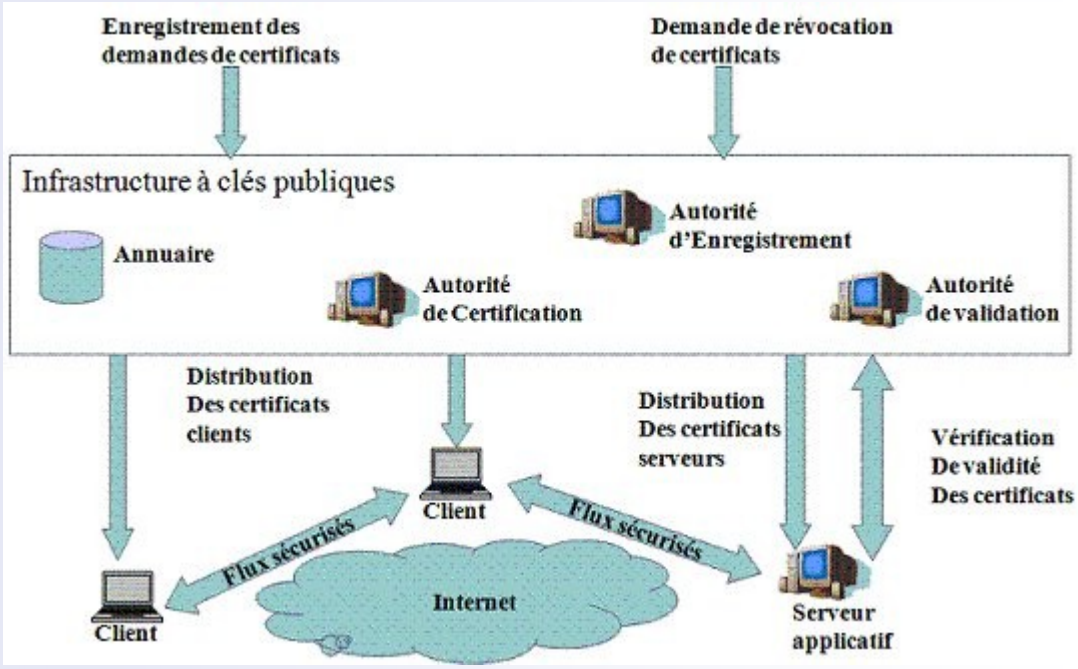


Image 27 : Schéma fonctionnel d'une PKI

D. Secure Socket Layer : SSL

Aperçus générale

SSL/TLS est un protocole de sécurisation des échanges développé par Netscape. Il assure les transactions Client / Serveur sur Internet. Il a été intégré dans les navigateurs web depuis 1994. La vesrion 3.1 est baptisée Transport Layer Security TLS. Cette version a été standardisée à l'IETF: RFC 2246. Le protocole fonctionne

au dessus de la couche TCP

Services de sécurité assurés par SSL

Confidentialité

- Obtenue par chiffrement symétrique

Intégrité

- En utilisant des MAC : MD5(128 bits), SHA1(160 bits)

Authentification

- Identification des deux entités (client optionnel) basée sur les certificats X. 509
- Authentification de l'origine des données basée sur des MAC

Sous protocoles de SSL

SSL se déroule selon quatre sous protocoles

1. Handshake
   - Authentification mutuelle
   - Négociation des algorithmes de chiffrement et de hachage
   - Échange des clés symétriques
2. Change Cipher Spec
   - Indiques la mise en place des algorithmes de chiffrement négocié
3. Record
   - Garantir la confidentialité à l'aide du chiffrement, et L'authentification à l'aide de condensât
4. Alert
   - Émission de messages d'alertes suites aux erreurs que peuvent s'envoyer le client et le serveur

Déroulement du protocole SSL

SSL se déroule en deux phases

1. Phase 1: authentification du serveur
   - Requête client
   - Le serveur envoie son certificat et une liste d'algo de crypto à négocier
   - Le client vérifie le certificat du serveur à l'aide de la clé publique du CA contenu dans le navigateur
   - Le client génère un pré-master secret (PMS)(48 octets) qui sera utilisé pour générer le master-key (48 octets).
   - PMS est chiffré avec la clé publique du serveur
   - Les données échangées entre le client et le serveur seront chiffrées et authentifiées avec des clés dérivées du master-secret
2. Phase 2: authentification du client
   - Le serveur peut demander au client de s'authentifier en lui demandant son certificat
   - Le client répond en envoyant son certificat puis en signant un message avec sa clé privé (contient des info sur la session et le contenu des messages précédents)

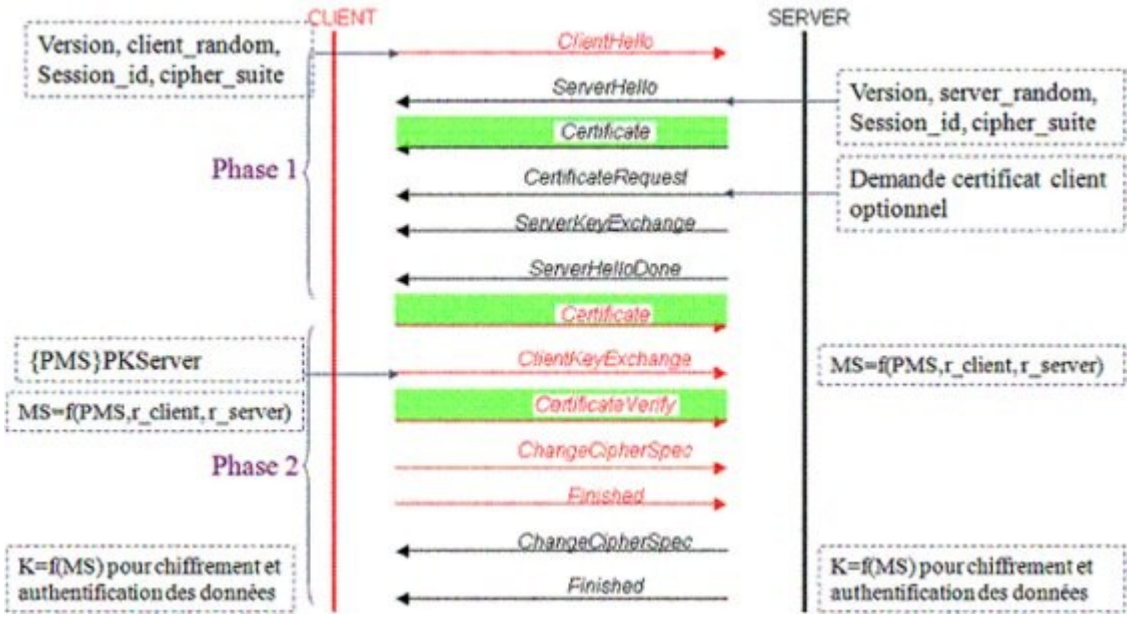


Image 28 : Le protocole SSL/TLS

Testez vos connaissances

IV

A. Chiffrement symétrique

Le chiffrement symétrique

- ☐ Garantie la confidentialité du message chiffré
- ☐ Utilise une paire de clés publique/privée
- ☐ Assure la non répudiation
- ☐ Repose sur la confidentialité de la clé utilisée
- ☐ Repose sur la confidentialité de l'algorithme de chiffrement utilisé

B. RSA

RSA

- ☐ Est un système cryptographique asymétrique
- ☐ Repose sur la difficulté de factoriser un grand nombre en ses facteurs premiers
- ☐ Repose sur la difficulté du logarithme discret
- ☐ Permet de faire une signature digitale
- ☐ Ne peut pas être utilisé pour assurer la confidentialité

C. Devinette ?

Ce schéma

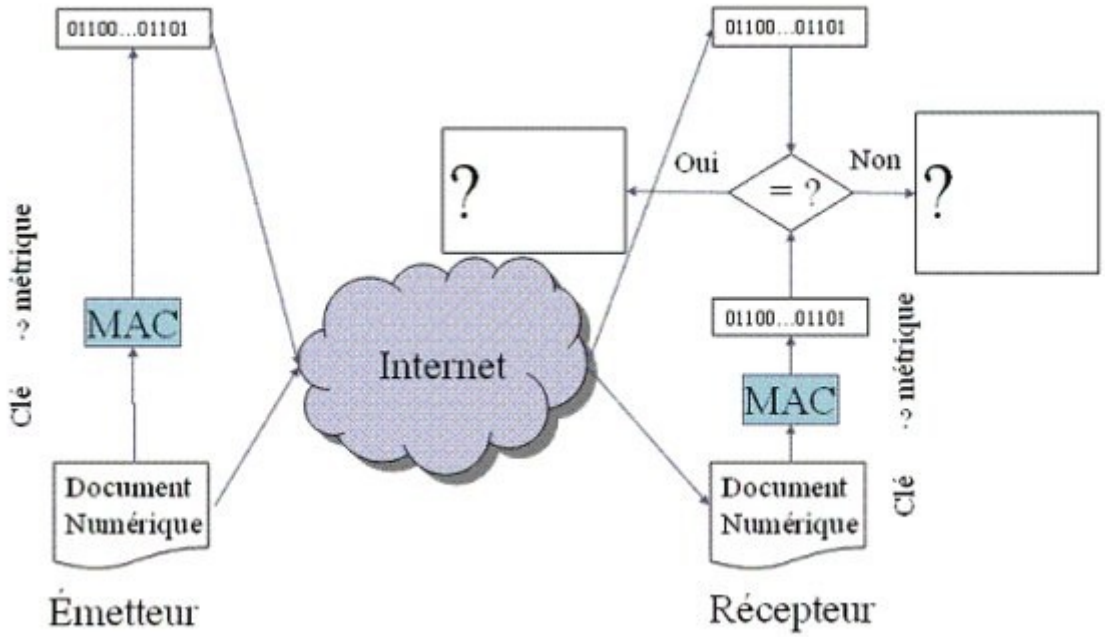


Image 29 : Q3

- [ ] Assure l'authentification de l'origine
- [ ] Utilise une clé asymétrique
- [ ] Permet d'assurer l'intégrité du Document Numérique échangé
- [ ] Utilise un MAC qui est une fonction de hashage paramétrée par une clé
- [ ] Permet de signer le document pour assurer la non répudiation

D. Non-répudiation de l'origine

Pour garantir la non-répudiation de l'origine, on peut utiliser

- [ ] Un chiffrement RSA avec la clé privée de l'émetteur
- [ ] Un MAC
- [ ] Une signature digitale
- [ ] Un chiffrement RSA avec la clé publique de l'émetteur
- [ ] Un échange de clé Diffie-Hellman

E. Man in the Middle

Soit le protocole cryptographique suivant

M1 : B ==> A : B.PKb
M2 : A ==> B : {m}PKb
Ce protocole est vulnérable à une attaque de type « man in the middle ». Un intrus "I" peut attaquer ce protocole comme suit :

☐	I intercepte M1 I bloque M1 I remplace M1 par : M1 : I ==> B : A.PKi I intercepte M2 I bloque M2 I remplace M2 par : M2 : I ==> A : {m}PKa
☐	I intercepte M1 I bloque M1 I remplace M1 par : M1 : I ==> A : B.PKi I intercepte M2 I bloque M2 I remplace M2 par : M2 : I ==> B : {m}PKb
☐	I intercepte M1 I bloque M1 I remplace M1 par : M1 : I ==> A : B.SKb I intercepte M2 I bloque M2 I remplace M2 par : M2 : I ==> B : {m}PKb

F. Certificat numérique

L'objectif d'un certificat numérique est d'assurer

☐	La confidentialité de la signature numérique du porteur du certificat
☐	L'authenticité de la clé publique correspondante à la clé privée du porteur du certificat
☐	La correspondance entre l'identité et la clé publique correspondante à la clé privée du porteur du certificat
☐	L'authenticité de la signature numérique de l'autorité de certification

G. SSL

Le protocole SSL/TLS permet d'assurer

- [ ] Le contrôle d'intégrité
- [ ] La confidentialité des échanges entre le client et le serveur
- [ ] L'authentification de l'origine de données
- [ ] L'authentification du serveur optionnellement
- [ ] L'authentification du client optionnellement

Ateliers OpenSSL

V

L'objectif de ces ateliers est de vous familiariser avec les concepts de la sécurité informatique en général et les mécanismes cryptographiques en particulier en utilisant une librairie (OpenSSL) permettant de réaliser les principales opérations cryptographiques.

Vous trouverez dans ce qui suit les transparents présentés en cours :

Dans les deux ateliers suivants, vous allez mettre en pratique les différentes notions vues en cours. Vous verrez en particulier comment réaliser les différents services de sécurité (confidentialité, authentification de l'origine, intégrité, non-répudiation) en utilisant les différents mécanismes cryptographiques offerts par la libraire OpenSSL.

Pour avoir plus de détails sur les commandes de OpenSSL et leurs options, vous pouvez vous référer au manuel suivants :

Pour réaliser ces ateliers vous devez installer la librairie OpenSSL et le serveur web Apache.

xampp permettra de les installer entre autres logiciels (PHP/MySQL, etc.)

Vous pouvez télécharge et installer xampp à partir d'ici :

A. Atelier Pratique I : Opérations Cryptographiques avec OpenSSL

Objectifs

Se familiariser avec les opérations cryptographiques: chiffrement, déchiffrement, hashage, signature digitale.

Attention : Organiser l'environnement de travail

- Pour des raisons de clarté, créer un répertoire pour la source (Source) et un répertoire pour le destinataire (Destinataire).
- Mettre tous les fichiers qui concernent la source dans le répertoire Source et tous les fichiers qui concernent le destinataire dans le répertoire Destinataire.
- A chaque fois qu'un fichier doit être transmis de la source vers le

destinataire, copier ce fichier dans le répertoire du destinataire.

- Pour pouvoir appeler l'exécutable openssl.exe, rajouter le chemin de ce dernier dans la variable d'environnement PATH. Si vous ne savez pas le faire demandez à votre enseignant.

Résumé du travail à réaliser

Voici un schéma résumant le scénario d'échange sécurisé à réaliser. Il permettra d'assurer la confidentialité, l'intégrité, l'authentification de l'origine et la non-répudiation.

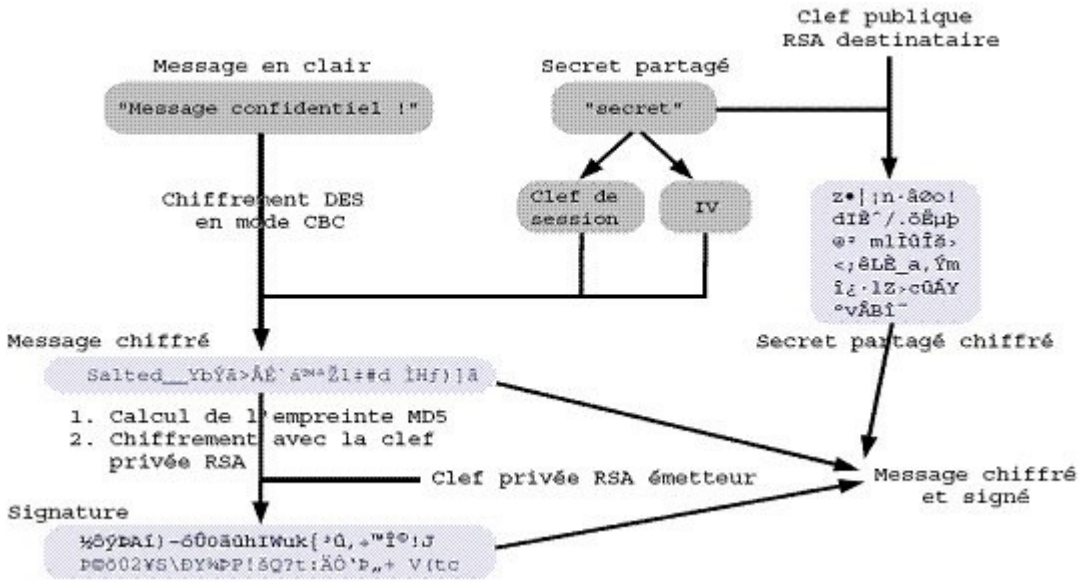


Image 30 : Résumé du travail à réaliser

1. Côté émetteur

Syntaxe : Message confidentiel

Créer un fichier "message.txt" et y mettre un texte confidentiel

Syntaxe : Générer les clés privées RSA de la source et la destination

...\Source> openssl genrsa -out src_rsa.pem -passout pass:srcpasswd -des 512

...\Destinataire> openssl genrsa -out dest_rsa.pem -passout pass:destpasswd -des 512

src_rsa.pem et dest_rsa.pem vont contenir les clés privées de la source et du destinataire protégés par les mots de passe srcpasswd et destpasswd respectivement.

Syntaxe : Extraction des clés publiques à partir des fichier "src_rsa.pem" et "dest_rsa.pem" vers "src_rsa_pub.pem" et "dest_rsa_pub.pem"

...\Source> openssl rsa -in src_rsa.pem -passin pass:srcpasswd -out src_rsa_pub.pem -pubout

...\Destinataire> openssl rsa -in dest_rsa.pem -passin pass:destpasswd -out dest_rsa_pub.pem -pubout

Ici l'option -pubout permet d'extraire la clé publique (par defaut c'est la clé privée qui est extraite)

L'option passin et passout c'est pour ptoteger les fichiers, pass:xxxx permet de faire une protection par mots de passe.

Syntaxe : échange de secret partagé

La source choisi un secret partagé (qui sera utilisé pour générer la clé symétrique partagée) par exemple "secret" (stocké dans secret.txt) et elle chiffre ce secret partagé avec la clé publique de la destination "dest_rsa_pub.pem" Ici "secret" sera utilisé par DES pour fabriquer la clé symétrique ET le vecteur d'initialisation nécessaire pour DES en mode CBC.

..\Source> openssl rsautl -in secret.txt -out secret.crypt -inkey dest_rsa_pub.pem -pubin -encrypt

Syntaxe : Chiffrement de "message.txt" avec DES-CBC en utilisant le secret partagé "secret" comme paramètre de génération de clé de session et de vecteur d'initialisation.

...\Source> openssl enc -des-cbc -in message.txt -out message.crypt -pass file:secret.txt

Syntaxe : Signature de message crypté "message.crypt"

Calcul de condensat (code de hashage) avec MD5

- ...\Source> openssl dgst -md5 -binary -out message.crypt.dgst message.crypt

Chiffrement du condensat (code de hashage) avec la clé privée de la source "src_rsa.pem"

- ...\Source> openssl rsautl -in message.crypt.dgst -out message.crypt.dgst.sign -sign -inkey src_rsa.pem

2. Côté Récepteur

La destination connait: dest_rsa.pem dest_rsa_pub.pem et recoit: "secret.crypt" "message.crypt" "message.crypt.dgst.sign"

Donc il faudra recopier ces trois fichiers du répertoire Source vers le répertoire Destinataire.

Syntaxe : Authentification de la source et de l'intégrité de donnée reçue.

Déchiffrement de l'empreinte (code de hashage) du message ==> dgst1

- ...\Destinataire> openssl rsautl -in message.crypt.dgst.sign -out dgst1 -pubin -inkey src_rsa_pub.pem

La destination calcule elle même le digest à partir de "message.crypt" ==>dgst2

- ...\Destinataire> openssl dgst -md5 -binary -out dgst2 message.crypt

La destination compare les deux condensats: (la commande FC fait un File Compare, /B fait une comparaison Binaire)

- ...\Destinataire> fc /B dgst1 dgst2

Comparaison des fichiers dgst1 et dgst2: aucune différence trouvée ==> donc l'intégrité du "message.crypt" est confirmée. ==> aussi l'authentification de la source est aussi verifiée.

Syntaxe : Déchifferement du message "message.crypt"

Déchifferement de la "clé partagée" (secret)

- ...\Destinataire> openssl rsautl -decrypt -in secret.crypt -out secret.txt -inkey dest_rsa.pem

Déchiffrement du "message.crypt" à l'aide de secret qui se trouve dans "secret.txt"

- ...\Destinataire> openssl enc -in message.crypt -out message.txt -pass file:secret.txt -d -des-cbc

Lire le message! ! !

- ...\Destinataire> type message.txt

Sevice confidentialté assuré

B. Atelier Pratique II : Sécuriser un serveur web Apache

Objectifs

Se familiariser avec les concepts d'une infrastructure à clés publiques :

- PKI
- Création d'une autorité de certification
- Création de certificats numériques
- Signature de certificats numériques

Configuration de la sécurité d'un serveur web Apache avec le protocole SSL

Configuration d'un navigateur Web pour l'authentification d'un client et d'un serveur web avec SSL.

Attention

- Si apache ne se lance pas après l'installation de xampp, penser à modifier le numéro de port. Il se peut qu'une application existante utilise le port par défaut de Apache.
- Il faut remplacer partout dans les commandes ci-dessous C:\...\xampp par le chemin exacte vers votre installation de xampp.
- Penser toujours à sauvegarder la dernière version d'un fichier de configuration avant une nouvelle modification.
- Il faut exécuter les commandes avec une grande attention. En cas de doute demander à votre enseignant. La moindre erreur pourrait nécessiter la réinstallation de xampp et refaire l'intégralité de l'atelier.

1. CHANGER LE REPERTOIRE DE PUBLICATION WEB DE APACHE

Syntaxe

1. Création d'un répertoire pour le test "delta" sous le DocumentRoot de apache: "htdocs"
2. Modifier le fichier conf: "httpd.conf" en conséquence :
   - #DocumentRoot "C:/.../xampp/htdocs"
   - DocumentRoot "C:/.../xampp/htdocs/delta"
   - #<Directory "C:/.../xampp/htdocs">
   - <Directory "C:/.../xampp/htdocs/delta">

3. Mettre une page web dans ce repertoire "delta": "index.html" (Vous pouvez créer une telle page avec MS Word en sauvegardant le document sous forme d'une page web).
4. Relancer apache.
5. Tester avec le navigateur: http://localhost

Ceci va constituer la zone libre accés du serveur

2. CREER UN REPERTOIRE POUR LA ZONE SECURISEE

Syntaxe

1. Sous "htdocs/delta" créer un répertoire "secure"
2. Modifier le fichier config "httpd-ssl.conf" en conséquence. Ce fichier se trouve sous "apache/conf/extra". Pour l'ouvrir, il faut utiliser l'éditeur « edit » à partir de la ligne de commande.
   - #DocumentRoot "C:/.../xampp/htdocs"
   - DocumentRoot "C:/.../xampp/htdocs/delta/secure"
3. Mettre une page web dans ce répertoire "delta/secure": "index.html"

3. CREER LES CETIFICAT ET LES CLES POUR LA CA ET LE SERVEUR

Syntaxe

1. Sous le répertoire "apache/conf" créer un répertoire "delta" et deux sous-repertoires "delta/certifs" et "delta/cles". Ces deux repertoires serviront à stocker les certificats et les clés de CA et de notre serveur.
2. Modifier le fichier config "httpd-ssl.conf" en conséquence. Ce fichier se trouve sous "apache/conf/extra". Pour l'ouvrir, il faut utiliser l'éditeur « edit » à partir de la ligne de commande.
   - #SSLCertificateFile conf/ssl.crt/server.crt
   - SSLCertificateFile conf/delta/certifs/serveurcert.pem
   - #SSLCertificateKeyFile conf/ssl.key/server.key
   - SSLCertificateKeyFile conf/delta/cles/serveurkey.pem
3. Création du certificat du CA
   A partir du répertoire "apache/bin" on lance la commande suivante:
   - >> openssl req -new -x509 -extensions v3_ca -keyout ..\conf\delta\cles\cakey.pem -out ..\conf\delta\certifs\cacert.pem -days 365 -config .\openssl.cnf *passwd= capasswd

   Le fichier "openssl.cnf" contient les définitions nécessaires pour la commande openssl
   Cette étape nous génère une clé pour la CA et le certificat du CA
4. Création du certificat du serveur
   A partir du répertoire "apache/bin" on lance la commande suivante:
   - >> openssl req -new -nodes -keyout ..\conf\delta\cles\serveurkey.pem -out ..\conf\delta\certifs\serveur-req.pem -config .\openssl.cnf *passwd= serveurpasswd

   Ceci va créer une requête "serveur-req.pem" qui correspond à un certificat non-signé, ainsi que la cle privée correspondante.
5. La CA signe le certificat du serveur "serveur-req.pem"
   Avant de lancer la commande, le fichier "openssl.cnf" doit être modifié en conséquence:

- Dir = ../conf/delta # Where everything is kept
- certs = $dir/certifs # Where the issued certs are kept
- database = $dir/certifs/index.txt # database index file.
- new_certs_dir = $dir/certifs # default place for new certs.
- certificate = $dir/certifs/cacert.pem # The CA certificate
- serial = $dir/certifs/serial # The current serial number
- private_key = $dir/cles/cakey.pem # The private key

Ajouter aussi un fichier "index.txt" vide dans "delta/certifs" ainsi qu'un fichier "serial" qui contient juste 01
A partir du répertoire "apache/bin" on lance la commande suivante:

- >> openssl ca -out ..\conf\delta\certifs\serveurcert.pem -config .\openssl.cnf -infiles ..\conf\delta\certifs\serveur-req.pem

Ceci nous génère le certificat "serveurcert.pem" du serveur signé par la CA

4. LES TESTS

Syntaxe

1. Relancer apache
2. On teste "https://localhost"
   Ca nous demande si on accepte le certificat du serveur: On regarde les détails du certificat et on accepte temporairement (pour la session)
   On peut redémarrer le navigateur pour voir que de nouveau le certificat n'est pas accepté automatiquement.
   On peut aussi l'accepter pour toujours (permenantly). Le certificat est alors ajouté à la liste des cetif "serveurs web"
3. On enlève le certificat du serveur du navigateur. On peut aussi inclure le certificat de la CA dans le navigateur et on se connecte de nouveau, le certificat du serveur sera accepté automatiquement car signé par un CA reconnu par le navigateur