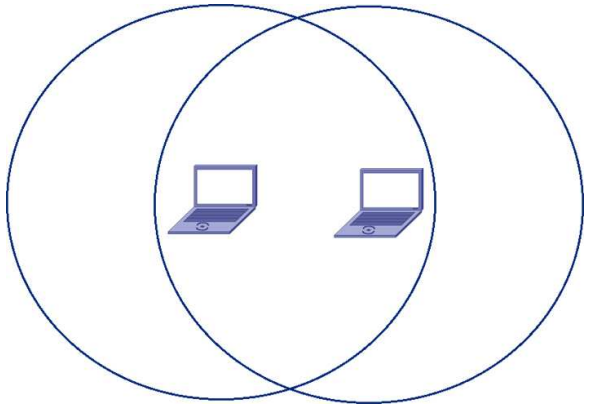
Sécurité et réseau ad hoc

Marine Minier

marine.minier@insa-lyon.fr

Introduction



- Sécurité (5h)
  - Définitions générales
  - Cryptographie
- Sécurité dans les réseaux sans-fil (1h)
  - WEP
  - Bluetooth
  - IPSec et VPN
- Sécurité dans les réseaux ad hoc (4h)
  - Autres formes de cryptographie
  - Divers propositions pratiques
- Sécurité RFID : cédric lauradoux (2h)
- Sécurité dans les réseaux de capteurs (2h)
- Evaluation : entre 2 et 4h.

Définitions générales : menaces et spécificités

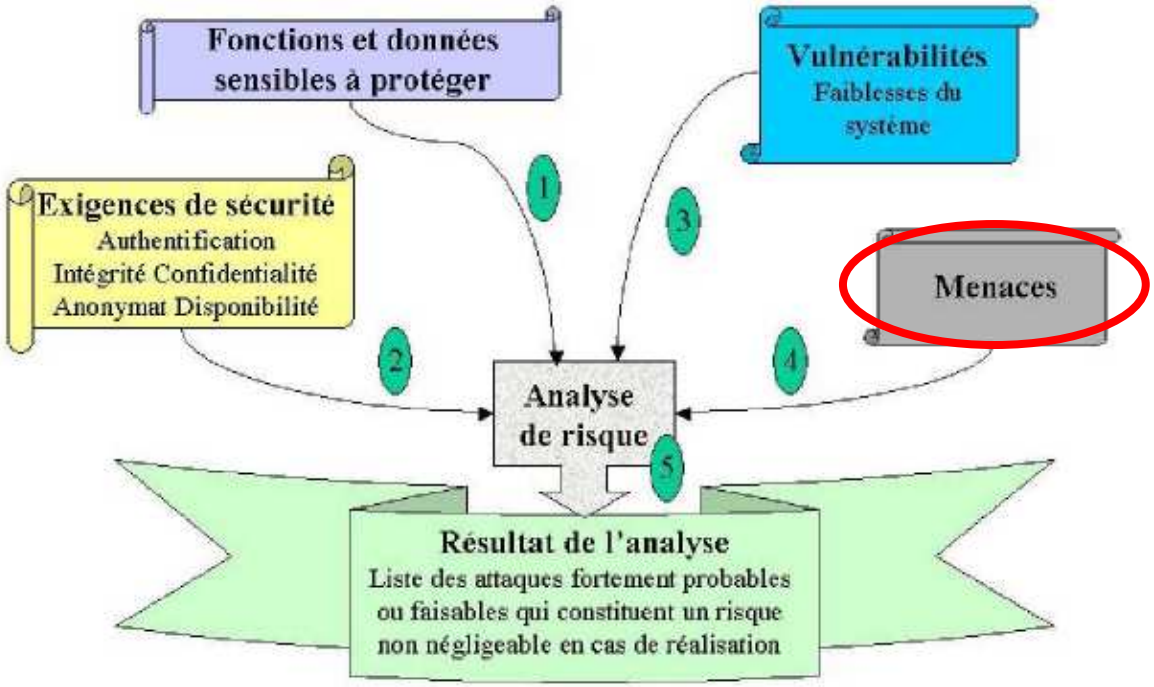
Qu’est ce que la sécurité ?

- Analyse de risques :
  - Déterminer les données sensibles
  - Recherche des exigences de sécurité fondées sur les critéres cryptographiques : authentification, intégrité, confidentialité, anonymat et disponibilité
  - Etude des vulnérabilités
  - Etude des menaces et de leurs occurrences
  - Mesure du risque

- => Détermine les parties critiques à protéger / les risques sur le réseau

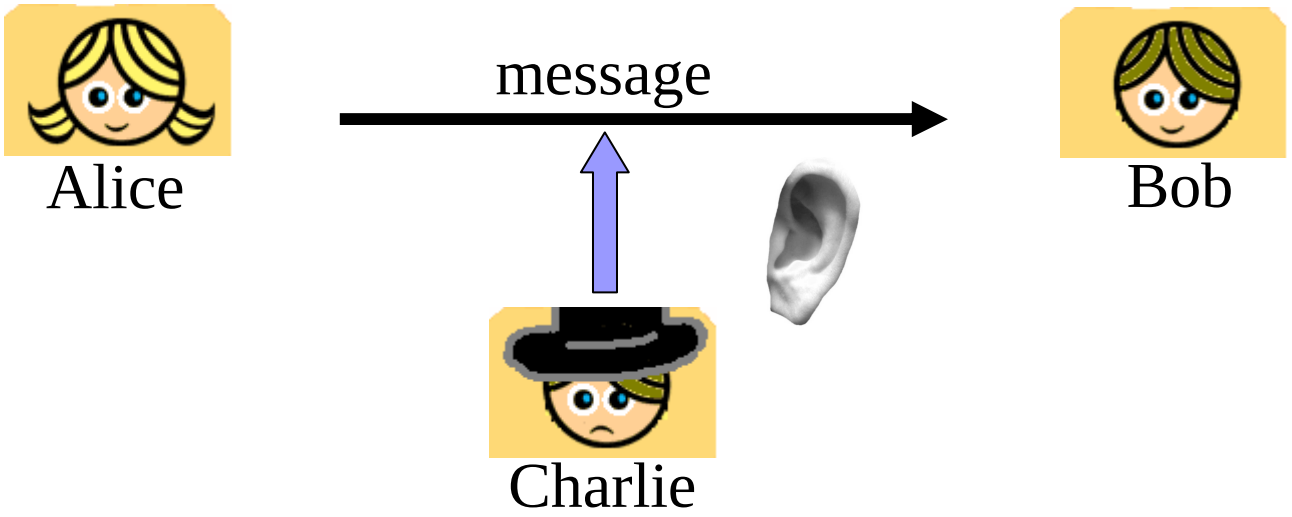
Analyse de risque :

- [Gayraud-Nuaymi-Dupont-Gombault-Tharon]



Menaces : principes d’attaques

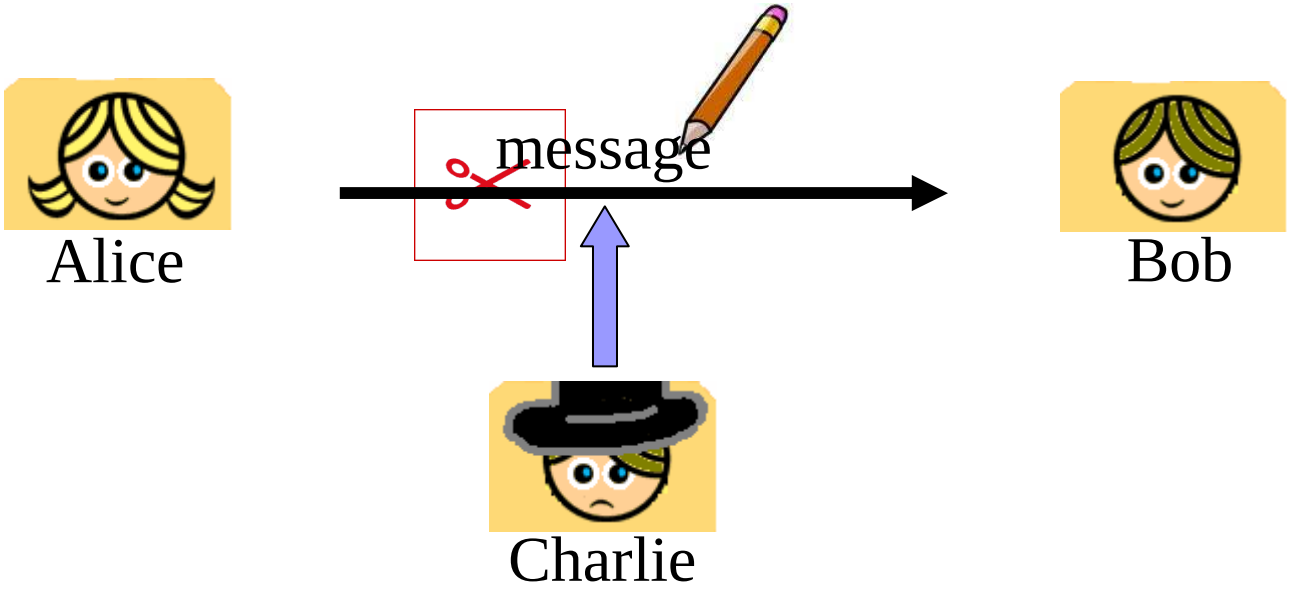
- Attaques passives



- Menace contre la *confidentialité* de l’information : une information sensible parvient à une personne autre que son destinataire légitime.

Menaces : principes d’attaques

- Attaques actives : interventions sur la ligne



- Menace contre *l’intégrité et l’authenticité* de l’information

Attaques actives : plusieurs attaques possibles

- *Impersonification* : modification de l'identité de l'émetteur ou du récepteur
- *Altération des données* (modification du contenu)
- *Destruction du message*
- *Retardement* de la transmission
- *Répudiation* du message = l'émetteur nie avoir envoyé le message

- Cryptographie : permet de lutter contre toutes ces attaques
  - Garantie la confidentialité, l'intégrité, l'authenticité (authentification et identification) et la signature

Solution à ces menaces : la cryptographie !

- La Cryptographie garantie :
  - la confidentialité : assurer que les données ne sont dévoilées qu'aux personnes autorisées
  - l'intégrité : assurer que les données ne sont pas altérées
  - l'authenticité :
    - Authentification : prouver l'origine d'une donnée
    - Identification : prouver qu'une personne est qui elle prétend être
  - La signature : rend impossible le fait de renier un document.

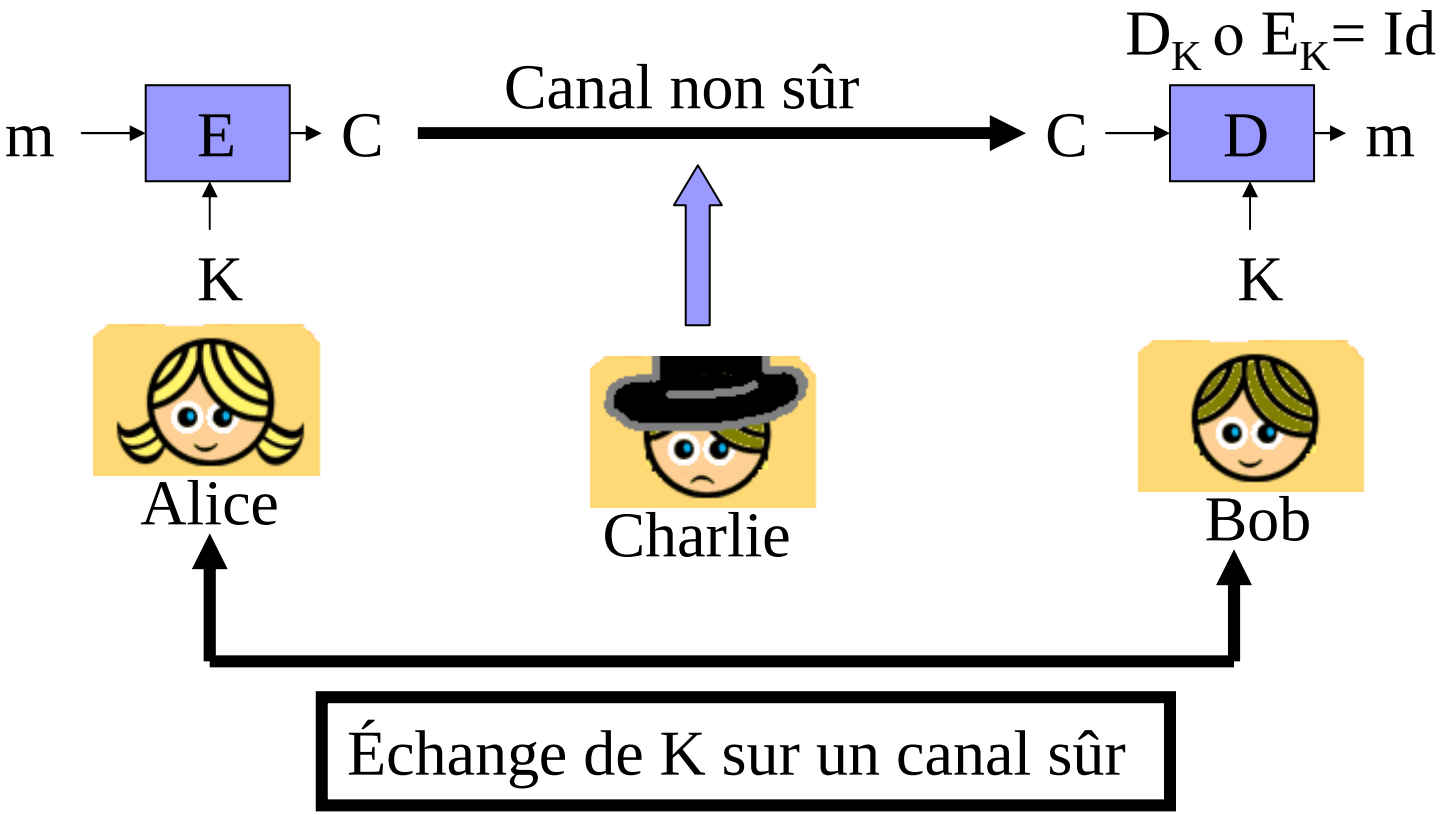
Introduction à la cryptographie classique

Assurer la confidentialité :

- Chiffrement du message :
  - Utilisation d'algorithmes de chiffrement paramétrés par des clés
- Deux méthodes :
  - Cryptographie symétrique ou à clé secrète
  - Cryptographie asymétrique ou à clé publique

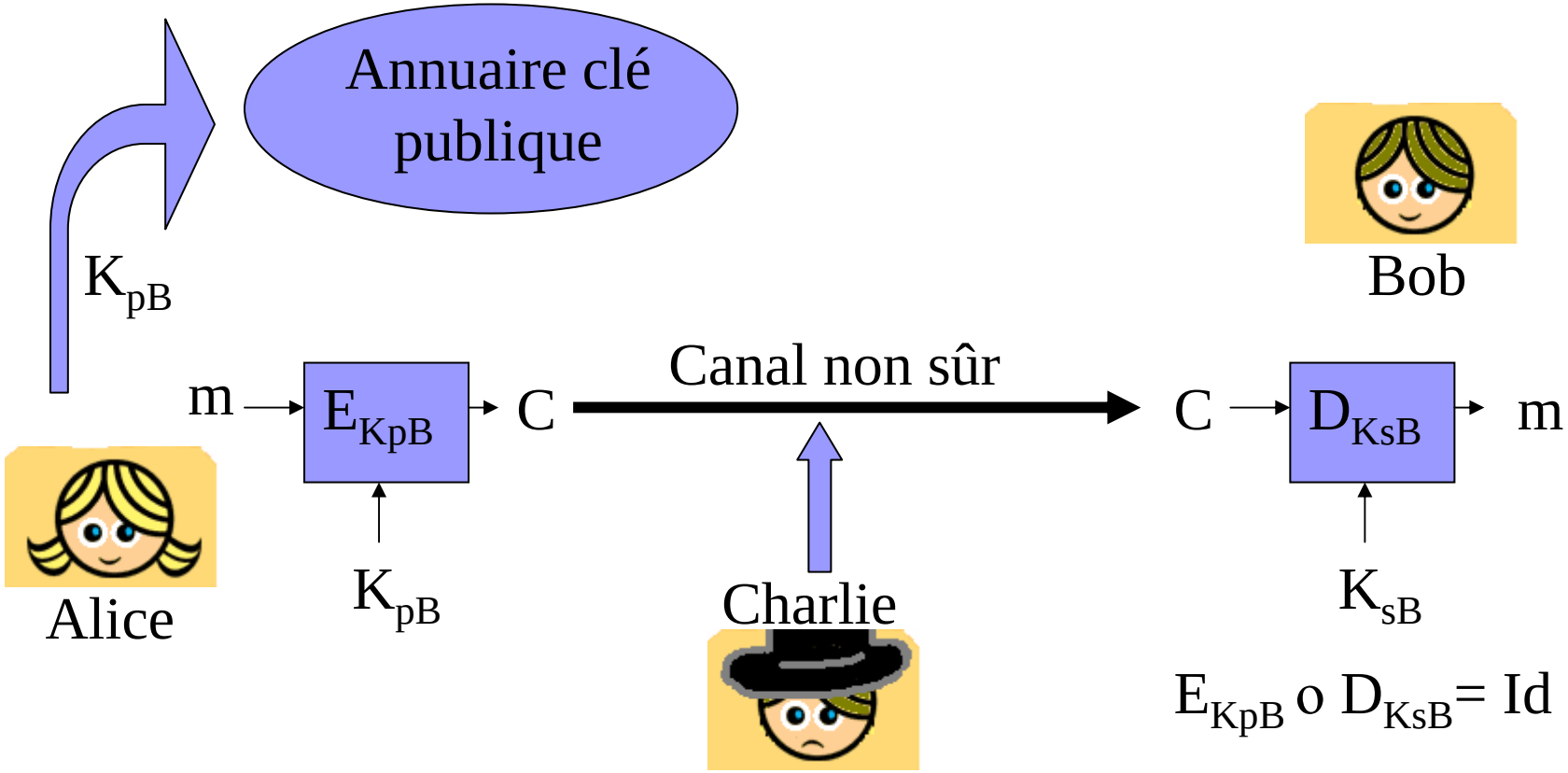
Deux méthodes pour chiffrer l'information (1/2)

- Cryptographie à clé secrète :



Deux méthodes pour chiffrer l'information (2/2)

- Cryptographie à clé publique :



Historique rapide (1/2)

- Algorithme à clé secrète plus rapide que clé publique (facteur 1000 entre les deux)

- Auparavant : la sécurité reposait sur le fait que l'algorithme utilisé était secret
  - Exemple : Alphabet de César : décalage de trois positions des lettres de l'alphabet

    => CESAR -> FHVDU

Historique rapide (2/2)

- Aujourd'hui : les algorithmes sont connus de tous : la sécurité repose uniquement sur le secret d'une clé (*principe de Kerckhoffs*).
  - Premier Exemple : Dernière guerre : Machine Enigma
  - Années 70 : développement des ordinateurs et des télécoms
  - 75-77 : Premier **standard de chiffrement** américain, le DES
  - 1976 : nouvelle forme de cryptographie : **la cryptographie à clé publique**, introduite par Diffie et Hellman (Exemple : RSA)

A quoi doit résister un bon algorithme de chiffrement ?

- Attaques de Charlie
  - but : retrouver un message m ou mieux la clé K.
  - Attaque à texte chiffré seul
  - Attaque à texte clair connu
  - Attaque à texte clair choisi

- => Complexité de ces attaques > à la recherche exhaustive (essayer toutes les clés)

Cryptographie symétrique

Cryptographie symétrique

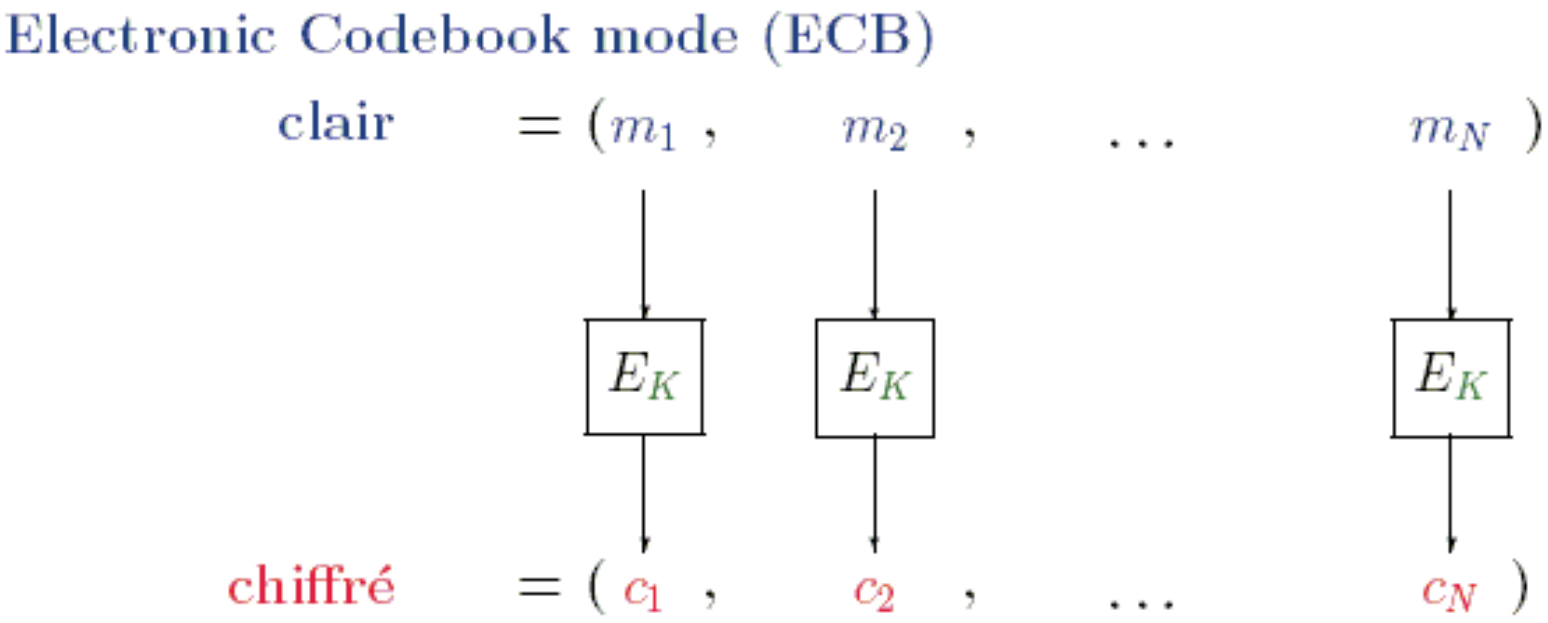
- La clé K doit être partagée par Alice et Bob
- Algorithmes étudiés
  - Algorithme de chiffrement par blocs
  - Algorithme de chiffrement à flot
  - Fonction de Hashage
- Quelques protocoles + attaques sur le WEP et attaques sur Bluetooth

Algorithmes symétriques de chiffrement par blocs

- Alice et Bob partagent la même clé K

- On chiffre par blocs :
  - Le texte clair m est divisé en blocs de taille fixe
  - On chiffre un bloc à la fois

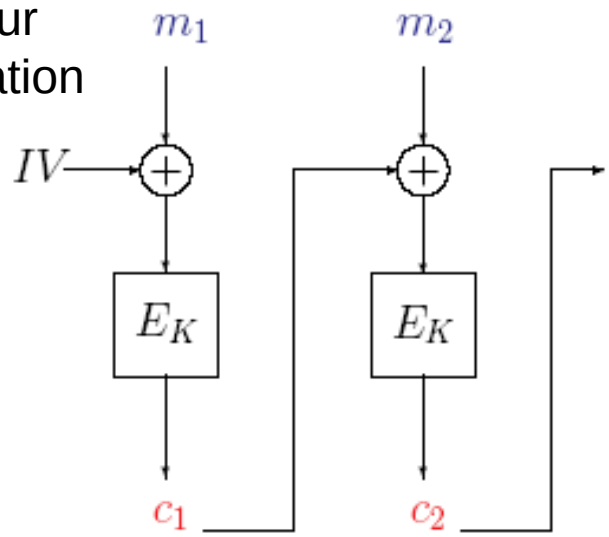
Mode ECB :


Electronic Codebook mode (ECB)
clair = (m_1 , m_2 , ... m_N)
E_K
E_K
E_K
chiffré = (c_1 , c_2 , ... c_N)

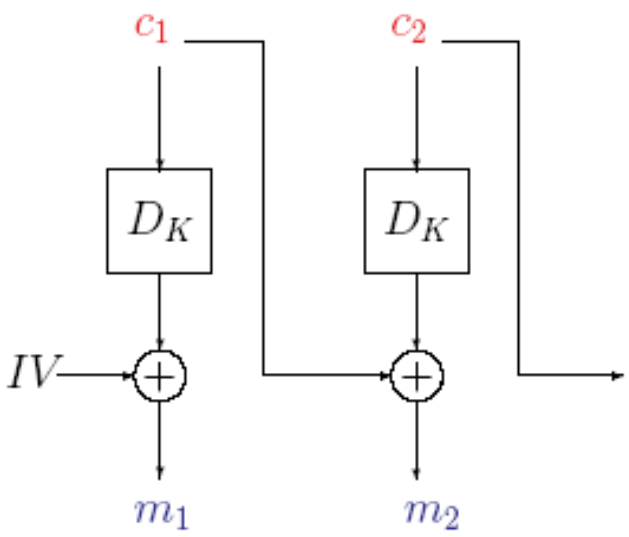
Mode CBC :

Cipher-Block Chaining mode (CBC)

IV = Valeur d'initialisation publique



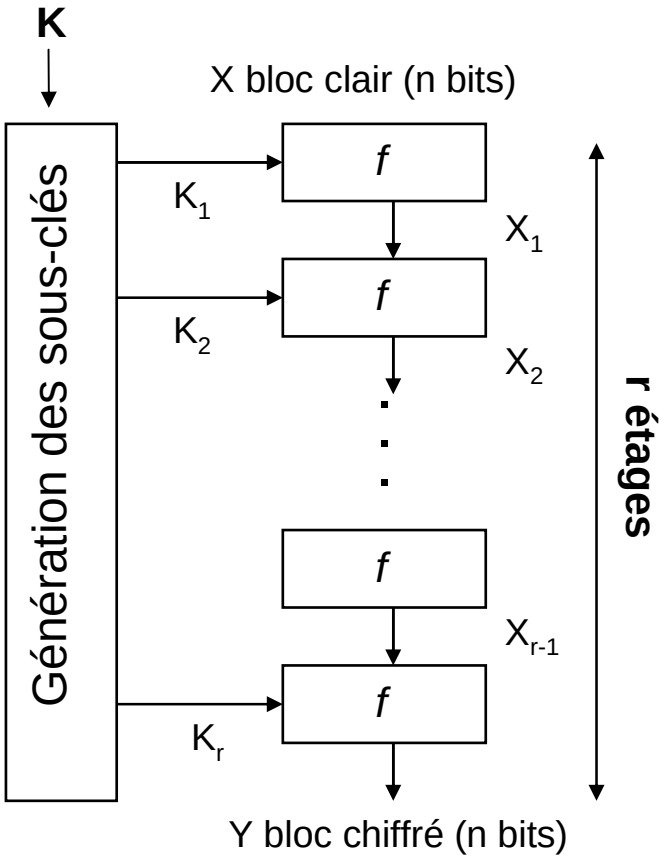
chiffrement



déchiffrement

Chiffrement par blocs itératifs

- Structure Générale d'un Algorithme de chiffrement par Blocs Itératif :



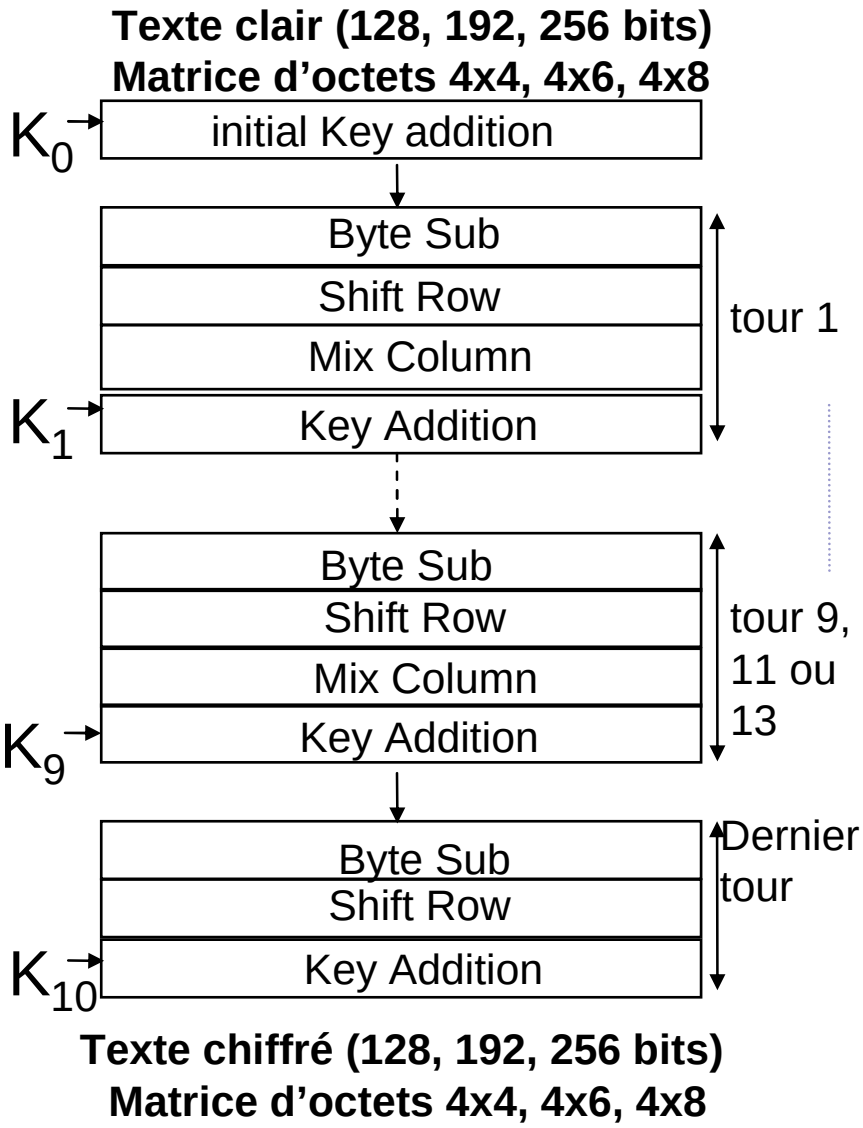
- La clé K est utilisée pour générer r sous-clés, une pour chaque étage.
- La fonction f est une permutation des blocs de n bits

Premier exemple : le DES

- DES = Data Encryption Standard
- Élaboré par le NBS (National Bureau of Standards, aujourd'hui NIST) à partir d'un algorithme d'IBM Lucifer.
- Standardisé en 1977
- Attaqué en 99 en 22h
- Remplacé par l'AES actuellement (depuis 2001)

L'AES (1/3)

- Rijndael, créé par V. Rijmen et J. Daemen, choisi comme AES en octobre 2000.
  - Algorithme de chiffrement par blocs utilisant une structure parallèle.
  - **Taille des blocs** :
    128, 192 ou 256 bits.
  - **Longueurs des clés** :
    128, 192, ou 256 bits.
  - Le **nombre de tours varie** entre 10 et 14 selon la taille des blocs et la longueur des clés.



L'AES (2/3) : : La Fonction Étage 1/2

✶ **Byte Substitution**

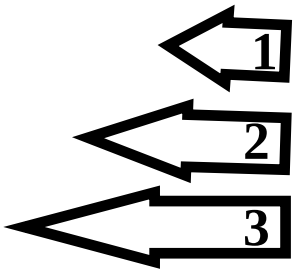
a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{13}
a_{20}	a_{21}	a_{22}	a_{23}
a_{30}	a_{31}	a_{32}	a_{33}

(8x8 S-box S) ↓

$S(a_{00})$	$S(a_{01})$	$S(a_{01})$	$S(a_{00})$
$S(a_{13})$	$S(a_{12})$	$S(a_{11})$	$S(a_{10})$
$S(a_{23})$	$S(a_{22})$	$S(a_{21})$	$S(a_{20})$
$S(a_{33})$	$S(a_{32})$	$S(a_{31})$	$S(a_{30})$

✸ **Shift Row**

a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{13}
a_{20}	a_{21}	a_{22}	a_{23}
a_{30}	a_{31}	a_{32}	a_{33}

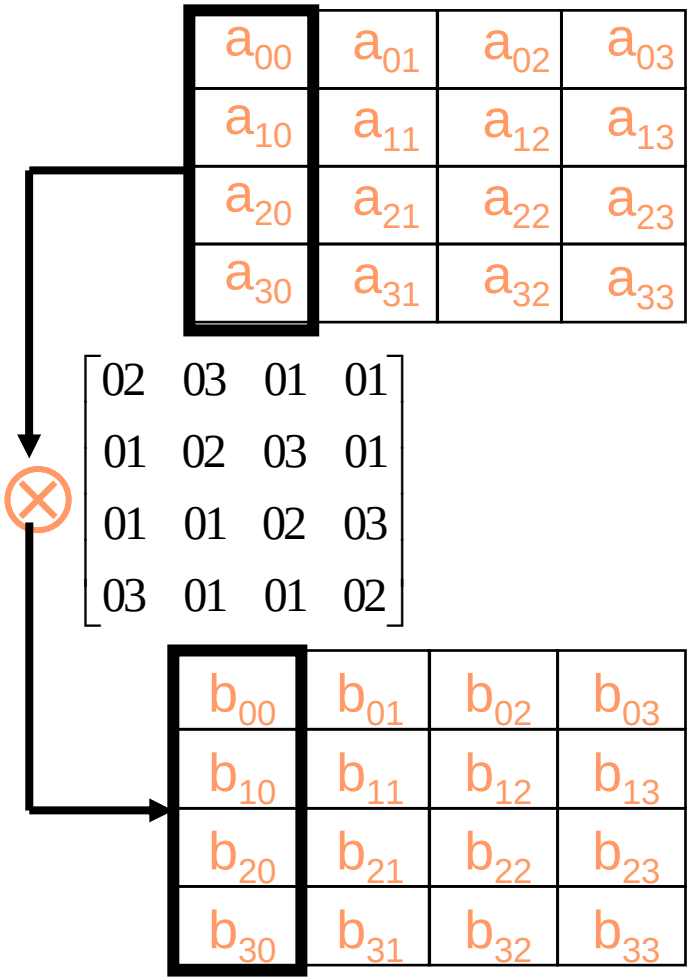


↓

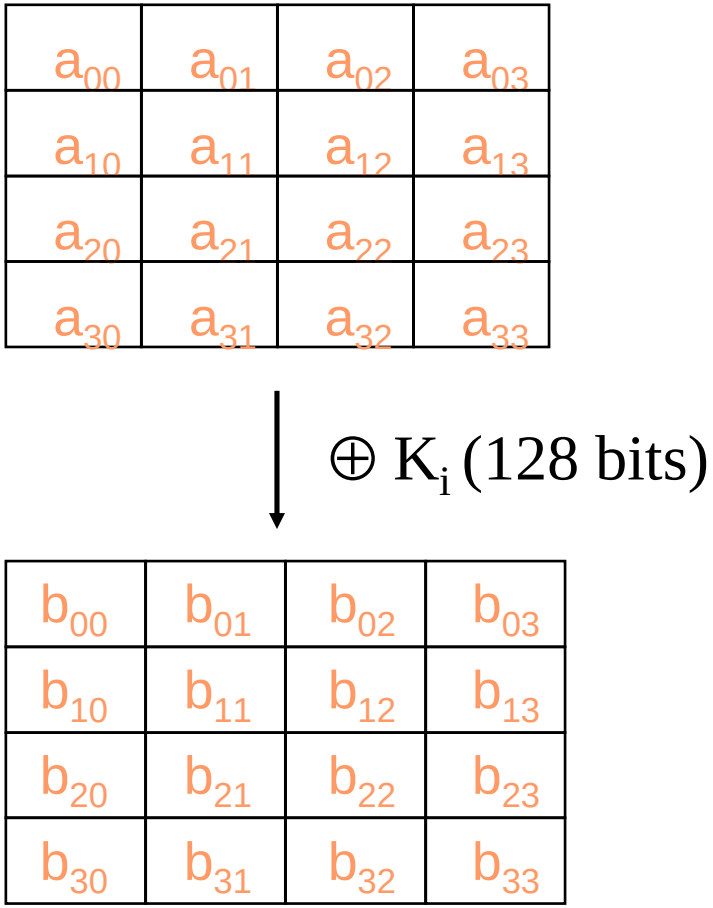
a_{00}	a_{01}	a_{02}	a_{03}
a_{11}	a_{12}	a_{13}	a_{10}
a_{22}	a_{23}	a_{20}	a_{21}
a_{32}	a_{30}	a_{33}	a_{31}

L'AES (3/3) : : La Fonction Étage 2/2

Mix Column



Key Addition



Quel chiffrement utilisé aujourd'hui ?

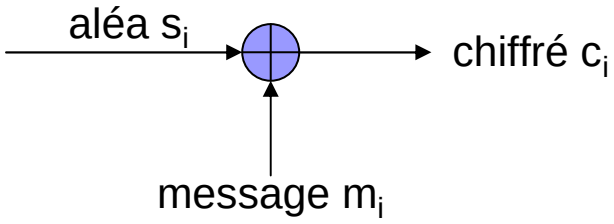
- Soit l'AES
- Soit le triple DES :
  - composition de deux DES
  - avec deux clés (112 bits de clé) :

$$C = DES_{K1}(DES^{-1}_{K2}(DES_{K1}(M)))$$

=>Pour se prémunir contre la recherche exhaustive

Autre algorithme de cryptographie symétrique : le chiffrement à flot

- Utilisation du « one time pad » :



$$m = m_0\ m_1\ m_2\ m_3\ \ldots$$
$$\oplus\ s = s_0\ s_1\ s_2\ s_3\ \ldots$$
$$= c = c_0\ c_1\ c_2\ c_3\ \ldots$$

- L'aléa est remplacé par un générateur pseudo-aléatoire (ou chiffrement à flot)
  - Initialisé par la clé commune K
  - Sécurité repose sur les qualités du générateur (grande période, très bon aléa,…)

Chiffrement à flot :

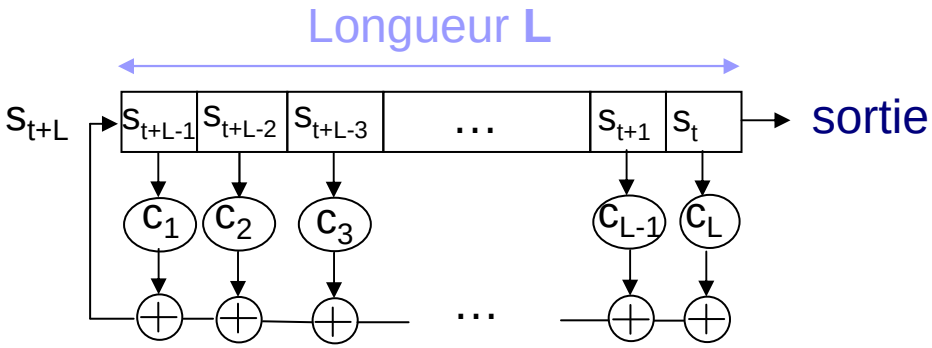
- Pourquoi des chiffrements à flot ?
  - Utilisation pour le software : chiffrement très rapide
  - Utilisation en hardware avec des ressources restreintes
  - Ne propage pas les erreurs (souvent utilisé en téléphonie mobile) (à la différence du chiffrement par blocs)

Conceptions classiques :

- En général, trois phases :
  - Un état initial de longueur L (L ≥ 2k où k est la longueur de clé)
  - Une fonction de remise à jour de l'état
  - Une fonction de filtrage pouvant dissimuler les propriétés de la fonction précédente

- Constructions les plus usitées :
  - État initial = clé (et/ou vecteur d'initialisation)
  - Utilisation d'un LFSR pour remettre l'état à jour
  - Fonction de filtrage :
    - Fonction booléenne qui filtre les sorties d'un seul LFSR
    - Fonction booléenne qui combine les sorties de plusieurs LFSRs

Le LFSR : Registre à rétroaction linéaire



Pour tout $t \geq L$, $s_t = \Sigma_{i=1..L} c_i s_{t-i}$

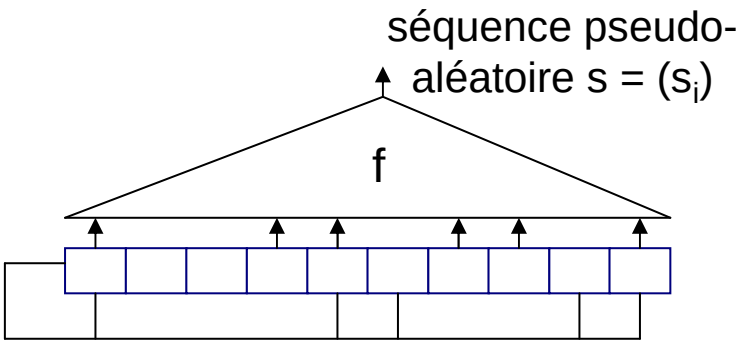
- Polynôme de rétroaction :

$P(X) = 1 + c_1X + c_2X^2 + \ldots + c_LX^L$ Choisi pour être primitif

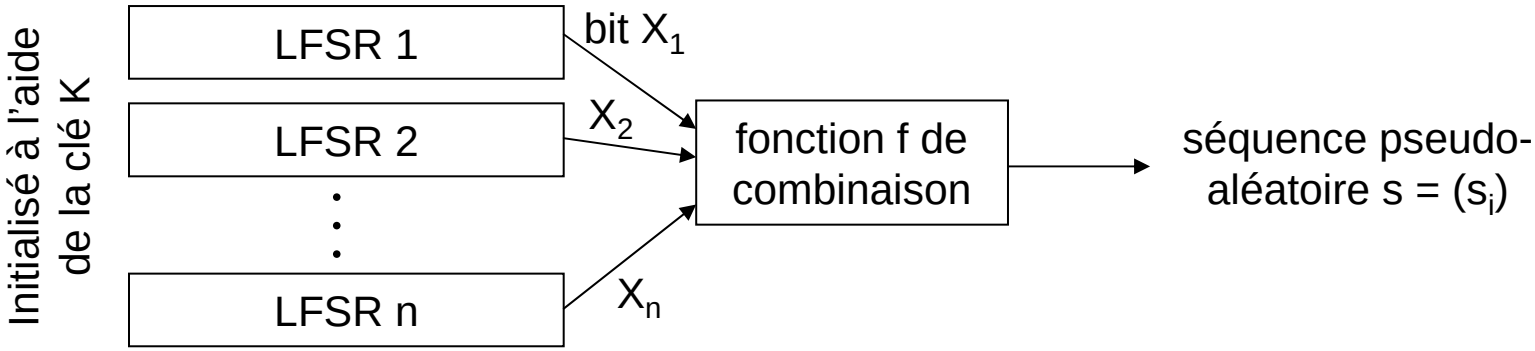
$P^*(X) = X^L + c_1X^{L-1} + c_2X^{L-2} + \ldots + c_L$

Utilisation de LFSRs :

- Le registre filtré :

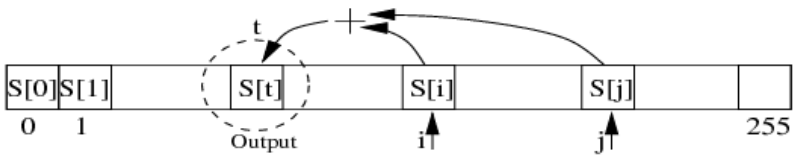


- La combinaison de registres :



Un exemple particulier RC4 :

- Principe général :
  - Génération de l'aléa à partir d'un tableau d'état S_i de 256 octets
- Initialisation du tableau à partir de la clé K
  - Pour i de 0 à 255, S_i = i
  - Pour i de 0 à 255, K_i = clé K
  - j = 0, pour i de 0 à 255
    - j = (j + S_i+ K_i) mod 256
    - Échanger S_i et S_j
- Génération de l'aléa :
  - i = (i+1) mod 256, j = (j + S_j) mod 256
  - Échanger S_i et S_j
  - t = (S_i + S_j) mod 256 => sortir S_t



Comparaison de performances :

- En hardware (2003)
  - DES : 1,1 Gbits/ seconde
  - AES : 1,95 Gbits/s.
  - RC4 : 0,685 Gbits/s. (vieil algorithme)

Fonctions de hachage

- Calcul d'un condensé h d'un message M :

  $$h = H(M)$$

- Propriété : résistance aux collisions

  Il doit être très difficile de trouver un couple de messages (M, M') qui ont le même condensé. (« one way hash function »).

- On part d'un message de taille quelconque et on construit un condensé de taille N bits
  - N > 160 bits pour éviter la recherche exhaustive et les attaques par collisions

Paradoxe des anniversaires

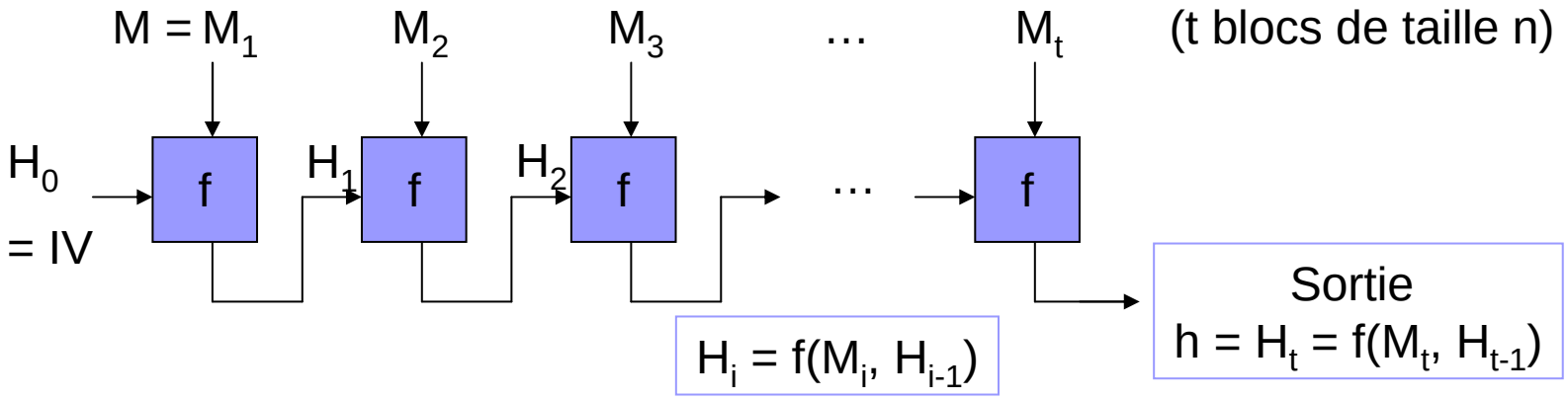
- Problème : Combien faut-il de personnes dans une salle pour avoir plus d'une chance sur deux pour que 2 personnes soient nées le même jour ?

- Réponse : 23 !
  - Nombre de personnes : $1{,}18.n^{1/2}$
  - Avec n = nb d'événements (ici 365)

- Cas du hachage : si le haché fait 128 bits, alors il faut essayer environ 2^{64} messages pour obtenir une collision
- => N > 160 bits

Méthode de constructions :

- Construites à partir d'une fonction de compression f :



- Exemple : f =AES et $H_i = AES_{Mi}(H_{i-1}) \oplus H_{i-1}$

Exemples :

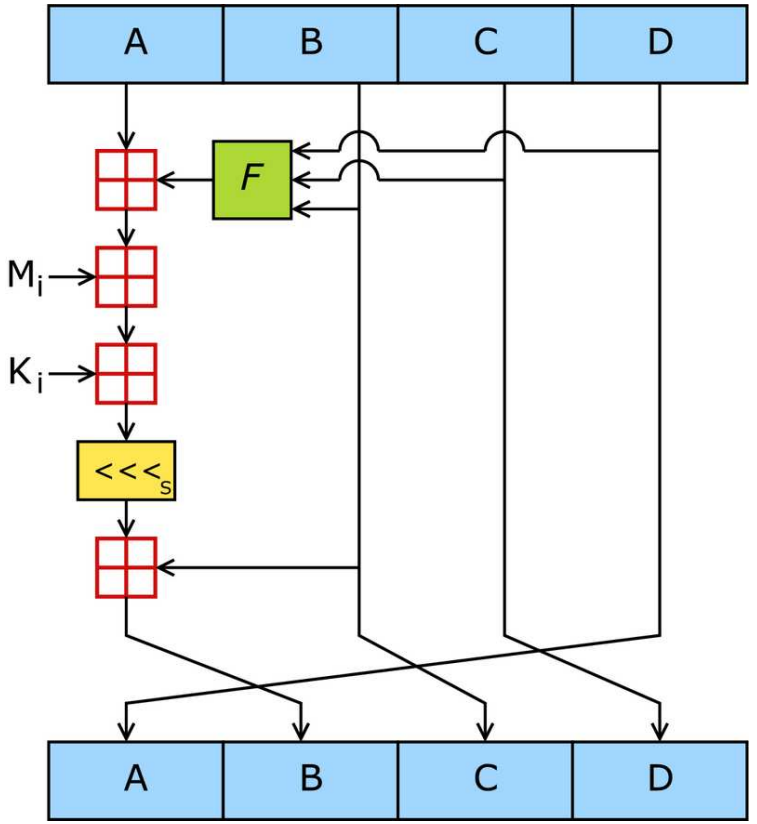
- MD4 [Rivest 92] , MD5 [Rivest 92]
  - MD5 : entrée de 512 bits -> hash de 128 bits
- SHA-0, SHA-1, SHA-256 ou 384 ou 512 proposé par la NSA (National Security Agency)
  - SHA-1 : entrée de 512 bits -> hash de 160 bits

MD5 :

- Composé de la répétition de 64 opérations regroupées en 4 fois 16 opérations

- $F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$

4 mots de 32 bits

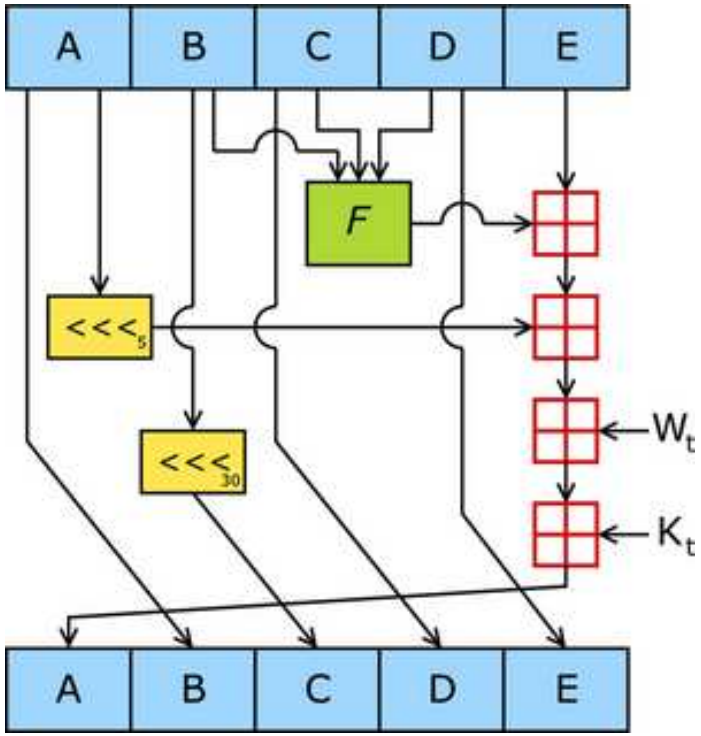


Une opération

SHA-1 :

- Composé de la répétition de 80 opérations regroupées en 4 fois 20 opérations
  - K_t = constante
  - W_t = valeur dépendant des blocs M_i du message

5 mots de 32 bits



Une opération

$$f_t(x,y,z) = \begin{cases} Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z), & \text{si } 0 \leq t \leq 19 \\ Parity(x,y,z) = x \oplus y \oplus z, & \text{si } 20 \leq t \leq 39 \\ Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z), & \text{si } 40 \leq t \leq 59 \\ Parity(x,y,z) = x \oplus y \oplus z, & \text{si } 60 \leq t \leq 79 \end{cases}$$

Utilité des fonctions de hachage

- Informatique : construction de table de hachage, liste chaînée utilisant ces fonctions.

- Permet de garantir l'intégrité
  - Téléchargement de packages (Openoffice,…)
  - ⇒ Vérification de l'intégrité des paquets par calcul de sommes MD5

  - Utilisation avec une signature numérique (voir plus loin)

  - Calcul de mot de passe

Les fonctions de hachage attaquées !

- Résultats de 2004 et 2005
- temps nécessaires pour trouver deux messages M et M’ fournissant le même haché h :
  - MD4 : 15 minutes
  - MD5 : 8 heures sur un PC à 1,6 GHz
  - SHA-0 : 2^{39} opérations, SHA-1 : 2^{63} opérations
- Seul SHA-1 peut encore être utilisée
- On parle d’un éventuel appel d’offre du NIST…

Exemples de collision pour MD5

Example: MD5 collision with the standard IV

IV according to [2]:

```
  context->state[0] = 0x67452301;
  context->state[1] = 0xefcdab89;
  context->state[2] = 0x98badcfe;
  context->state[3] = 0x10325476;
```

First message:

```
0xA6,0x64,0xEA,0xB8,0x89,0x04,0xC2,0xAC,
0x48,0x43,0x41,0x0E,0x0A,0x63,0x42,0x54,
0x16,0x60,0x6C,0x81,0x44,0x2D,0xD6,0x8D,
0x40,0x04,0x58,0x3E,0xB8,0xFB,0x7F,0x89,
0x55,0xAD,0x34,0x06,0x09,0xF4,0xB3,0x02,
0x83,0xE4,0x88,0x83,0x25,0x71,0x41,0x5A,
0x08,0x51,0x25,0xE8,0xF7,0xCD,0xC9,0x9F,
0xD9,0x1D,0xBD,0xF2,0x80,0x37,0x3C,0x5B,
0x97,0x9E,0xBD,0xB4,0x0E,0x2A,0x6E,0x17,
0xA6,0x23,0x57,0x24,0xD1,0xDF,0x41,0xB4,
0x46,0x73,0xF9,0x96,0xF1,0x62,0x4A,0xDD,
0x10,0x29,0x31,0x67,0xD0,0x09,0xB1,0x8F,
0x75,0xA7,0x7F,0x79,0x30,0xD9,0x5C,0xEB,
0x02,0xE8,0xAD,0xBA,0x7A,0xC8,0x55,0x5C,
0xED,0x74,0xCA,0xDD,0x5F,0xC9,0x93,0x6D,
0xB1,0x9B,0x4A,0xD8,0x35,0xCC,0x67,0xE3.
```

Second message:

```
0xA6,0x64,0xEA,0xB8,0x89,0x04,0xC2,0xAC,
0x48,0x43,0x41,0x0E,0x0A,0x63,0x42,0x54,
0x16,0x60,0x6C,0x01,0x44,0x2D,0xD6,0x8D,
0x40,0x04,0x58,0x3E,0xB8,0xFB,0x7F,0x89,
0x55,0xAD,0x34,0x06,0x09,0xF4,0xB3,0x02,
0x83,0xE4,0x88,0x83,0x25,0xF1,0x41,0x5A,
0x08,0x51,0x25,0xE8,0xF7,0xCD,0xC9,0x9F,
0xD9,0x1D,0xBD,0x72,0x80,0x37,0x3C,0x5B,
0x97,0x9E,0xBD,0xB4,0x0E,0x2A,0x6E,0x17,
0xA6,0x23,0x57,0x24,0xD1,0xDF,0x41,0xB4,
0x46,0x73,0xF9,0x16,0xF1,0x62,0x4A,0xDD,
0x10,0x29,0x31,0x67,0xD0,0x09,0xB1,0x8F,
0x75,0xA7,0x7F,0x79,0x30,0xD9,0x5C,0xEB,
0x02,0xE8,0xAD,0xBA,0x7A,0x48,0x55,0x5C,
0xED,0x74,0xCA,0xDD,0x5F,0xC9,0x93,0x6D,
0xB1,0x9B,0x4A,0x58,0x35,0xCC,0x67,0xE3.
```

Common MD5 hash:

```
0x2B,0xA3,0xBE,0x5A,0xA5,0x41,0x00,0x6B,
0x62,0x37,0x01,0x11,0x28,0x2D,0x19,0xF5.
```

MACs :

- Message authentication Algorithms
- = Fonction de hachage avec clé

- Exemple :
  - H(k,p,m,k) : application de deux fonctions de hachage avec une clé k
  - p = padding = on complète le message m avec des 0, des 1 ou une valeur aléatoire

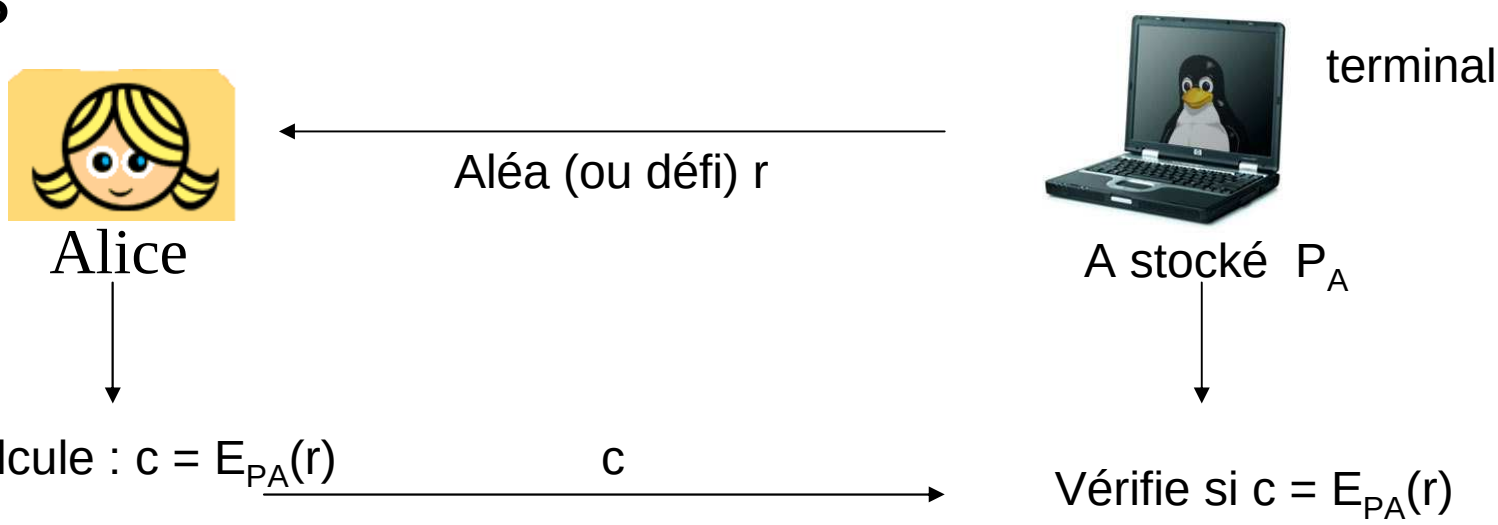
Protocoles dédiés :

- Cryptographie = algorithmes + protocoles
- La solidité d’une communication dépend à la fois de :
  - La solidité des algorithmes cryptographiques
  - Des protocoles utilisées

Protocole aléa/retour :

- Première connexion : Alice envoie PA et le terminal stocke PA.

Puis



SKEY : RFC 2289

Alice

(1000, $c_A = h^{1000}(P_A)$)

terminal

Stocke (1000, cA = h1000(PA))

Alice

1000-1

terminal

Calcule $c = h^{999}(P)$

c

Vérifie si $h(c) = c_A$

Stocke (999, c)

Conclusion partielle

- Les algorithmes de chiffrement à clé secrète ne permettent pas de garantir la non-répudiation
- Problème de transmission de la clé partagée
- Quand bcp d'utilisateurs => problème de gestion de clé
- Problème de renouvellement des clés

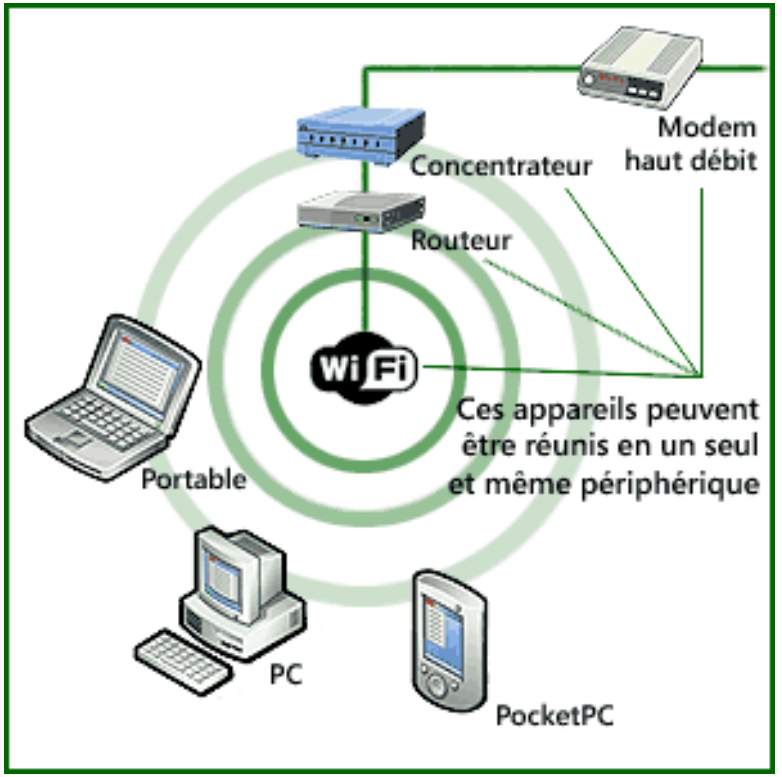
=> Solution : cryptographie à clé publique !

Sécurité des réseaux sans fil :

- Attaque sur le WEP
- Attaque sur Bluetooth

Première étude de cas : le WEP

- Sécurité dans les réseaux sans fil
- WEP : Wired Equivalent Protocol
- Ce protocole sécurise les données de la couche liaison pour les transmissions sans fil de la norme 802.11 (première version 99)



WEP (1/3)

- Il présuppose l'existence d'une clé secrète entre les parties communicantes (la clé WEP) pour protéger le corps des frames transmises

- L'utilisation du WEP est optionnel

- Il n'y a pas de protocoles de gestion de clé

=> Une seule clé partagée par plusieurs utilisateurs

WEP (2/3)

- Pour chiffrer un message M
  - Checksum : calcule de c(M), P=(M, c(M))

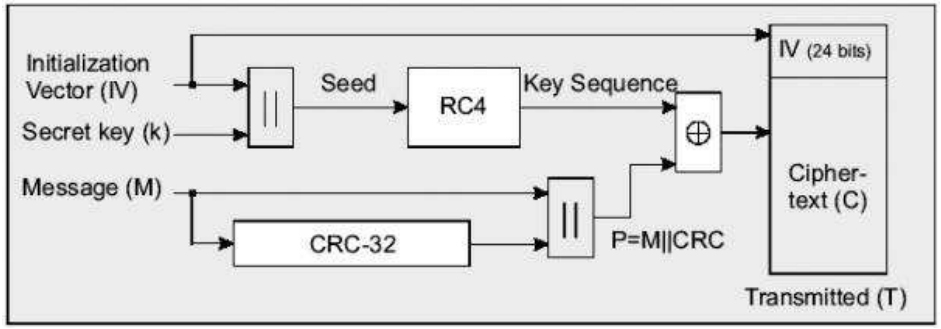
    code linéaire CRC-32 (intégrité)

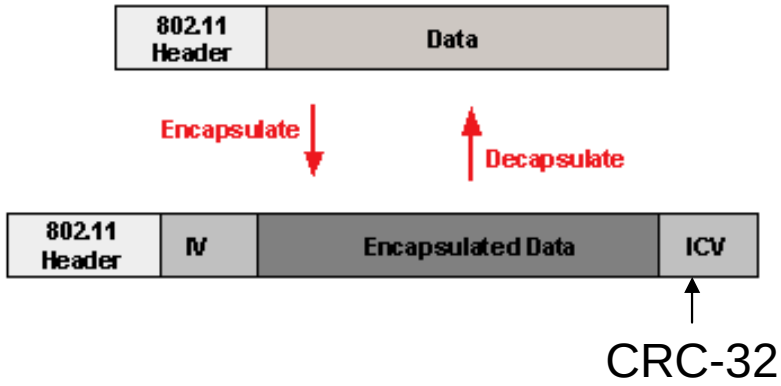
  - Chiffrement : P est chiffré avec RC4 (confidentialité)

  - Un vecteur d'initialisation (IV) v est choisi et est concaténé à k :

    C =P+RC4(v,k)

    Transmission de v et C





WEP : authentification (3/3)

- Pour s'authentifier : protocole aléa-retour
  - Aléa r de 128 bits
  - L'aléa est chiffré avec la méthode précédente avant vérification :
    - C' = r+RC4(v,k)
    - Le serveur vérifie si C' = C (la valeur calculée par le serveur)

WEP : pourquoi un IV ?

- Pourquoi un IV ?
  - Si pas d'IV : $C_1=P_1+RC4(k)$ et $C_2=P_2+RC4(k)$

    => $C_1 + C_2 = P_1 + P_2$
  - Si on connaît P_1 => déduit P_2
  - En chiffrant avec un IV, on change d'IV à chaque message

    => cette attaque est évitée

WEP : quelques problèmes…

- **Une clé statique** : aucun chiffrement n'est sûr si on réutilise toujours la même clé

- **Taille de l'IV** : 24 bits => seulement 16,777,216 d'IV possibles
- 802.11 ne rend pas obligatoire l'incrémentation de l'iV !

- **Longueur de clé** :
  - 40 bits (5 caractères ASCII)
  - ou 104 bits (13 caractères ASCII)
  - => attaque par recherche exhaustive possible !

=> Taille totale : entre 64 et 128 bits

- **CRC 32 : le code est linéaire: c(x+y) = c(x) +c(y) !**
  - Pas de protection de l'intégrité des données
  - Attaque par changement de seulement 1 bit du message

- **Pas de spécification sur la distribution des clés**

Quelques problèmes (suite)...

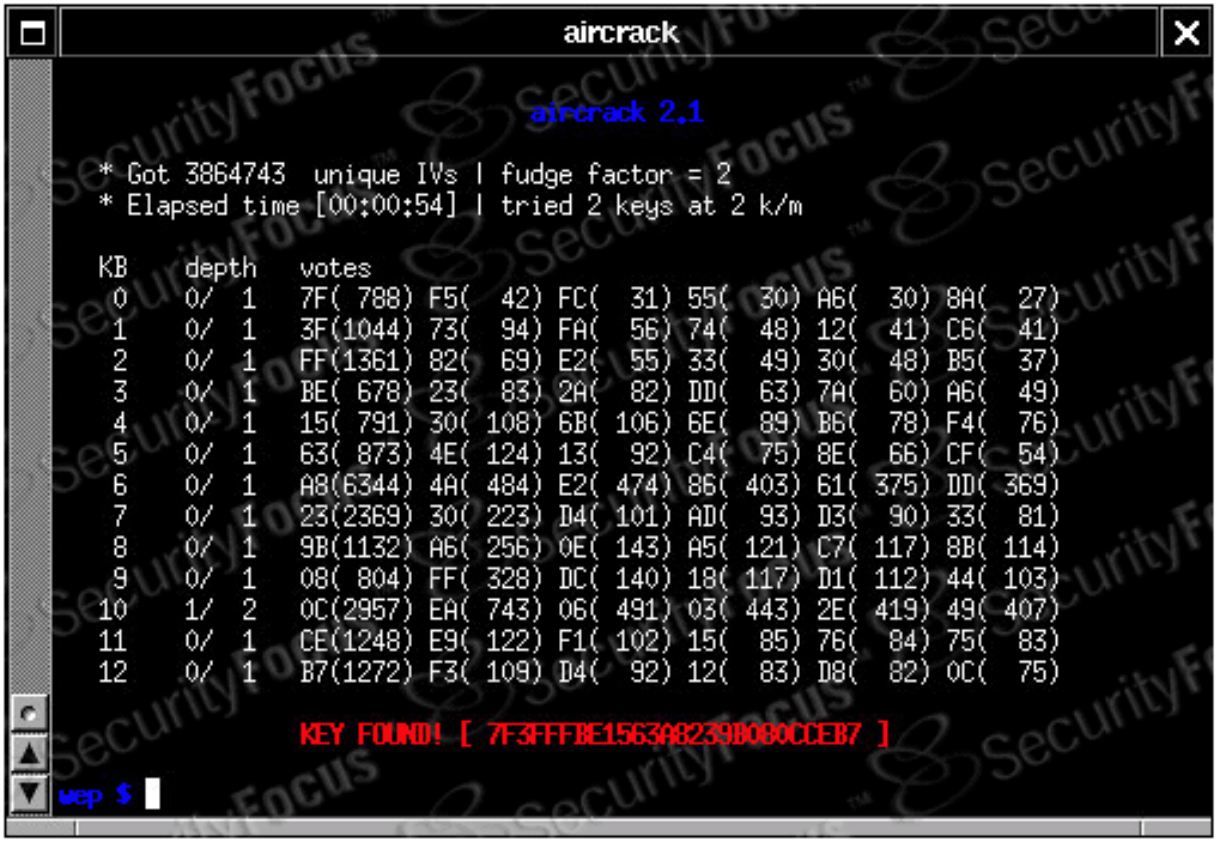
- Attaque sur la confidentialité
- Deux attaques statistiques contre RC4 avec des IVs
  - FSM 2001 : des IV sont faibles et révèlent de l'information sur la clé à l'aide du premier octet de sortie
  - Amélioration de cette attaque par Hulton => utilisation des premiers octets de sortie => permet de réduire la quantité de données à capturer
  - Attaque de KoreK (2004) : généralisation des deux attaques précédentes + injection de paquets.
  - On détermine le reste de la clé par recherche exhaustive

Implémentation des attaques

- Attaque de KoreK
  - Clé de 40 bits : nécessite la capture de 150.000 paquets avec des IVs différents
  - Clé de 104 bits : capture de 500.000 de paquets avec IV diff.
- Amélioration par injection de trafic (ARP)
  - Récupérer un paquet WEP, enlever le dernier octet. => le CRC/ICV est cassée.
  - Test sur la valeur de l' octet d'avant :
    - si il valait 0 => XOR des 4 derniers octets et d'une autre valeur => CRC encore valide ? Retransmission du paquet, passe-t'il ? Oui, bonne valeur, non
    - => essayer la valeur 1 =>,…
    - Recherche exhaustive sur la valeur de un octet et on remonte,…

Exemple d’attaque sur le WEP

- Exemple :
  - Aircrack de C. Devine
  - Très rapide
- Avec injection de paquets,
- => qques min. pour trouver la clé

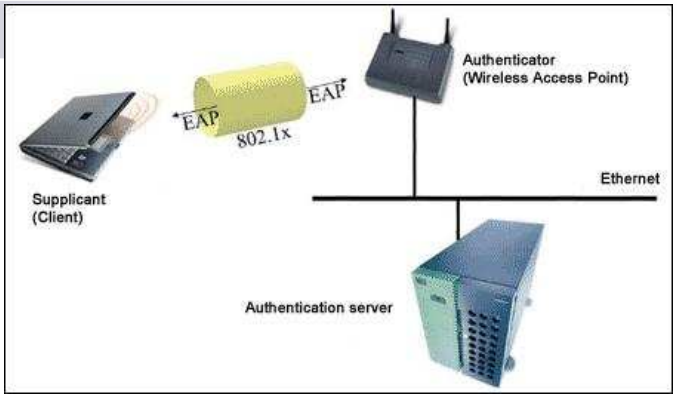


Le WEP est mort : résumé

- Les principales faiblesses :
  - Faiblesse de RC4 < construction clé (K, IV)
  - Taille trop faible des IV + réutilisation des IV autorisée
  - Pas de contrôle d'intégrité avec CRC32
  - Pas de mécanisme de mise à jour des clés

- 3 outils nécessaires à l'attaque
  - Aerodump : découverte des réseaux autour
  - Aireplay : injection artificielle de trafic
  - Aircrack : casseur de clé WEP utilisant les Ivs uniques collectés auparavant

Changements



- Utiliser 802.1x pour l'authentification EAP (Extensive Authentication Protocol RFC 2284) centralisée

- Utiliser WPA pour la confidentialité
  - Changement de clé de chiffrement de façon périodique
  - Clé de 128 bits
  - IV de 48 bits
  - Impossibilité de réutiliser un même IV avec la même clé
  - Utilisation d'un contrôle d'intégrité du message (MIC) avec SHA-1
  - Malheureusement : pas encore l'AES => WPA 2 !
  - WPA 2 intègre des protocoles standards dans toutes les couches
    - Intégration de IP-Sec, https,TCP protégé par TLS,…

- Attention : attaque de WPA et WPA2 dans le mode PSK (mode dégradé) => mais seulement brute force

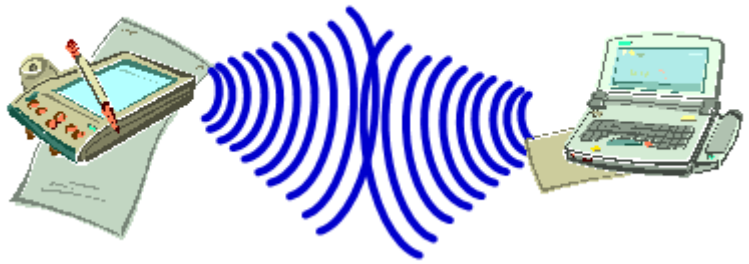
Nouvelle norme : 802.11i (2004)

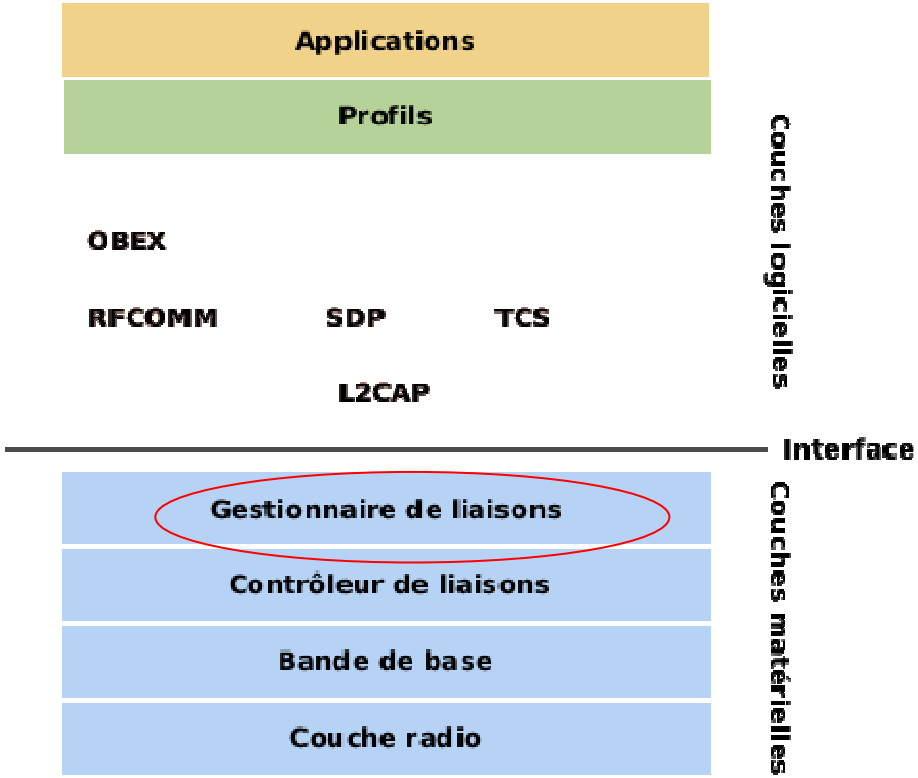
- Nom commercial : WPA2
  - Séparation authentification util. et chiffrement
  - Architecture de sécurité appelé RSN (robust security network) en 4 phases :
    - Mise en accord d'une politique de sécurité
    - Authentification 802.1X (EAP)
    - Dérivation et distribution des clés
    - Chiffrement et intégrité (AES, SHA-1)

Bluetooth et les téléphones portables

Bluetooth : introduction

- Protocole de communications sans-fil courtes distances
- déployé depuis plusieurs années dans téléphones mobiles, ordinateurs portables, GPS, routeurs, imprimantes, appareils photos, etc.
- Il fonctionne dans la gamme des 2.4GHz tout comme 802.11.
- 79 canaux Bluetooth





http://www.secuobs.com/news/05022006-bluetooth1.shtml

Bluetooth : réseau

- Un périphérique Bluetooth maître peut communiquer avec 7 autres périphériques esclaves au maximum.
- =>"réseau" = Piconet
- Scatternet
  - = interconnexion de Piconets, grâce à des "routeurs".
  - Nb max de Piconets = 10

Bluetooth : sécurité (1/4)

- PB actuel : augmentation de la distance de communication.
  - Exemple : Se rendre dans un lieu relativement fréquenté avec un scanner Bluetooth

    => risques encourus d'avoir un équipement Bluetooth activé dans sa poche.

  - Pb possibles par compromission Bluetooth : déni de service, récupération du carnet d'adresses, consultation des derniers appels, lecture des SMS,…
  - Etablissement d'un appel ou d'une connexion Internet.

Bluetooth : sécurité (2/4)

- 3 modes de sécurité Bluetooth :
  - Mode 1 : Pas de mécanisme de sécurité
  - Mode 2 : Sécurité assurée au niveau applicatif
  - Mode 3 : Sécurité assurée au niveau liaison de données

- Un équipement Bluetooth est caractérisé principalement par :
  - Son adresse BT (BD_ADDR), = une adresse MAC
  - Code PIN = optionnel
  - Sa classe, propre au type de périphérique : oreillette Bluetooth, téléphone mobile
  - Presque toute la sécurité = implémentation matérielle.
  - Interface avec l'OS = via le protocole HCI

Bluetooth : sécurité (3/4)

- Sécurité dans la couche gestionnaire de liaisons (GL)
- Système de gestion de clés propriétaires
- Assure :
  - l'authentification,
  - le pairage,
  - la création et la modification des clés,
  - Le chiffrement

Bluetooth : sécurité (4/4)

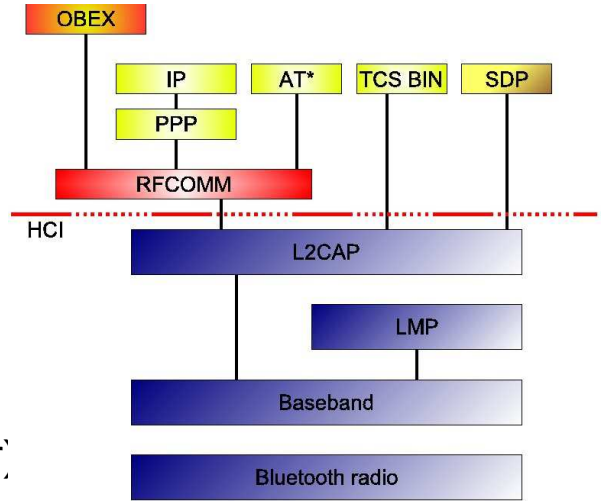
- Algorithmes de cryptographie utilisés :
  - E0 : chiffrement à flot => sert au chiffrement
  - Génération des clés à l'aide de trois algorithmes : E22, E21 et E3
  - Authentification : algorithme E3
  - Intégrité : algorithme SAFER+ (chiffrement par blocs)

  - Les problèmes cryptographiques :
    - attaque contre le chiffrement E0 et le mode particulier utilisé dans Bluetooth en 2^{40} avec 2^{35} frames
    - Si pas de code PIN, pas d'authentification de l'utilisateur
    - Le code PIN doit être partagé !
    - Problème de longueur de clés fonction de la législation des pays

Bluetooth : Les couches (1/2)

- **HCI** : assure l'abstraction matérielle
  - = Interface entre l'OS et le firmware Bluetoothe
  - Gère notamment la découverte des équipements distants

- **L2CAP** = équivalent d'un protocole d'accès au média
  - propre à Bluetooth
  - multiplexe des protocoles de couches supérieures (RFCOMM par exemple)
  - via des canaux appelés PSM (Protocol/Service Multiplexer)
  - Exemple : RFCOMM (de type liaison série Bluetooth) utilise PSM 3, SDP (Service Discovery Protocol) PSM 1
  - Equivalent de TCP/UDP



  - difficile de s'assurer pas de porte dérobée
  - L'outil psm_scan permet de lister les PSM accessibles

Bluetooth : Les couches (2/2)

- **SDP : liste les services disponibles sur un périphérique et les informations relatives :**
  - **PSM/Ports RFCOMM**
  - **=> possible d'utiliser un service sans repertorié par le serveur SDP distant.**

- **RFCOMM** : effectue des communications de type RS232 (série) sur L2CAP en Bluetooth

- De nombreux équipements communiquent via RFCOMM :
  - selon l'implémentation de la pile Bluetooth, et le port RFCOMM,
  - une authentification de type pairing peut être requise.

- **OBEX** = protocole d'échange d'objets
  - entrées de calendriers, de carnets d'adresses
  - simples fichiers.
  - => possible d'envoyer (commande PUSH), ou de recevoir (commande PULL) des données
  - OBEX fonctionne au dessus de RFCOMM

Bluetooth : différents outils

- **Hcitool** : détection de périphérique bluetooth
- **l2ping** : permet d'envoyer à un périphérique des paquets de niveau L2CAP de type ECHO REQUEST, à la manière d'un ping
- **sdptool** : permet d'effectuer différentes opérations au niveau L2CAP
  - requêtes directes vers les périphériques distants
  - configuration du serveur sdpd (ajout/suppression de services par exemple) sur la machine locale afin de répondre aux requêtes SDP entrantes.
- **rfcomm** : permet d'établir une communication de type RFCOMM entre 2 périphériques. => demande de création pour pairing

Bluetooth : quelques attaques (1/4)

- Détection de périphériques en mode non détectable (pas de réponse à des broadcasts)
  - "attaque" de bruteforce sur l'adresse Bluetooth
    - Plage d'adresse sur 6 octets (de 00:00:00:00:00:00 à FF:FF:FF:FF:FF:FF)
    - => Tout = 11 heures
    - Mais attaques sur des plages plus petites (plages de constructeurs,…)
    - => scan limité alors à 3 octets => une heure
    - Exemple d'outils : redfang, btscanner (+ fonctionnalités)
- Attaque sur le pairing = Processus qui autorise la connection d'un périphérique à un service local
  - Demande explicite sur le terminal distant par avertissement
  - Ou aucune demande => cas des codes PIN Bluetooth prédéfinis par le constructeurs (PIN par défaut 0000, 1234,…) => attaque simple !

Bluetooth : quelques attaques (2/4)

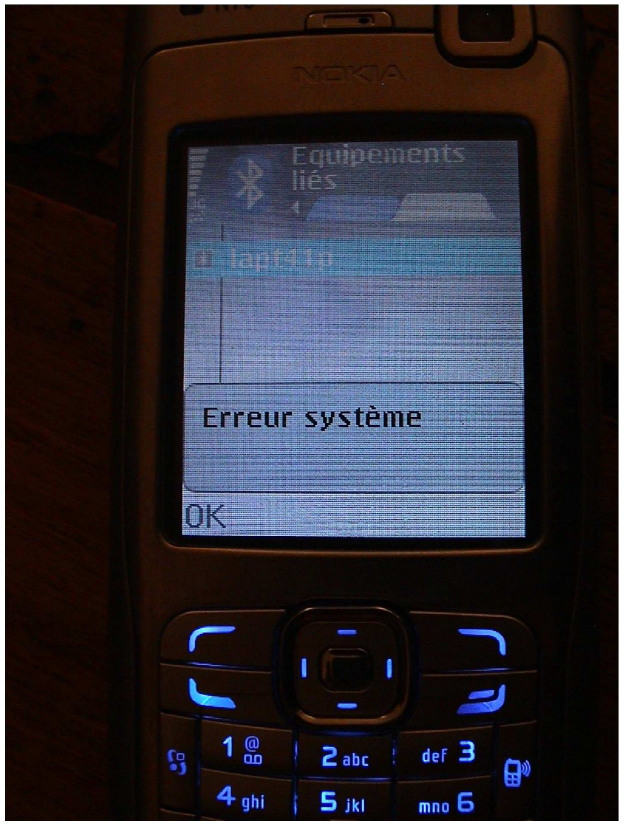
- Le pb du multi-pairing
  - = autoriser plusieurs terminaux à se connecter sur le même service
  - Outil : carwhisperer – ecoute du flux radio sur oreillettes + injection via le bon port RFCOMM
- BlueBug = la faille la plus lourde
  - Connection sur un port RFCOMM sans pairing
  - Accès à un ensemble de commandes particuliers
  - Outil : rfcomm_scan + commande AT btxml (carnet adresse)
  - Contrôle quasi-intégrale du téléphone !
    - => compromission de données, appels,…

Bluetooth : quelques attaques (3/4)

- Helomoto : outil pour attaquer les téléphones Motorola
  - BlueJacking : envoi d'un objet OBEX puis interruption
  - Attaquant devient 1 périphérique de confiance
  - Puis connection RFCOMM sur RFCOMM Headset sans authentification
  - Puis BlueBug…
- BlueSmack : blocage des périph. (crash pile ou SE) via requête l2ping trop longue ou bcp de requêtes
  - Exple : PDA iPaq
- D'autres : BlueSnarf,…=> Audit : Bloover II, BTBrowser,…

Bluetooth : quelques attaques (4/4)

- Exemple de www.secuobs.com : crash de la pile Bluetooth d'un NOKIA N70 :



Cryptographie à clé publique

Cryptographie à clé publique

- Pour chiffrer un message, Alice utilise la clé publique de Bob et seul lui peut déchiffrer le message à l'aide de sa clé secrète

- Je ne donnerai pas ici les preuves permettant de garantir ces algorithmes

Plan

- Principaux systèmes de chiffrement
  - RSA
    - Les pièges à éviter
    - Records de factorisation de nombre RSA
  - ElGamal
- Shémas de signature
  - RSA, RSA-PSS, ElGamal, DSS, DSA, EC-DSA

Systèmes de chiffrement

Problèmes mathématiques

- Deux grands problèmes
  - La factorisation de grands nombres
  - Le problème du logarithme discret

Rappel

- Le modulo :
  - a = b mod n ⇔ a+k.n = b+k'.n
  - L'ensemble des éléments 0,…n-1 défini par la relation modulo se note **Z/nZ**
  - Z/nZ est un anneau et un corps si n premier.

- $\phi(n)$: Fonction indicatrice d'Euler = nombre de nombre premier avec n.
  - Si n premier : $\phi(n) = n-1$
  - $\phi(pq) = \phi(p)\phi(q)$ si p et q premier

- Le problème difficile sur lequel repose RSA : la factorisation
  - Il est très difficile de trouver p et q / n=p.q en ne connaissant que n

RSA naïf (RFC 2437)

- Alice fabrique sa clé
  - n=pq avec p et q deux grands nombres premiers
  - e premier avec $\phi(n) = (p-1)(q-1)$ et d tel que $ed = 1 \bmod (p-1)(q-1)$
  - Rend publique (n,e)
- Bob veut envoyer un message m à Alice :
  - Bob calcule $c = m^e \bmod n$
  - Bob transmet c à Alice
- Alice déchiffre c en calculant :
  - $c^d = m^{ed} = m^1 \bmod n$

Principes de construction du RSA

- Connaissant n retrouver p et q => problème difficile (pas d'algorithme en temps polynomiale)
- Factoriser n ⇔retrouver d ⇔Inverser x^e mod n
- Il existe une infinité de nombres premiers
  - On sait en construire (Fermat, Carmichael)
  - On sait tester si ils sont premiers (Miller-Rabin)

Taille des clés RSA :

- Aujourd'hui, factorisation de clés RSA (=n) de plus de 512 bits (154 chiffres décimaux)

- Taille minimum préconisé :
  - Au moins 768 bits
  - 1024 bits conseillé

Principes de précaution pour RSA

- p et q doivent être grand ($\simeq$ 100 chiffres décimaux)
- p-q doit être grand (méthode de factorisation de Fermat)
- $p \pm 1$ et $q \pm 1$ doivent avoir un grand facteur premier chacun ($\simeq$ 100 bits)
- D’autres conditions,…

Car

- On a des algorithmes pour faciliter la factorisation des grands nombres
  - Méthode de Fermat
  - Crible quadratique, sur corps premiers,…
  - Méthode « rho » de Pollard,…

Factorisation des nombres RSA

Record de Factorisation depuis 1970

années	70	83	86	89	90	93	96	99	03
Nombre de décimaux	39	50	80	100	116	120	130	155	160

- **Nouveau record en 2005 : RSA-200 digits (663 bits)**

RSA-200 =
27997833911221327870829467638722601621070446786955428537560009929326128400107609345671052955360856061822351910951365788637105954482006576775098580557613579098734950144178863178946295187237869221823983

=

35324619344027701212726049781984643686711974001976250236493034687761212536794232000585479565280883 49

X

79258699544783330333470858414800596877379758573642199607343303414557678728181521353814093047401854 67

Chausse-trappe

- Même message avec l'exposant public 3 vers trois destinataires :
  - $c_1 = m^3 \bmod n_1$
  - $c_2 = m^3 \bmod n_2$
  - $c_3 = m^3 \bmod n_3$

=>Calcul de m^3 par calcul de la racine cubique modulo $n_1 n_2 n_3$

Principe de El Gamal

- Repose sur le problème du log discret :
  - Soit p un grand nombre premier et g une racine primitive modulo p, il s'agit de retrouver a connaissant A et g /

$$g^a = A \bmod p \text{ avec } 0 \leq a \leq p-2$$

- Aussi difficile que la factorisation

Le cryptosystème El Gamal

- On choisit p premier (public) et g (public)
- La clé publique d'Alice est $y=g^x$ / clé secrète x
- Bob veut envoyer un message m à Alice :
  - Il tire un aléa r
  - Calcule y^r
  - Transmet ($A=my^r$, $B=g^r$)
- Alice déchiffre
  - $B^x = g^{xr} = (g^x)^r = y^r$
  - Calcule $A(y^r)^{-1} = m$

Recommandations

- Ne pas utiliser deux fois le même nombre aléatoire r
- p-1 doit avoir un grand facteur premier

- p doit être grand (pareil que pour RSA)
  - > 512 bits
  - On recommande 768 ou 1024 bits

Record de calcul de log discret

Thursday, September 22nd, 2005.

We are pleased to announce a new record for the discrete logarithm problem over GF(2^n). Using the function field sieve of Adleman [Ad94], we were able to compute discrete logarithms for **607 bits and 613 bits** prime. The first computation gives an interesting comparison between the function field sieve and Coppersmith's algorithm since the same field finite was already addressed by Thome using the later algorithm.

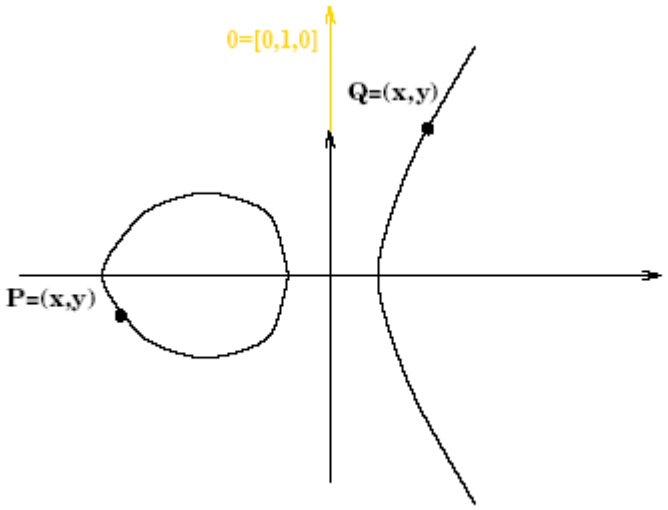
The two computations were done using different computers. For the first one, we used a **single 1.15GHz 16-processors HP AlphaServer** GS1280 computer **during one month**. For the second one, we used **four 16-processors nodes of the itanium 2 based Bull computer Teranova during 17 days (1.3GHz CPUs).**

Autres cryptosystèmes à clé publique

- Cryptosystèmes basés sur les codes correcteurs (Mac Eliece)
- Cryptosystèmes utilisant les courbes elliptiques
  - Courbes définies par :

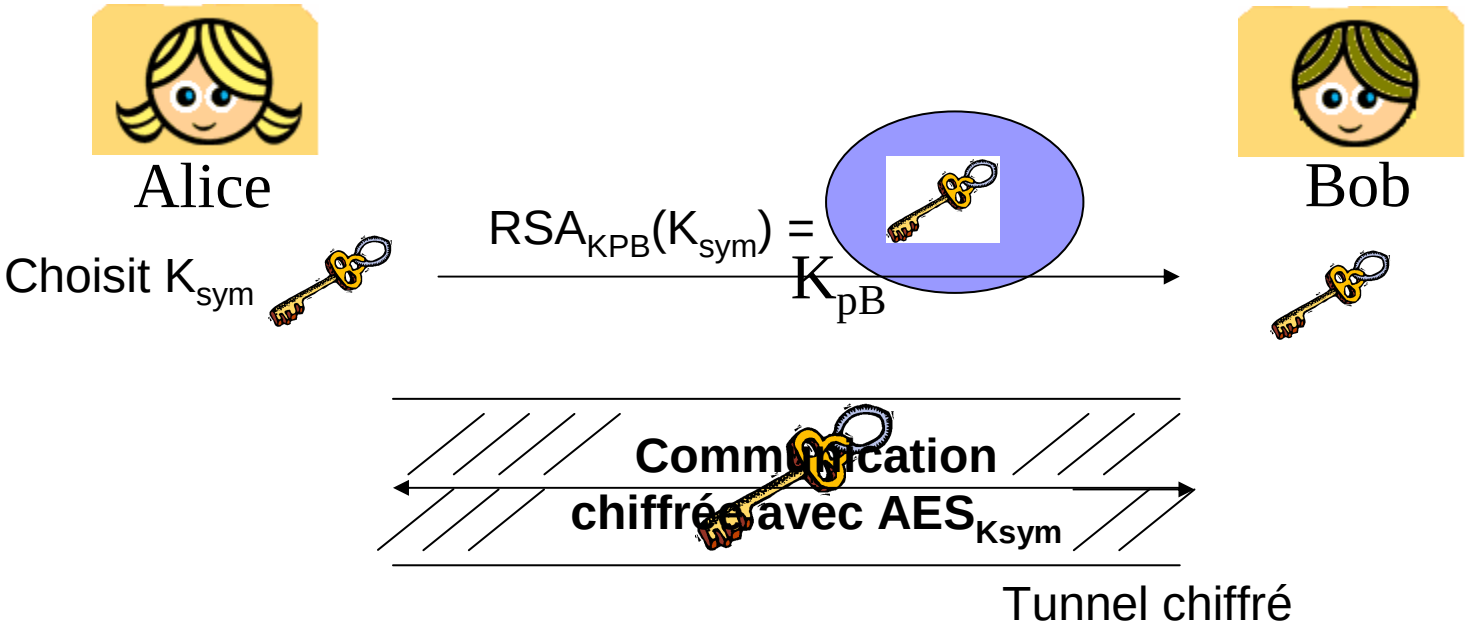
$P=(x,y) / y^2=x^3 -27\ c_4x-54c_6$

(courbe de Weierstrass)



Protocoles hybrides

- Cryptographie asymétrique pour transmettre des clés symétriques K_{sym}
- Cryptographie symétrique pour chiffrer



Ce qu’il reste à voir !

- Signature / authentification
- Identification
- Quelques protocoles
- La certification : comment garantir l’authentification

Principe de El Gamal

- Repose sur le problème du log discret :
  - Soit p un grand nombre premier et g une racine primitive modulo p, il s'agit de retrouver a connaissant A et g /

$$g^a = A \bmod p \text{ avec } 0 \leq a \leq p-2$$

- Aussi difficile que la factorisation

Le cryptosystème El Gamal

- On choisit p premier (public) et g (public)
- La clé publique d'Alice est $y=g^x$ / clé secrète x
- Bob veut envoyer un message m à Alice :
  - Il tire un aléa r
  - Calcule y^r
  - Transmet ($A=my^r$, $B=g^r$)
- Alice déchiffre
  - $B^x = g^{xr} = (g^x)^r = y^r$
  - Calcule $A(y^r)^{-1} = m$

Ce qu’il reste à voir !

- Signature / authentification
- Identification
- Quelques protocoles
- La certification : comment garantir l’authentification

Signatures

- Sur chacun des cryptosystèmes précédents, on peut construire des schémas de signatures

- En faisant évidemment attention !

Signatures : principe général

Alice

m

Annuaire clé publique

K_{pA}

Bob

Signature :

$Sign_{Alice}=Sign_{KsA}(m)$

$(m, Sign_{KsA}(m))$

Bob vérifie si

$m?=Sign^{-1}_{KpA}(Sign))$

- A cause de Charlie  qui pourrait changer m en m', on signe HASH(m) pour garantir l'intégrité du message

Propriétés d’une signature S

- S ne peut être contrefait
- S n’ai pas réutilisable
- Un message signé est inaltérable
- La signature S ne peut être renié

⇒ Sur support électronique, S doit dépendre du message M sinon copie et réemploi

- Signer n’est pas chiffrer !

Signature RSA

- Publique : n et e, Secret : exposant Alice d
- Alice signe le message m en calculant :

$$S = m^d \bmod n$$

- Bob vérifie en calculant : $m = S^e \bmod n$

- Performance : quelques centaines de signature par seconde

Problème ?

- Fraude existentielle : s aléatoire alors $m=s^e \bmod n$ => (m,s) couple (message,signature) valide !
- D'autres fraudes…

- Solution ajouter de la redondance (un condensé de m à la fin)

  => norme ISO-9796
- Mais pas encore sûr de sa solidité…

Signer M avec El Gamal

- Publique : p premier et g générateur
- Secret de Alice x et publie : $y=g^x \bmod p$
- Alice tire au hasard r
- Calcul de $a=g^r \bmod p$
- Calcul b / $M=ax+rb \bmod p$
- Transmission de (M,a,b)
- Bob vérifie que $y^a a^b=g^M \bmod p$

- Sûr mais Problème : très lent !

Signature sûre Digital Signature Scheme (anciennement DSA) (1/2)

- Public :
  - q premier (160 bits)
  - p=1 mod q (premier de 512 +64.t bits)
  - $g / g^q = 1 \bmod p$
- Alice :
  - secret : a
  - public : $A=g^a \bmod p$

Digital Signature Scheme (2/2)

- Alice choisit au hasard k
- Calcule $K=(g^k \bmod p) \bmod q$
- Calcule $s=(HASH(m)+aK)k^{-1} \bmod q$
- Transmet (m, K, s)

- Bob vérifie :
  - $1 \leq K,s \leq q$?
  - $(A^{K.s^{-1}} g^{HASH(m).s^{-1}} \bmod p) \bmod q \stackrel{?}{=} K$

- HASH = SHA1

Problème encore !

- Ce processus reste très lent pour un message long.
- C’est pour cela que dans la version présentée, on signe un haché du message m et pas le message dans son entier ! C’est ce qui se passe dans la vraie vie
- Il existe une version plus rapide de cette signature appelée EC-DSA qui utilise les courbes elliptiques.

Identification

- Vue dans le cas de la clé symétrique appelé protocole à une passe
- En cryptographie asymétrique, protocoles à deux passes ou plus

Protocoles par challenge (1/2)

- Par signature :

Alice

Bob

r

Choisit une aléa r

Alice calcule avec sa clé secrète $S=Sign_{KSA}(r)$

S

S signature valide de r ?

Vérification avec clé publique d'Alice

Protocoles par challenge (2/2)

- Par déchiffrement :

Alice

Bob

$C = Enc_{KPA}(r)$

Choisit un aléa r

Le chiffre avec la clé publique d'Alice

Alice déchiffre r

r' = Dec(C)

r'

r' ?= r

Protocoles sans divulgation de connaissances

- Un exemple : protocole de Shnorr (ElGamal)
  - Publique : p et q premiers / q | p-1, et g
  - Alice : secret a / Publique : $\underline{A = g^{-a} \bmod p}$

Alice

Bob

Choisit k et calcule

$K=g^k \bmod p$

K →

← Choisit un aléa r

r

Alice calcule

$y=k+ar \bmod q$

y →

$g^y A^r \stackrel{?}{=} K \pmod p$

D’autres protocoles de ce type

- Fondé sur RSA => Guillou-Quisquater
- Fiat-Shamir
- Okamoto

- => permet de créer des protocoles d’identification

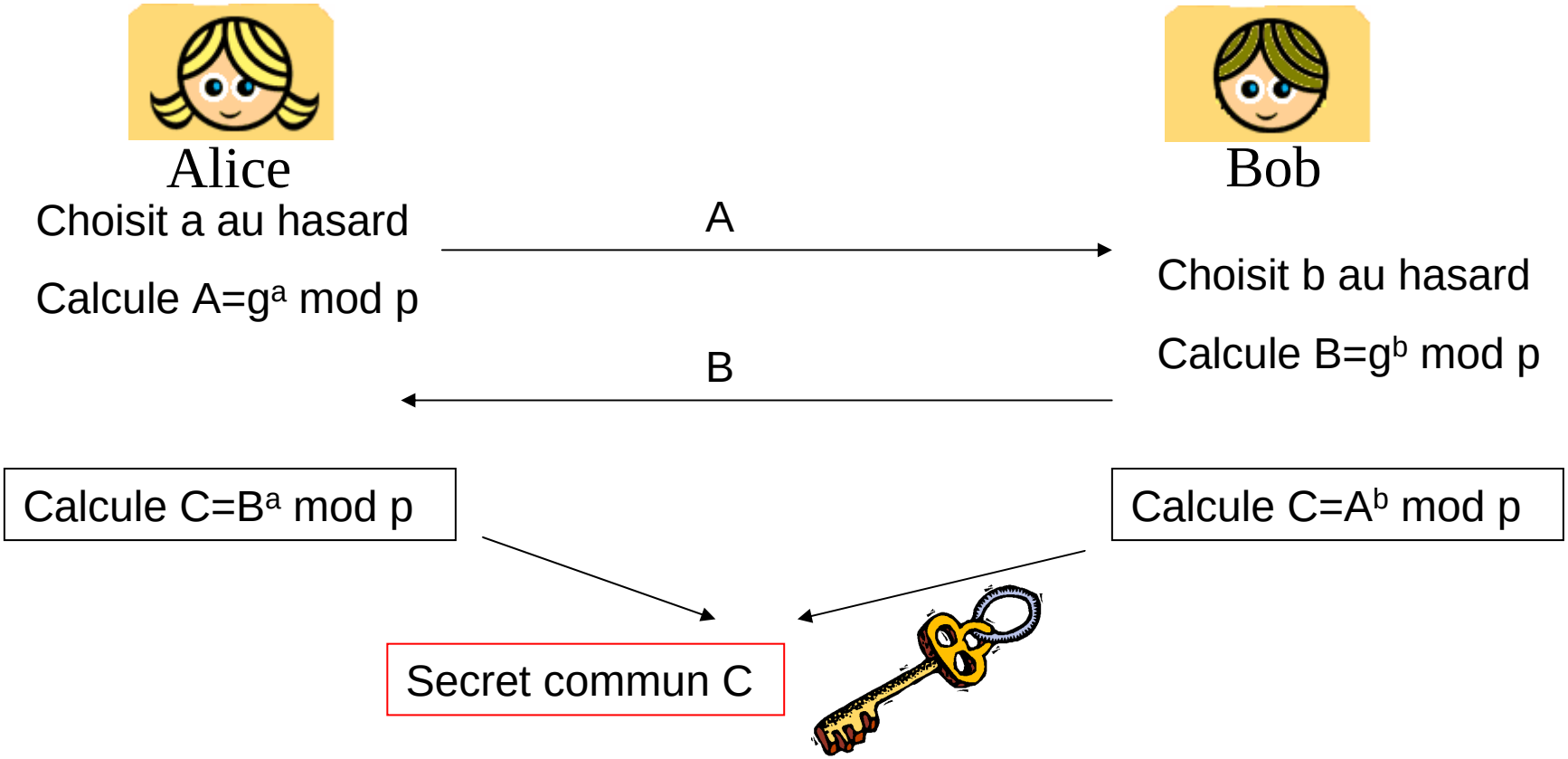
Protocoles d’échange de clés

- Le plus connu : Diffie Hellman qui permet de générer un secret commun (clé)

- Repose sur le problème suivant :
  - Si p et g sont publiques
  - Etant donné $A=g^x \bmod p$ et $B=g^y \bmod p$, x et y inconnus, calculer $g^{xy} \bmod p$.

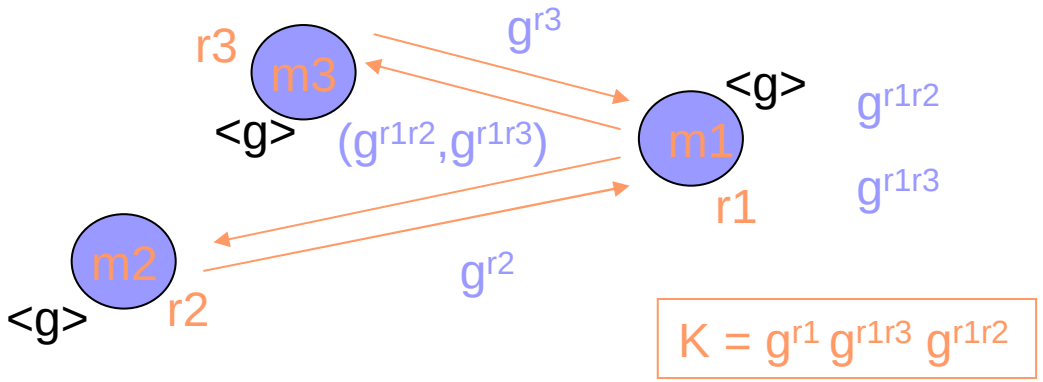
Protocole Diffie Hellman

- Publique : p premier, g racine primitive mod p



Protocole Diffie-Hellman (2/2)

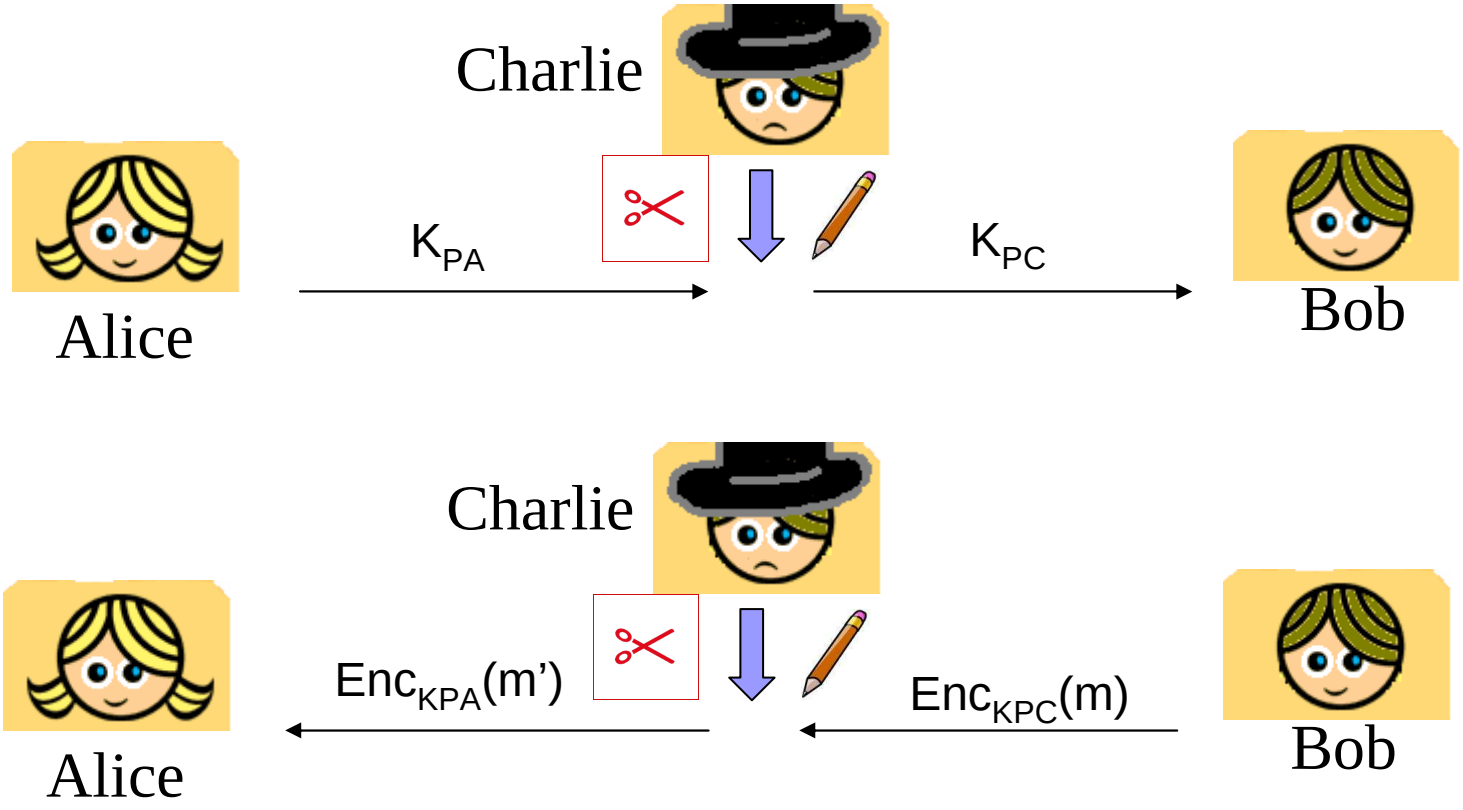
- Se généralise à plusieurs utilisateurs : création de clé de groupe



- Utilisation moderne : réseau ad hoc, peer to peer,…

La certification (1/2)

- Pourquoi a-t-on besoin d'une certification ?



La certification (2/2)

- Garantir que la clé publique d'Alice est bien la clé publique d'Alice

  => Garantir l'authentification

- Annuaire de clés publiques garanti par une autorité qui signe l'identité d'Alice et la clé publique de Alice

Certificats X.509 (1/2)

- Les certificats sont émis par des autorités de certification (CA)
- Le certificat d'Alice contient les champs suivants :
- $CA<A> = (SN, AI, I_{CA}, I_A, A_p, t_A, S_{CA}(SN, AI, I_{CA}, I_A, A_p, t_A))$;
  - SN : numéro de série
  - AI : identification de l'algorithme de signature
  - I_{CA}, I_A : « distinguished names » de CA et de Alice
  - A_p : clé publique de Alice
  - t_A : période de validité du certificat

Signature de l'Id d'Alice par le CA

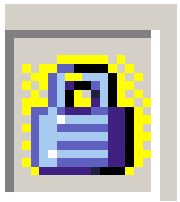
Certificats X.509 (2/2)

- La production du certificat d'Alice nécessite une communication sécurisée entre Alice et le CA

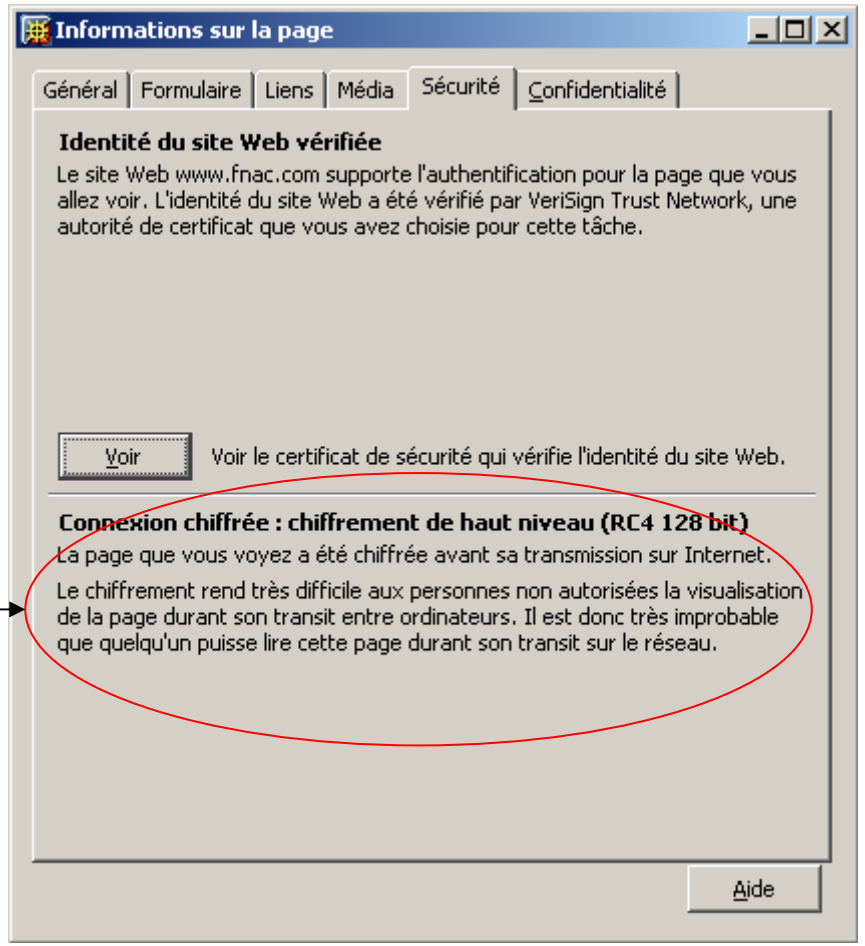
- Alice peut se présenter physiquement au CA

Exemple : Certificat X.509 de la Fnac (1/4)

- Présentation du certificat

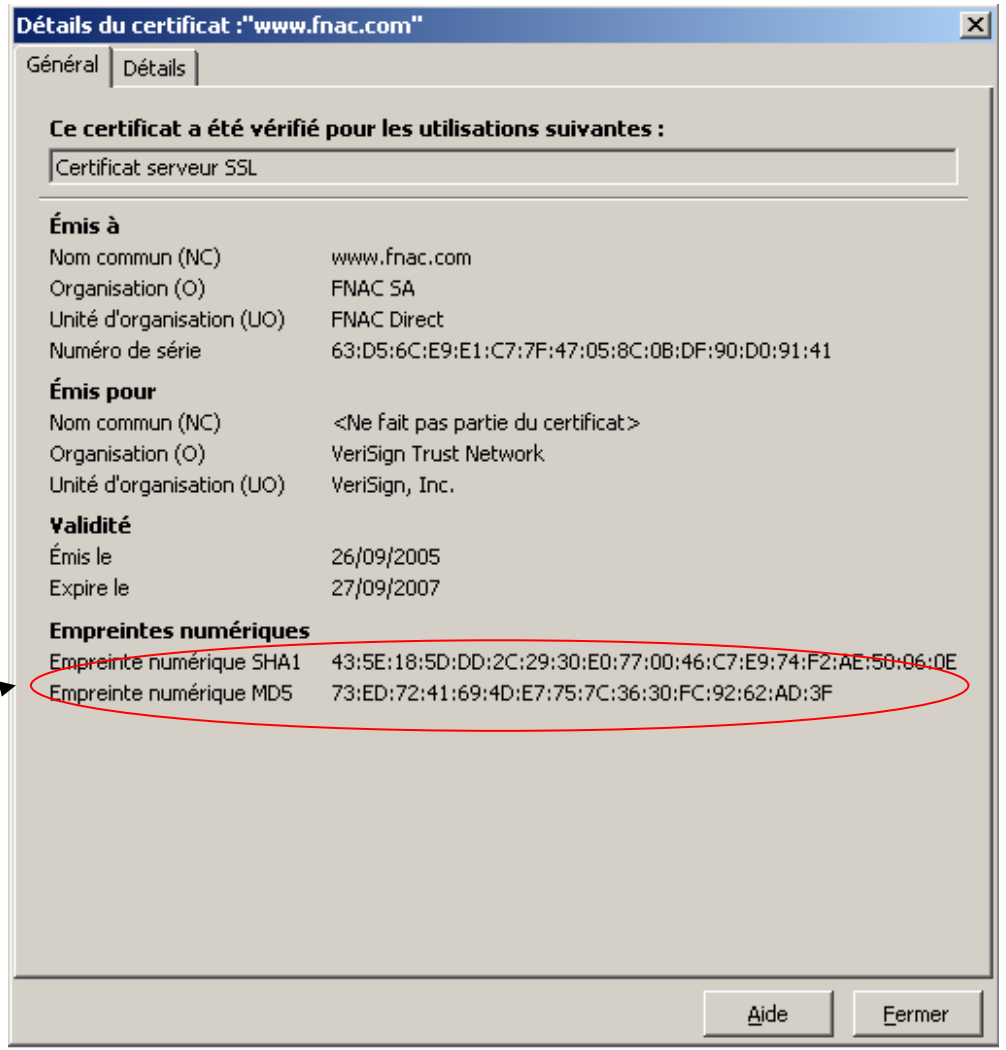


Construction d'une connexion chiffrée après vérification du certificat



Exemple : Certificat X.509 de la Fnac (2/4)

- Certificat lui-même



Valeur des hachés du certificat (vérification intégrité)

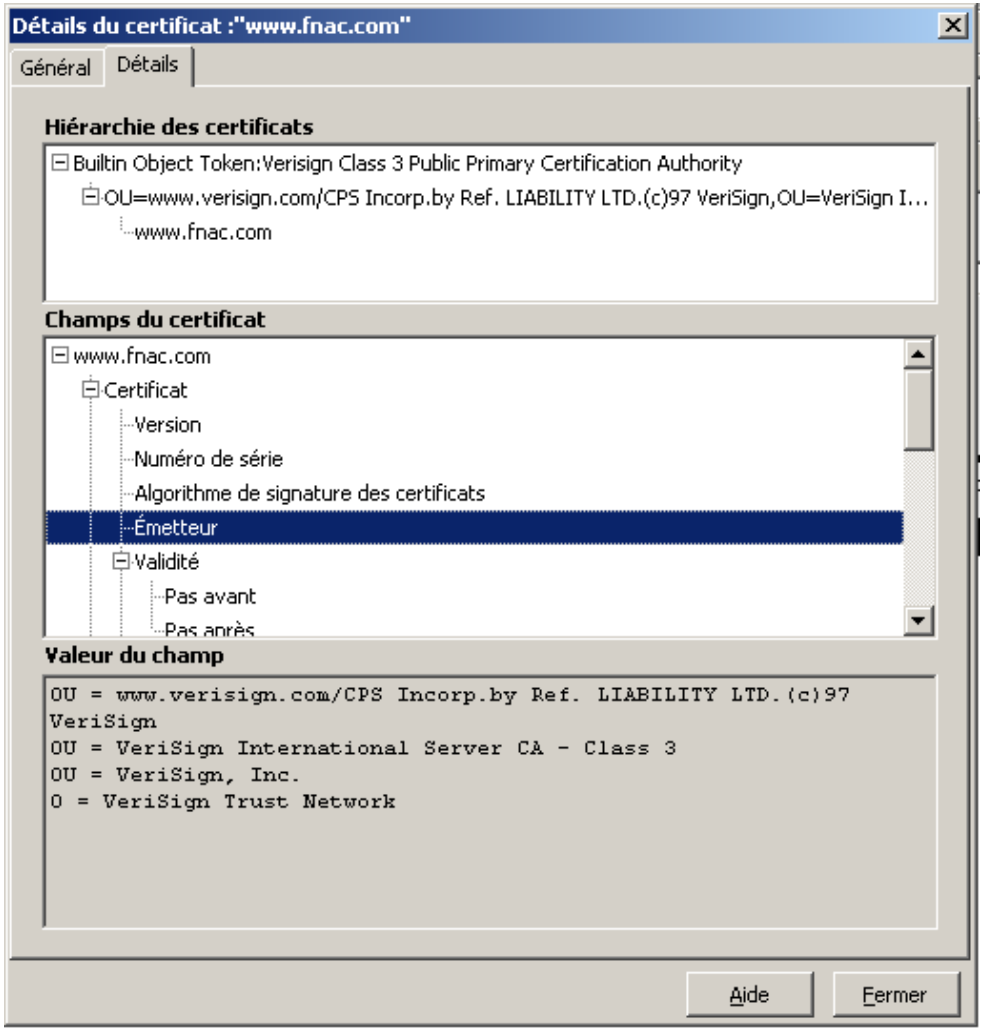
Exemple : Certificat X.509 de la Fnac (3/4)

- Information sur l'émetteur :

  Ici VeriSign

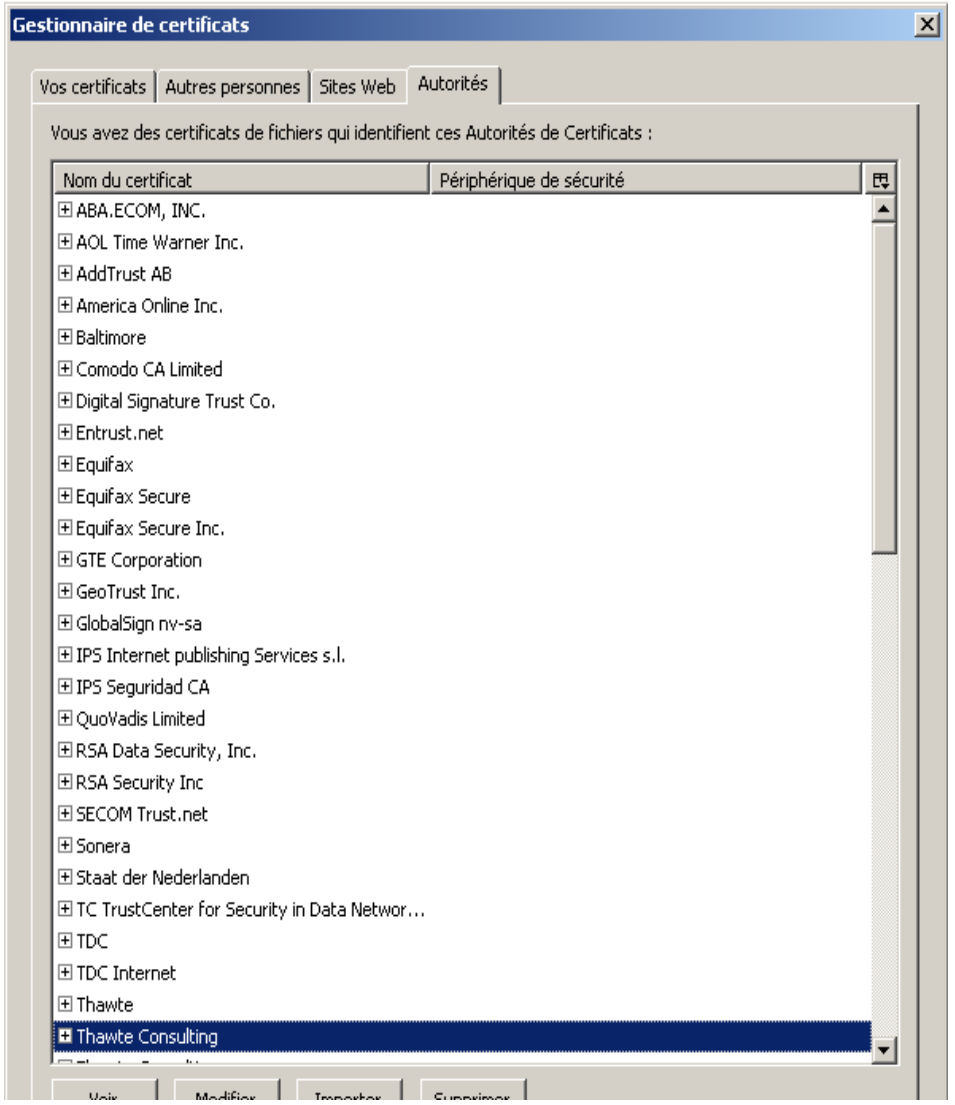
⇒ Tous les détails sont donnés ici :

  Clé publique,valeur de la signature,…



Exemple : Certificat X.509 de la Fnac (4/4)

- Liste des autorités de cerfications

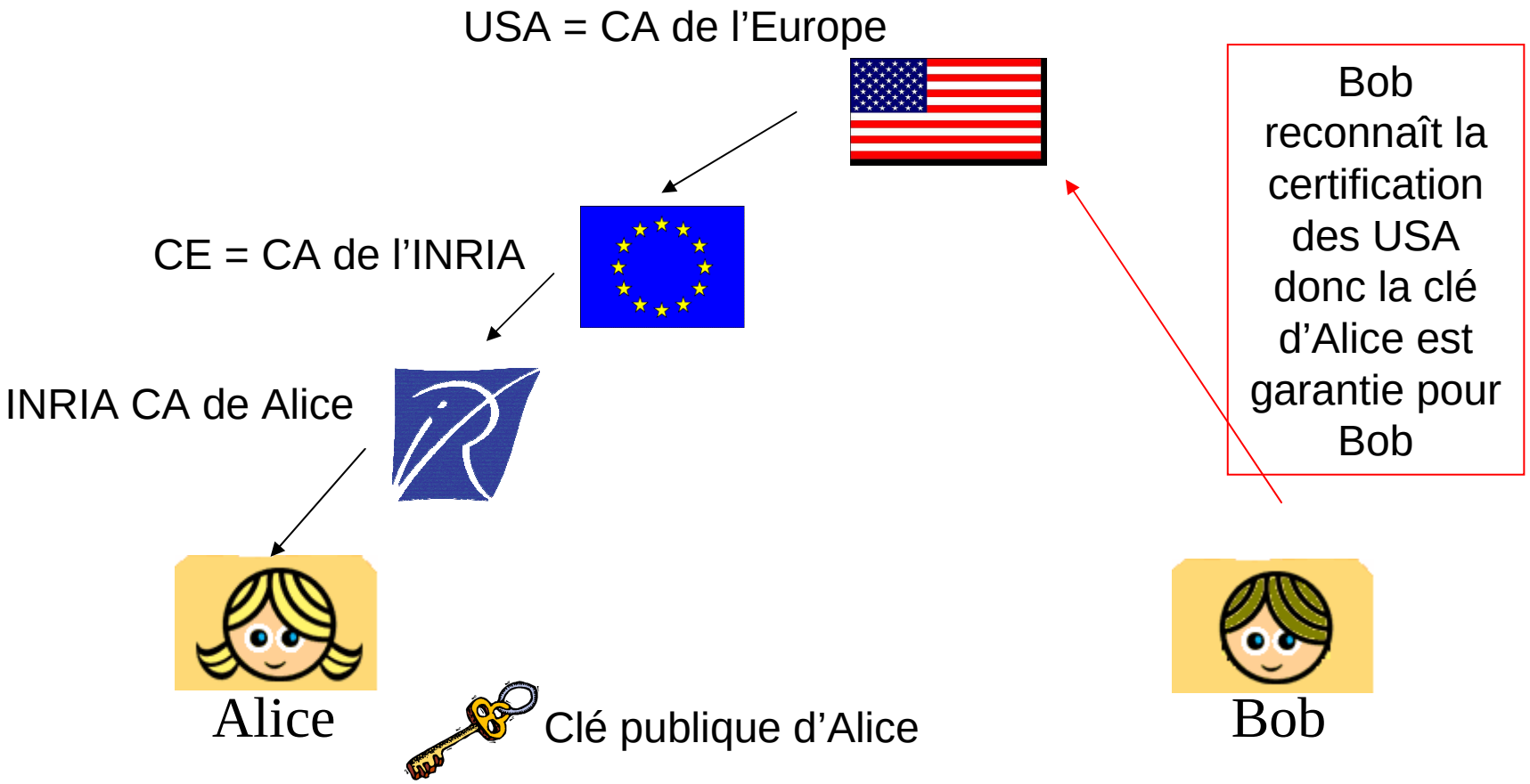


Chemin de certification (1/2)

- Pour être sûr de la clé publique d'Alice, Bob veut vérifier le certificat d'Alice qu'il a obtenu depuis LDAP (par exemple)
- On suppose que ce certificat a été produit par CA_1 inconnu de Bob
- Bob obtient pour CA_1 un certificat vérifié par CA_2,…
- Jusqu'à un CA reconnu par Bob

- Ceci = chemin de certification
- X509 v3 : autorise un chemin de certification de taille 10

Chemin de certification (2/2)



Conclusion partielle

- Clé publique lent mais permet de garantir
  - Chiffrement sans échange de clé préalable
  - La Signature
  - L’identification
  - L’authentification par certification
  - Permet d’échanger une clé symétrique partagée
- Clé symétrique
  - Rapide pour le chiffrement
  - Garantit l’intégrité (fonction de hachage)

Schéma habituel d'utilisation

Client

Alice vérifie auprès du CA l'identité de Bob

Authentification par certification

Envoie de (K_{PA}, $sign_{CA}(hash(K_{PA}))$

Serveur

Bob vérifie auprès du CA l'Identité d'Alice

Authentification par certification

Envoie de (K_{PB}, $sign_{CA}(hash(K_{PB}))$

Choisit un algo symétrique et une clé K

Envoie le choix de l'algo et la clé K chiffrée avec K_{PA} et signé avec K_{SB}

($chiff_{KPA}(K)$, $sign_{KSB}(hash\ chiff_{KPA}(K))$)

Tunnel sécurisé

IPSec et VPN

IP-Sec : introduction

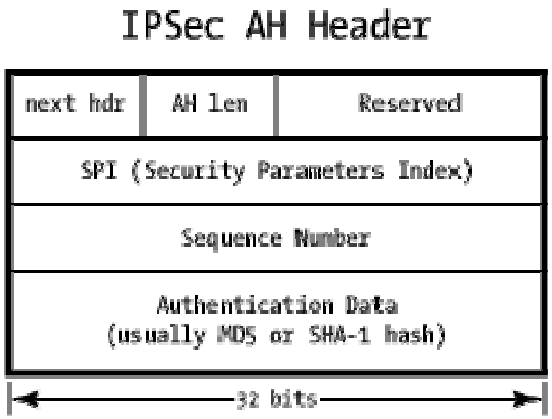
- Internet Security protocol, intégré à IPv6
- Objectifs : sécuriser les trames IP :
  - Confidentialité des données et protection partielle contre l'analyse du trafic
  - Authentification des données et contrôle d'accès continu
  - Protection contre le rejeu
- Principe :
  - ajout de champs d’authentification dans l’en-tête IP
  - chiffrement des données
- Avantage : sécurisation niveau réseau
- Inconvénients : coût, interfaces avec les autres protocoles à standardiser

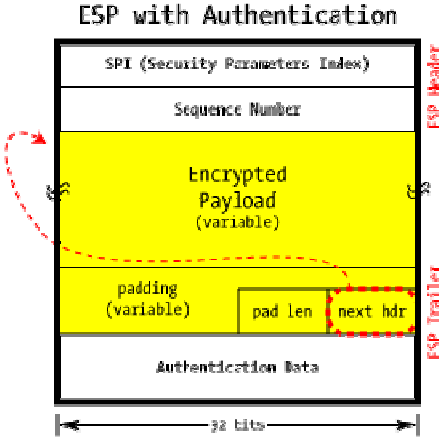
IP-Sec : les algos utilisés

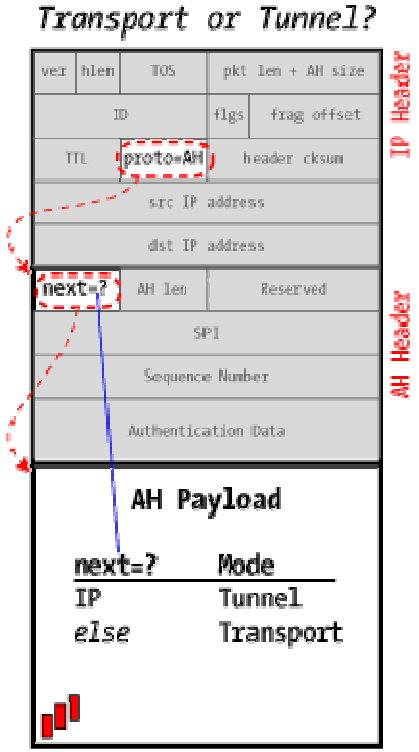
- IP-Sec s'appuie sur différents protocoles et algorithmes en fonction du niveau de sécurité souhaité :
  - **Authentification** par signature électronique à clé publique (RSA).
  - **Contrôle de l'intégrité** par fonction de hachage (MD5).
  - **Confidentialité** par l'intermédiaire d'algorithmes symétriques, tels que DES, 3DES ou IDEA.

IP-Sec

- Fonctionne avec deux protocoles possibles :
- AH (juste authentificat°) ESP (chiffrement)



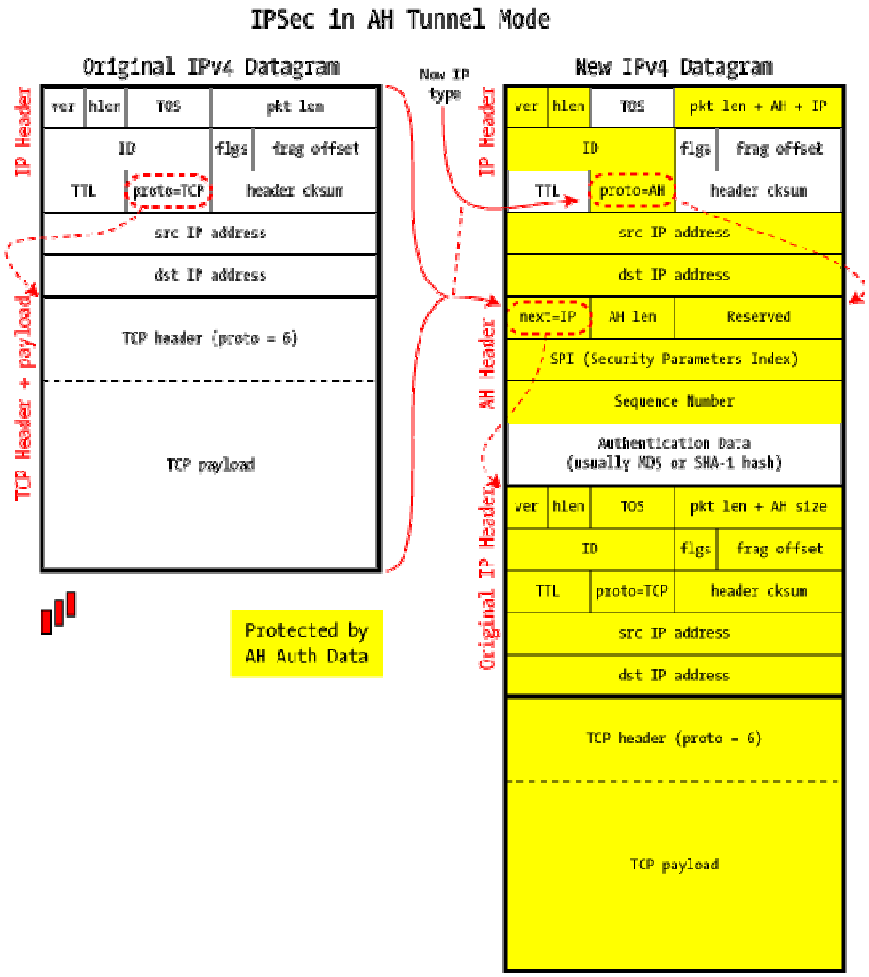




  - Deux modes possibles d'utilisation avec les deux :
    - Transport
    - Tunnel

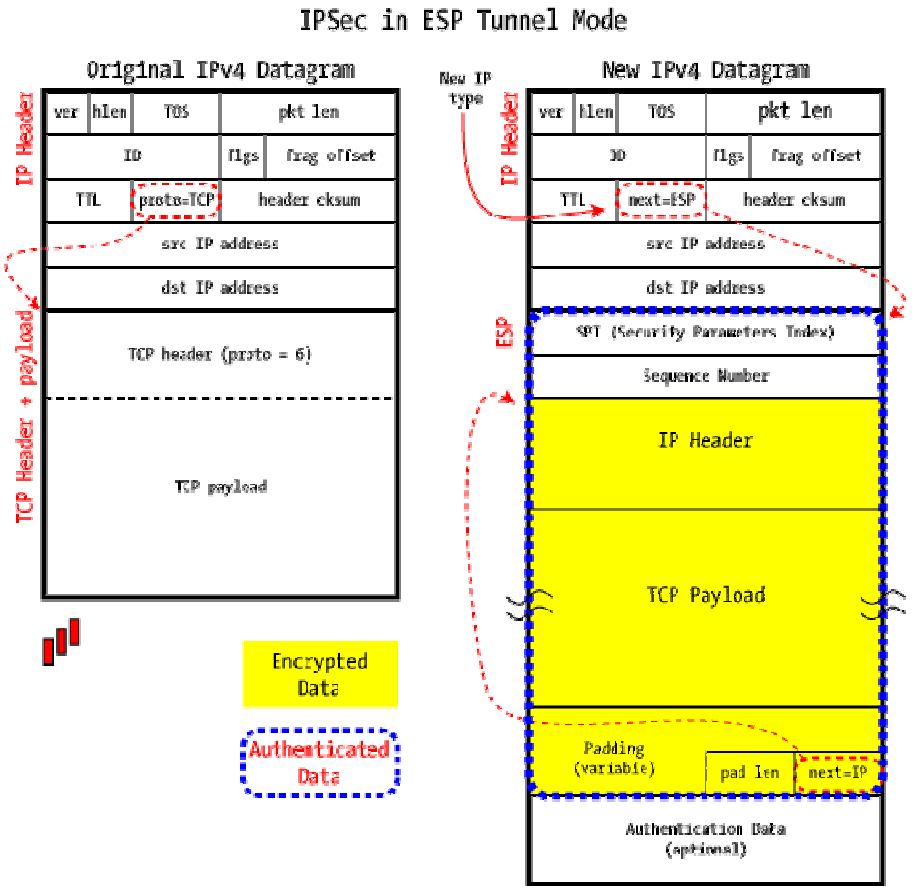
IP-Sec : mode tunnel avec AH

- Avec juste AH (authentification)



IP-Sec : mode tunnel avec ESP

- Avec ESP (chiffrement des données)



IP-Sec : échange de clés

- Utilisation de IKE : Internet Key Exchange
  - Permet à deux points donnés de définir leur « association » de sécurité (algorithmes,…) ainsi que les clés et les secrets qui seront utilisés.
  - utilise ISAKMP (Internet Security Association Key Management Protocol)

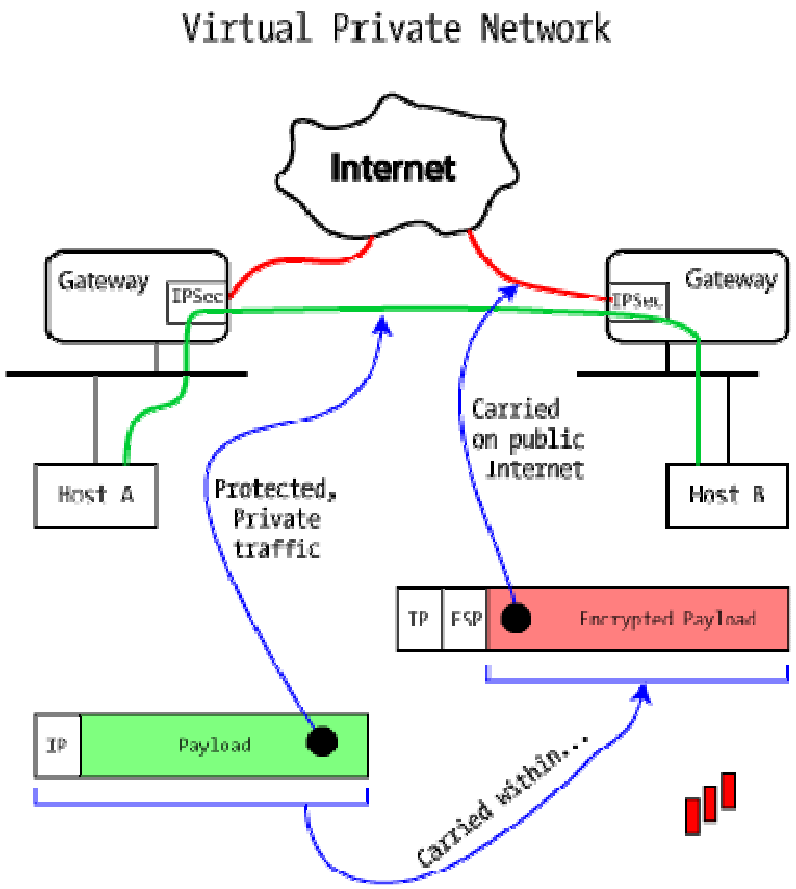
IP-Sec : les problèmes

- Le rapport « charge totale/ charge utile » augmente.

- Coût en terme de temps supplémentaire engendré par tous les calculs que nécessite
  - MD5 (hachage pour l'intégrité)
  - 3DES (algorithme symétrique pour confidentialité)
  - RSA (authentification par signature à clé publique)

Les VPNs : Virtual private networks

- Interconnection par tunnel de LAN disséminés
- Mobilité des utilisateurs
  - Les utilisateurs peuvent se connecter par modem et accéder au VPN qui leur alloue une adresse IP
- Types de VLAN
  - ensemble de ports/segments
  - ensemble d'adresses MAC (niveau 2)
  - sous-réseau protocolaire (IP) (niveau 3)
  - réseau fondé sur des règles



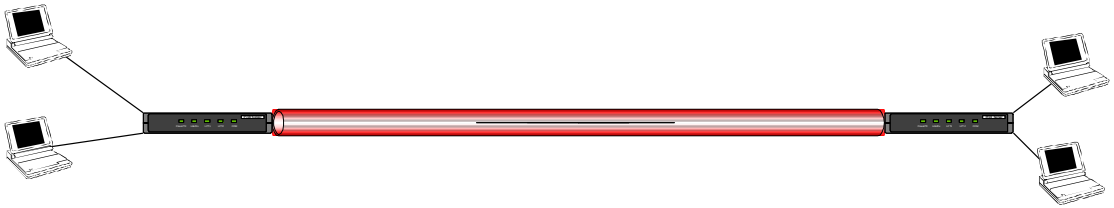
VPNs : autres avantages

- **IP-Sec** est à ce jour le protocole le plus utilisé dans les VPNs avec PPTP (Point to point Tunneling Protocol)
- Les paquets sont chiffrés quand ils quittent un LAN et déchiffrer quand ils entrent dans un autre LAN
  - Garantie de sécurité et d'isolation
  - Chiffrement, intégrité, authentification
- Avantages
  - transparence
  - securité
  - coût
  - Accessible depuis internet

IPsec et VPN

- **IPSec mode transport:** En mode transport, la session IPSec est établie entre deux hosts
  - Avantage: la session est sécurisée de bout en bout
  - Inconvénient: nécessité d'une implémentation de IPSec sur tous les hosts; autant de sessions IPSec que de couples de hosts

- **IPSec mode tunnel:** En mode tunnel, la session IPSec est établie entre deux passerelles IPSec, ou un host et une passerelle
  - Avantage: l'ensemble des communications traversant les passerelles VPN peuvent être sécurisées; pas de modification des hosts
  - Inconvénient: nécessite des passerelles VPN



IPsec et VPNs : conclusion

- Aujourd'hui, l'utilisation d'un VPN est la manière la plus fiable de sécuriser un réseau wireless

  => C'est aussi la méthode la plus utilisée

- Mais il faut savoir que les performances vont diminuer (significativement) : Bande passante diminuée de 30% en moyenne.

- Tous les LANs doivent être sécurisés pour obtenir une sécurité globale

PKI

PKI : Public Key Infrastructure

- But : distribution de clé publique avec sécurité et gestion de certificats

- Principe général et fonction :
  - Enregistrement de demandes et de vérifications des critères pour l'attribution d'un certificat
    - Id du demandeur vérifier
    - Possession de la clé secrète associée
  - Création des certificats
  - Diffusion des certificats avec publication des clés publiques

PKI : Public Key Infrastructure

- Archivage des certificats pour suivi de sécurité et de pérennité
- Renouvellement des certificats
- Suspension de certificats (pas de standard, peu aisé à mettre en œuvre,surtout administrative)
- Révocation de certificat sur date, perte, vol ou compromission des clés
- Création et gestion des listes de révocation des certificats
- Délégation de pouvoir à d’autres entités reconnues de confiance

Principaux problèmes

- Suspension de certificats : pas de standard
- Création et publication des listes de révocation des certificats
  - Pas de standard pour révocation automatique
  - Moyens administratifs : implémentés de façon sécurisée
    - Le propriétaire de la clé doit prouver que sa clé est inutilisable

Problème de gestion !

- Listes de révocations doivent
  - Être protégées pour ne pas être corrompues
  - Être accessibles en permanence et à jour
  - => synchronisation des horloges de toutes les pers. concernées
- Listes de révocations peuvent être très grande
  - Exemple : paiement des impôts par Internet
  - Maintenues en permanence
  - Enorme base de données, accessible à tout instant !

À Titre de comparaison

- Comme une carte d’identité national
  - Preuve de l’identité quand création de la carte
  - Unique, liée à une identité et non falsifiable
  - Déclaration en préfecture quand vol ou perte afin d’en obtenir une autre et pour ne pas qu’il y est une mauvaise utilisation de la disparue

Conclusion PKI

- Important :
  - Étude de faisabilité préalable pour estimer
    - Besoins (matériels) selon le nombre de personnes concernées
    - La validation par des organismes de confiance
    - Le déploiement
- Un exemple : les impôts http://www.ir.dgi.minefi.gouv.fr/
- Plus d'informations : http://www.ssi.gouv.fr/

Les réseaux ad hoc