Sécurité des réseaux

Marine MINIER
INSA de Lyon

Plan du cours

- Introduction générale: menaces et historique
- Cryptographie
  - Cryptographie à clé publique
    - Principaux cryptosystèmes + protocoles
    - Signature + protocoles
  - Cryptographie à clé secrète ou symétrique
    - Description + Premier exemples d'applications
- Principes de certification : X.509
- Applications pratiques
  - OpenSSL, SSH et ses dérivés
  - PGP
  - https
  - IP-Sec et VPN
  - Cartes à puce
  - Les PKI
- Ce qu'on a pas vu : la sécurité système du type vers, virus, comment infecter un ordinateur
- Comment faire une bonne architecture réseau pour être le plus résistant possible
- Conclusion

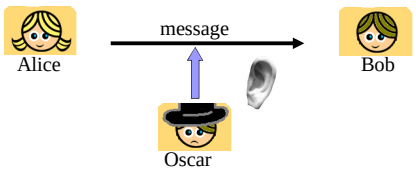
Bibliographie

- Crypto
  - D. Stinson
  - B. Schneier
- Sécu
  - Junod, Avoine
  - Gernaouthi
  - Le pujolle !

Menaces et historique

Menaces : utilité de la cryptographie

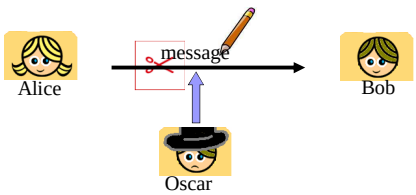
- Attaques passives



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

Menaces : utilité de la cryptographie

- 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

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

historique

Age archaïque

- IRAK XVIème avant JC :
  - potier : recette secrète sur une tablette d'argile : suppression des consonnes et modification de l'orthographe
- -600 : Nabuchodonosor (Babylone)
  - tatouage sur le cuir chevelu
- VIIeme avant JC : scytale
- Ier avant JC : chiffrement de César
  - Alphabet de César : décalage de trois positions des lettres de l'alphabet => CESAR -> FHVDU
- transposition, substitution (mono/poly-alphabétique, homophonique,...) - Vigenère

Age technique

- 2ème guerre mondiale: Enigma
  - Turing, Bletchley park
  - Guerre du pacifique: Indiens Navajo

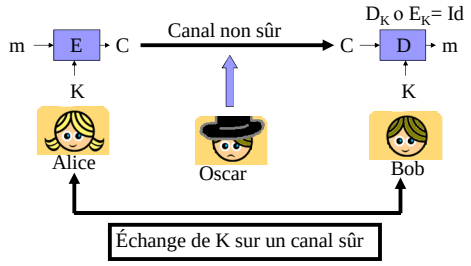
Age moderne

- 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)

Cryptographie

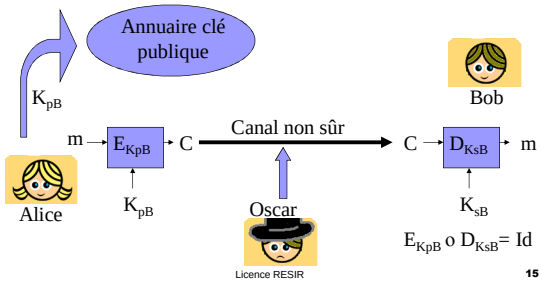
Deux méthodes pour chiffrer l'information (1/2)

- Cryptographie à clé secrète :



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

- Cryptographie à clé publique :



A quoi doit résister un bon algorithme de chiffrement ?

- Attaques de Oscar
  - 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 à clé publique

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 à sens unique
  - 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)

- Bob 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)
- Alice veut envoyer un message m à Bob :
  - Alice calcule $c = m^e \bmod n$
  - Allice transmet c à Bob
- Bob déchiffre c en calculant :
  - $c^d = m^{ed} = m^1 \bmod n$

RSA

- Un exemple !

Principes de construction du RSA

- Connaissant n retrouver p et q => problème difficile (pas d'algorithme en temps polynomiale)
- Factoriser n ⇔retrouver d ⇔Inverser $x^e \bmod n$
- 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) d'au moins 700 bits (plus de 220 chiffres décimaux)
- Taille minimum préconisé :
  - 1024 bits (~ 300 chiffres décimaux)

Principes de précaution pour RSA

- p et q doivent être grand (' 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 (' 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	05	12
Nombre de décimaux	39	50	80	100	116	120	130	155	160	200	212

- **Record de 2013 : RSA-210 digits (696 bits)**

```
RSA-210 = 245246644900278211976517663573088018467026787678332759743414451715061600830038587216952208399332071549103626827191679864079776723243005600592035631246561218465817904100131859299619933817012149335034875870551067

RSA-210 = 435958568325940791799951965387214406385470910265220196318705482144524085345275999740244625255428455944579
        × 562545761726884103756277007304447481743876944007510545104946851094548396577479473472146228550799322939273
```


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 /

    $g^a = A \bmod p$ 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 de Bob est $y=g^x$ / clé secrète x
- Alice veut envoyer un message m à Bob :
  - Il tire un aléa r
  - Calcule y^r
  - Transmet ($A=my^r$, $B=g^r$)
- Bob déchiffre
  - $B^x = g^{xr} = (g^x)^r = y^r$
  - Calcule $A(y^r)^{-1} = m$

El Gamal

- Un exemple !

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

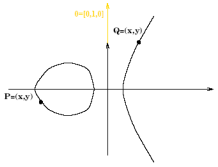
Record de calcul de log discret

On 25 June 2014, Razvan Barbulescu, Pierrick Gaudry, Aurore Guillevic, and François Morain announced a new computation of a discrete logarithm in a finite field whose order has 160 digits and is a degree 2 extension of a prime field.

The algorithm used was the number field sieve (NFS), with various modifications. The total computing time was equivalent to 68 days on one core of CPU (sieving) and 30 hours on a GPU (linear algebra).

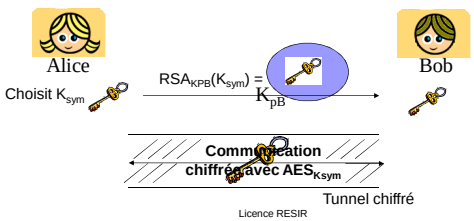
Autres cryptosystèmes à clé publique

- 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)

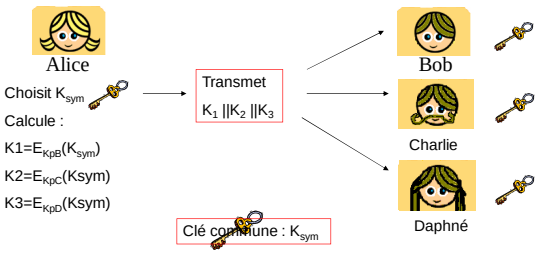


Protocoles hybrides

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



Se généralise à plusieurs destinataires



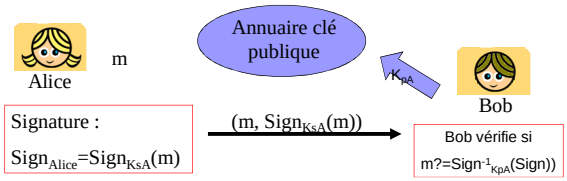
Ce qu'il reste à voir !

- 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



- A cause de Oscar  qui pourrait changer m en m', on signe HASH(m) pour garantir l'intégrité du message
- Exemple : DSS, DSA (fondé sur RSA),…

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 !

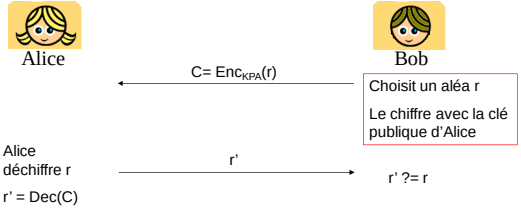
- Ce processus reste très lent pour un message long.
- C'est pour cela que dans la version présentée, on signe un haché du message m et pas le message dans son entier ! C'est ce qui se passe dans la vraie vie
- Il existe une version plus rapide de cette signature appelée EC-DSA qui utilise les courbes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

Authentification/Identification

- Authentification d'un document : par signature
- Identification d'une personne : par mot de passe
  - But : ne pas transmettre le mot de passe en clair !

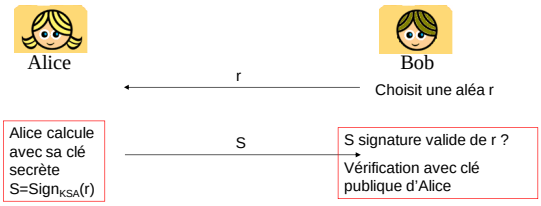
Identification par challenge : crypto à clé publique (1/2)

- Par déchiffrement :



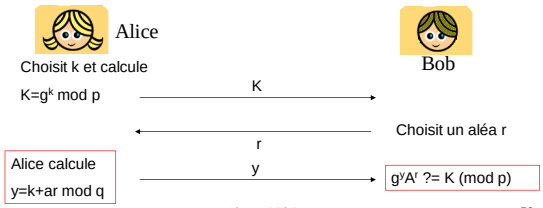
Protocoles par challenge (2/2)

- Par signature :



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 : $\underline{A = g^{-a} \bmod p}$



D'autres protocoles de ce type

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

- => permet de créer des protocoles d'identification : SRP en est un exemple

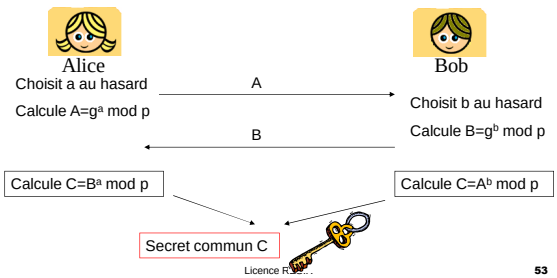
Protocoles d'échange de clés

- 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 mod p et B=g^y mod p, x et y inconnus, calculer g^{xy} mod p.

Protocole Diffie Hellman

- Publique : p premier, g racine primitive mod p

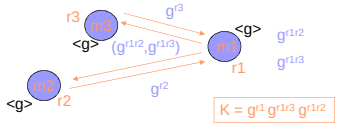


Diffie-Hellman

- Exemple !

Protocole Diffie-Hellman (2/2)

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



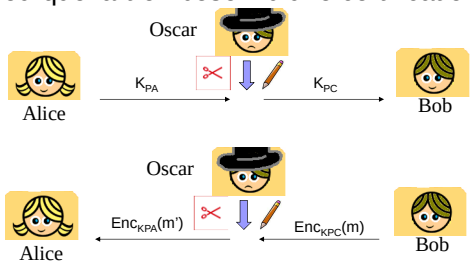
- Utilisation moderne : réseau ad hoc, peer to peer,…

Applications de la cryptographie

- Vient de voir
  - Principaux algorithmes en clé symétrique
  - Principaux algorithmes en clé asymétrique
  - Principaux protocoles
- S'intéresser à leurs utilisations

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

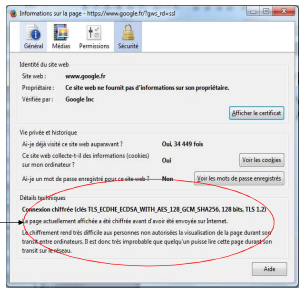
Certificats X.509 (2/2)

- 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 google.fr (1/4)

- Présentation du certificat

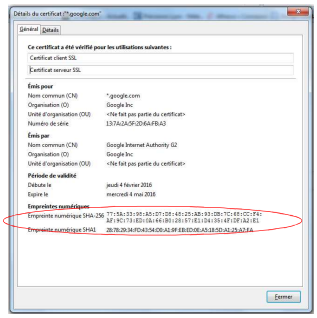
Construction d'une connexion chiffrée après vérification du certificat



Exemple : Certificat X.509 de google.fr (2/4)

- Certificat lui-même

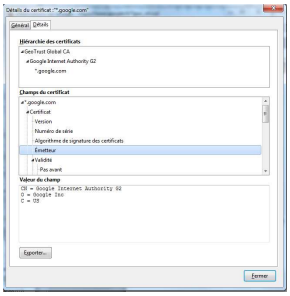
Valeur des hachés du certificat (vérification intégrité)



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

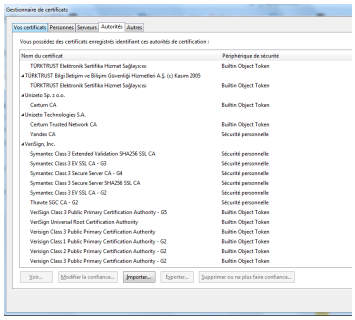
- Information sur l'émetteur :
  Ici VeriSign

⇒ Tous les détails sont donnés ici :
Clé publique,valeur de la signature,…



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

- Liste des autorités de cerfications

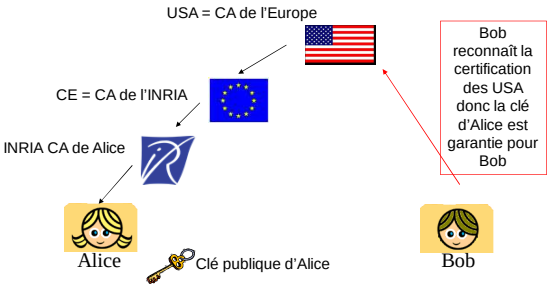


Chemin de certification (1/2)

- 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)

Cryptographie symétrique

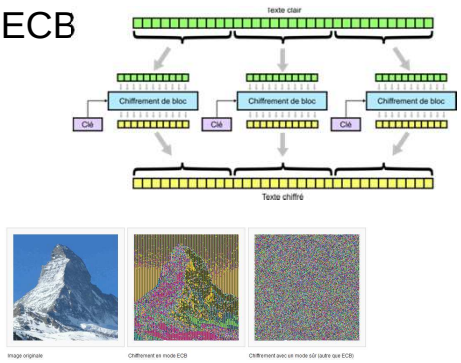
Cryptographie symétrique

- 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

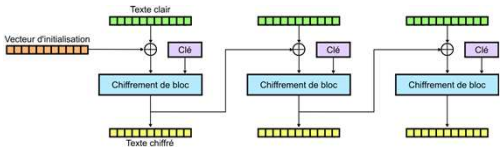
Algorithmes symétriques de chiffrement par blocs

- 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

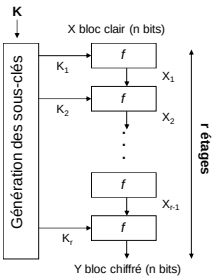


Mode CBC :



Chiffrement par blocs itératifs

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

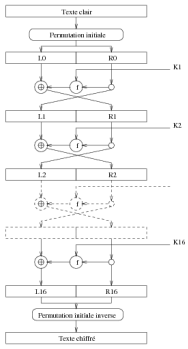


- La clé K est utilisée pour générer r sous-clés, une pour chaque étage.
- La fonction f est une permutation des blocs de n bits

Premier exemple : le DES

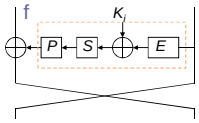
- DES = Data Encryption Standard
- Élaboré par le NBS (National Bureau of Standards, aujourd'hui NIST) à partir d'un algorithme d'IBM Lucifer.
- Standardisé en 1977

Le DES (1/3)



- Description : algorithme de chiffrement par blocs
  - Entrée : bloc de 64 bits
  - Sortie : bloc de 64 bits
  - Clé : 64 bits dont 56 sont utilisés (le dernier bit de chaque octet = bit de parité)
  - Algorithme entièrement symétrique : chiffrement = déchiffrement
  - Nombre de tours : 16 tours

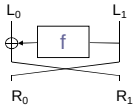
Le DES (2/3)



- Fonction d'étage f :
  - Schéma de Feistel
  - f :
    - E : expansion de 32 vers 48 bits
    - Xor avec une sous clé K_i de 48 bits
    - S : boites S (permutations non linéaires) de 48 bits vers 32 bits (8.6 vers 8.4 bits)
    - P : permutation de 32 bits vers 32 bits

Le DES (3/3)

- Le schéma de Feistel est une permutation :



  - $R_0 = L_1$ et $R_1 = L_0 \oplus f(L_1)$
  - Inverse : $L_1 = R_0$ et $L_0 = R_1 \oplus f(L_1)$

  - Exemple d'une des boites S (il y en a 8)

Entrée sur 6 bits : colonne : 4 bits du milieu

2.1er bit + dernier

S_1	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

Le DES attaqué ! (1/2)

- Loi de Moore : « le nombre de transistors des microprocesseurs sur une puce de silicium double tous les dix-huit mois. »

1983	1985	1987	1990	1993	1996	1999	2005
256 Kb/s	1 Mb/s	4 Mb/s	16 Mb/s	64 Mb/s	256 Mb/s	1 Gb/s	2 Gb/s

- Taille de la clé du DES : 56 bits soit 2^{56} essais !

Le DES attaqué ! (2/2)

- **99 :** Attaque des laboratoires RSA contre le DES (clé retrouvée en 22 heures)
  - À l'aide d'une machine dédiée : Deep Crack (250 000 dollars)
  - De 100 000 PCs par calul distribué
- **Changement de taille de clé : 128 bits minimum**
- **97 : Appel d'offre du NIST** pour choisir un nouvel algorithme de chiffrement par blocs pour le 21ème siècle
  - Nom : **AES**
  - 15 propositions, 5 finalistes
  - choix de **Rijndael** en **octobre 00**.

Mots de passe Unix (1/2)

- Utilisation d'un DES à 25 tours
- Mot de Passe (MP) = clé du DES pour chiffrer une valeur d'initialisation constante IV

  $H = DES_{MP}(IV)$
- On enregistre H dans /etc/passwd
- Pour vérifier le mot de passe MP' donné au login, on vérifie :

  $DES_{MP'}(IV) ?= H$

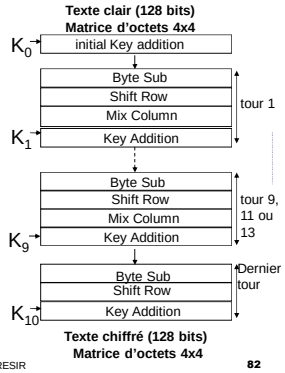
Mot de passe Unix (2/2)

- Réalité : on tire en plus à la première connexion une valeur de 12 bits (sel) pour paramétrer le DES
  => (ajout d'une permutation)
  => en fait 2^{12} = 4096 DES possibles
- Augmente de 12 bits la recherche exhaustive
- On enregistre le sel en plus dans /etc/passwd :

account:coded password data:uid:gid:GCOS-field:homedir:shell
gigawalt:tURfuu4.4hY0U:129:129:Walters:/home/gigawalt:/bin/csh
sel

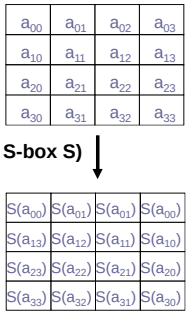
L'AES (1/3)

- 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 bits.
  - **Longueurs des clés** : 128, 192, ou 256 bits.
  - Le **nombre de tours varie** entre 10 et 14 selon la longueur des clés.

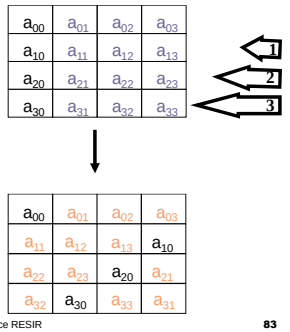


L'AES (2/3) : : La Fonction Étage 1/2

★ **Byte Substitution**

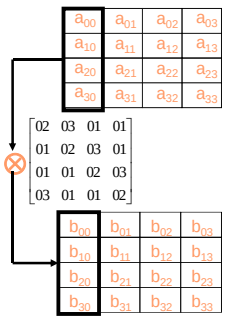


Shift Row

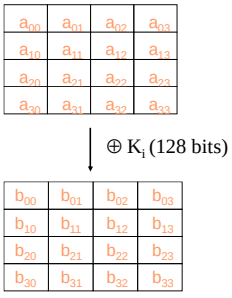


L'AES (3/3) : : La Fonction Étage 2/2

✹ **Mix Column**



❋ **Key Addition**



Quel chiffrement utilisé aujourd'hui ?

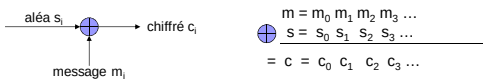
- Soit l'AES
- Soit le triple DES :
  - composition de deux DES
  - avec deux clés (112 bits de clé) :

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

=>Pour se prémunir contre la recherche exhaustive

Autre algorithme de cryptographie symétrique : le chiffrement à flot

- Utilisation du « one time pad » :



- 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,…)

Pourquoi des chiffrements à flot ?

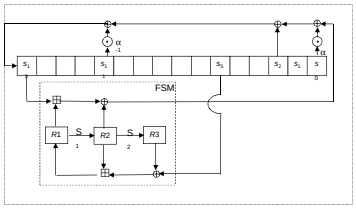
- Utilisation pour le software : chiffrement très rapide
- Utilisation en hardware avec des ressources restreintes
- Ne propage pas les erreurs (souvent utilisé en téléphonie mobile) (≠ chiffrement par blocs)
- Nouvelle utilisation : dans HFE pour la partie addition

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

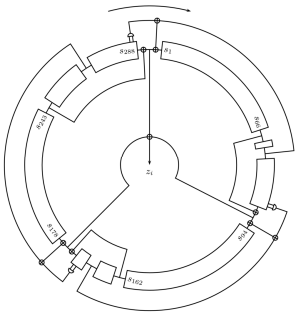
Exemple avec des LFSRs

- SNOW 3G utilisé dans 3GPP (vos téléphones !)
- LFSR sur GF(2^{32})



Exemple sans LFSR et hardware

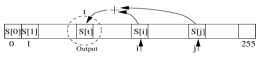
- Trivium
  - Init : 1152 steps



Un exemple particulier RC4 :

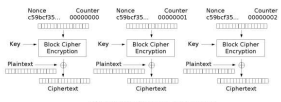
- Principe général :
  - Génération de l'aléa à partir d'un tableau d'état S_i de 256 octets
- Initialisation du tableau à partir de la clé K
  - Pour i de 0 à 255, $S_i = i$
  - Pour i de 0 à 255, K_i = clé K
  - j = 0, pour i de 0 à 255
    - $j = (j + S_i + K_i) \bmod 256$
    - Échanger S_i et S_j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

=> Biais !

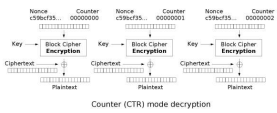


Stream with block: the counter mode

- The counter mode: from a counter and a key => produce stream with a block cipher



ciphering



deciphering

Comparaison de performances :

- En hardware (2003)
  - DES : 1,1 Gbytes/ seconde
  - AES : 1,95 Gbytes/s.
  - RC4 : 0,685 Gbytes/s. (vieil algorithme)
  - Trivium : 10 Gbytes/s

What is a cryptographic Hash function? (1/2)

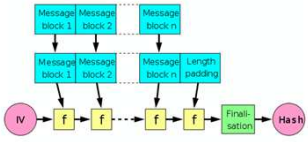
- A function H that maps a message of an arbitrary length {0,1}* into a fingerprint of fixed size $\{0,1\}^n$: h=H(M).
- Used: everywhere !
- Must be:
  - Preimage resistant: for all y in $\{0,1\}^n$, very difficult (2^n tries) to find x in $\{0,1\}^n$ / y=H(x)
    => One-wayness
  - 2^{nd} preimage resistant: for all x in {0,1}*, very difficult (2^n tries) to find x≠x' / H(x)=H(x')
  - Collision resistant: very difficult ($2^{n/2}$ tries) to find x≠x' / H(x)=H(x')
    => at least 160 bits of output !

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

What is a cryptographic Hash function? (2/2)

- Usual constructions:
  - repeat a compression function of fixed input/output size (typically 256 bits)
- Example: Merkle-Damgard

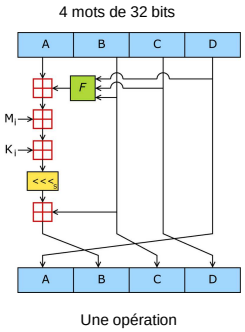


Exemples :

- 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 :

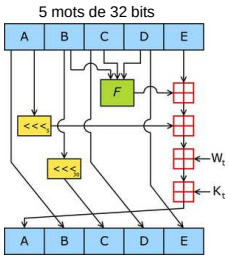
- Composé de la répétition de 64 opérations regroupées en 4 fois 16 opérations
- $F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$



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



Une opération

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

Which cryptographic Hash functions I could use?

- Hum Hum… when it's red, it's wrong…

MD4 [Rivest 90] RFC 1320 — [Dobbertin 96] À la main

MD5 [Rivest 91] RFC 1321 — [Wang et al. 04] millisecondes; [Karpman et al. 15] 2^{61} opérations

SHA-0 (92) FIPS pub-180 — [Joux et al. 04] 1 heure

SHA-1 (95) FIPS pub-180-1 — [Wang et al. 05] 2^{69} opérations

SHA-2 (01) FIPS pub-180-2

HAVAL [Zheng et al 92] — [Wang et al. 04]

RIPEMD [Dobbertin et al. 92]

RIPEMD-160 [Dobbertin et al. 96] — [Wang et al. 04]

Example of MD5 collisions

```
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,
0x83,0xB4,0x88,0x83,0x25,0x71,0x41,0x5A,
0x08,0x51,0x25,0xE8,0xF7,0xCD,0xC9,0x9F,
0xD9,0x1D,0xBD,0xF2,0x80,0x37,0x3C,0x5B,
0x97,0x9E,0xBD,0xB4,0x0E,0x2A,0x6E,0x17,
0xA6,0x23,0x57,0x24,0xDl,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,0xC9,0x55,0x5C,
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,
0x83,0xB4,0x88,0x83,0x25,0xF1,0x41,0x5A,
0x08,0x51,0x25,0xE8,0xF7,0xCD,0xC9,0x9F,
0xD9,0x1D,0xBD,0x72,0x80,0x37,0x3C,0x5B,
0x97,0x9E,0xBD,0xB4,0x0E,0x2A,0x6E,0x17,
0xA6,0x23,0x57,0x24,0xDl,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,0x41,0x55,0x5C,
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.
```


Example of SHA-1 free start collisions (8th October 2015)

Table 2-1. A freestart collision for SHA-1

	Message 1																
IV_1	50	6b	01	78	ff	6d	18	90	20	22	91	fd	3a	de	38	71	b2	c6	65	ea
M_1							9d	44	38	28	a5	ea	3d	f0	86	ea	a0	fa	77	83	a7	36
							33	24	48	4d	af	70	2a	aa	a3	da	b6	79	d8	a6	9e	2d
							54	38	20	ed	a7	ff	fb	52	d3	ff	49	3f	c3	ff	55	1e
							fb	ff	d9	7f	55	fe	ee	f2	08	5a	f3	12	08	86	88	a9
$Compr(IV_1,M_1)$	f0	20	48	6f	07	1b	f1	10	53	54	7a	86	f4	a7	15	3b	3c	95	0f	4b
	Message 2																			
IV_2	50	6b	01	78	ff	6d	18	91	a0	22	91	fd	3a	de	38	71	b2	c6	65	ea
M_2							3f	44	38	38	81	ea	3d	ec	a0	ea	a0	ee	51	83	a7	2c
							33	24	48	5d	ab	70	2a	b6	6f	da	b6	6d	d4	a6	9e	2f
							94	38	20	fd	13	ff	fb	4e	ef	ff	49	3b	7f	ff	55	04
							db	ff	d9	6f	71	fe	ee	ee	e4	5a	f3	06	04	86	88	ab
$Compr(IV_2,M_2)$	f0	20	48	6f	07	1b	f1	10	53	54	7a	86	f4	a7	15	3b	3c	95	0f	4b

So, enter in the SHA-3 competition…

- http://csrc.nist.gov/groups/ST/hash/sha-3/
- International competition launched by NIST

Oct. 2008: Submission deadline	64 algorithms received by the NIST
Dec. 2008	51 algo. Retained in the first round
Juil. 2009	14 candidates for the 2nd round
Dec. 2010	5 finalists
End of 2012	And the winner is…

Ongoing competition…

- 2nd round Finalists:

Hash Name	Principal Submitter	Best Attack on Main NIST Requirements	Best Attack on other Hash Requirements
Blue Midnight Wish	Svein Johan Knapskog		
CubeHash	Daniel J. Bernstein	preimage	
ECHO	Henri Gilbert		
Fugue	Charanjit S. Jutla		
Hamsi	Özgül Küçük		
Luffa	Dai Watanabe		
Shabal	Jean-François Misarsky		
SHAvite-3	Orr Dunkelman		
SIMD	Gaëtan Leurent		

- 3d Finalists:

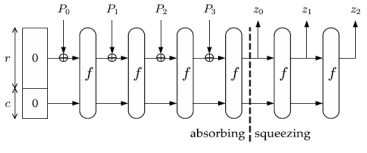
Hash Name	Principal Submitter	Best Attack on Main NIST Requirements	Best Attack on other Hash Requirements
BLAKE	Jean-Philippe Aumasson		
Grøstl	Lars R. Knudsen		
JH	Hongjun Wu	preimage	
Keccak	The Keccak Team		
Skein	Bruce Schneier		

And the winner is…

- Since the 2 of October 2012… Keccak !

keccak

Based on the sponge construction



- Variable-length input, indefinite-length output
- Secure against generic attacks with < $2^{c/2}$ calls to f
  - Indifferentiability proof assumes f is random permutation
  - Attacks exploiting specific properties of f are not covered
- Provable security against generic attacks

Keccak

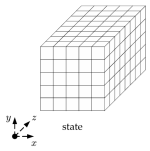
- KECCAK follows the **hermetic sponge strategy**
  - Instantiation of a sponge function
  - Permutation f shall be designed such that it has no exploitable properties
  - KECCAK uses a permutation KECCAK-f [r + c]
  - Primary choice: KECCAK-f [1600]: r + c = 1600
    - r = 1024 and c = 576 for $2^{c/2} = 2^{288}$ security, faster
    - r = 512 and c = 1088 for $2^{c/2} = 2^{544}$ security, slower

The KECCAK-*f* permutation

- KECCAK-*f* is an iterated permutation
  - Like a block cipher: Sequence of identical rounds, simple step mappings
  - BUT: No key schedule, Round constants instead of round keys, Inverse permutation need not be efficient

Inside KECCAK-*f* (1/6)

- The step mappings: θ, ρ, π, χ, ι
- The state: an array of $5 \times 5 \times 2^l$ bits



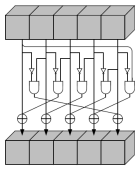
- 5 ×5 lanes, each containing 2^l bits (1, 2, 4, 8, 16, 32 or 64)
- (5 × 5)-bit slices, 2^l (1, 2, 4, 8, 16, 32 or 64) of them

Inside KECCAK-*f* (2/6)

- A round consists of 5 invertible step mappings:
  - θ for diffusion
  - ρ for inter-slice dispersion
  - π for disturbing horizontal/vertical alignment
  - χ for non-linearity
  - ι to break symmetry (round constant addition)
- Number of rounds: 12 + l KECCAK-*f* [25] has 12 rounds KECCAK-*f* [1600] has 18 rounds

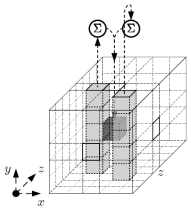
Inside KECCAK-*f* (3/6)

- χ for non-linearity

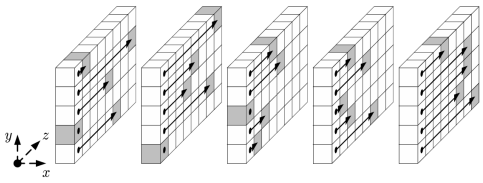


Inside KECCAK-*f* (4/6)

- θ for diffusion
  - Each input bit affects 11 output bits
  - 50 bitwise XORs and 5 rotations

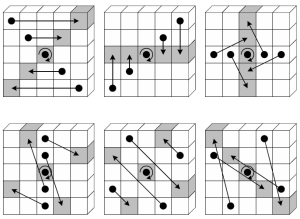


Inside KECCAK-*f* (5/6)



- ρ for inter-slice dispersion
  - Moves bits of a slice to 25 different slices
  - 24 rotations

Inside KECCAK-*f* (6/6)



- π for disturbing horizontal/vertical alignment
  - Cycle with period 24 around a fixed origin
  - Linear mapping of (*x*, *y*) coordinates in GF(5)

Why Keccak ?

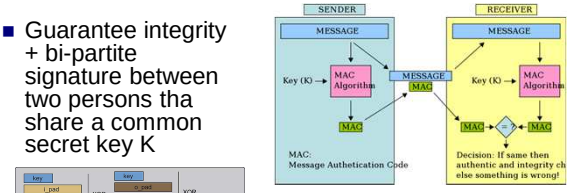
- Really efficient in both software and hardware
- No threating attacks: [BC 12] 2^{920} !
- => good candidate ! And the winner !

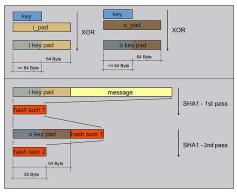
Utilité des fonctions de hachage

- 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

MACs: Message authentication Codes

- Guarantee integrity + bi-partite signature between two persons tha share a common secret key K





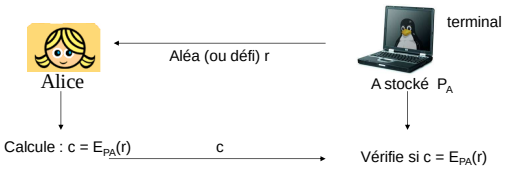
- Examples:
  - HMAC: based on hash functions used with a key K
  - CBC-MAC: based on block ciphers with a shared key K

Protocoles dédiés :

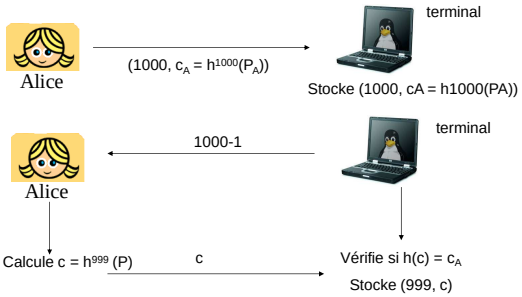
- Cryptographie = algorithmes + protocoles
- La solidité d'une communication dépend à la fois de :
  - La solidité des algorithmes cryptographiques
  - Des protocoles utilisées

Protocole aléa/retour :

- Première connexion : comme précedemment
- Puis



SKEY : RFC 2289

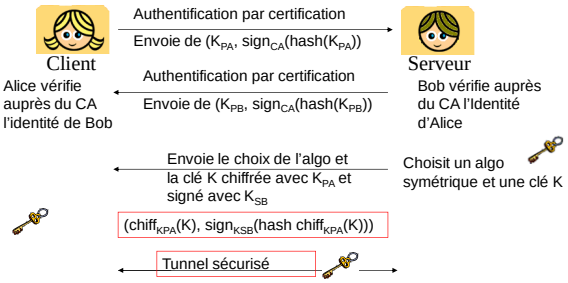


Usages Informatiques

En pratique

- La cryptographie est une « boite à outils »
- Il suffit de l'utiliser correctement
- Quelques applications :
  - OpenSSL
  - SSH et ses dérivés
  - PGP et le « Web of Trust »
  - PKI et exemples d'utilisation

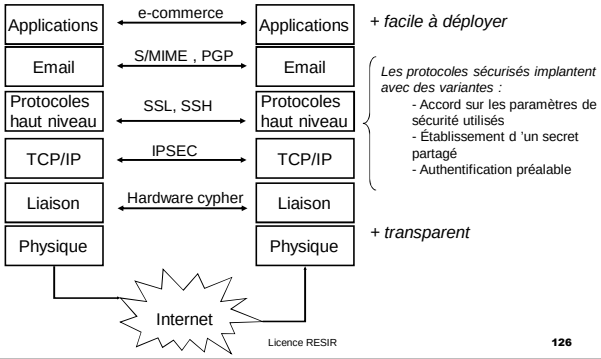
Schéma habituel d'utilisation



But : sécurité à tous les niveaux !

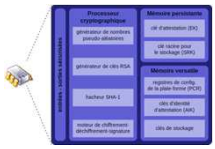
- PKI permet de sécuriser les communications à l'intérieur d'une entreprise, de chiffrer les données, les flux,…
- Mais comment sécuriser les communications ?

Protocoles et sécurité



Sécurité hardware

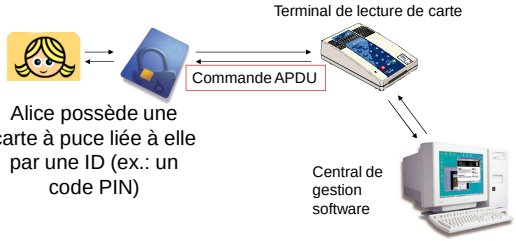
- TPM



- Cartes à puce

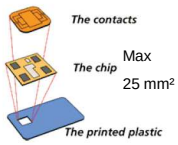
Les cartes à puce

- Fonctionnement



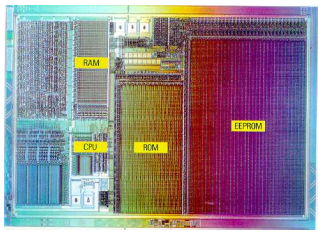
Les cartes à puce

- Architecture :
  - CPU : 8, 16 & 32 bits
  - Mémoire :
    - RAM : => 5 Ko
    - EEPROM/Flash : => 64 Ko
    - ROM : => 208 Ko



  - Cellule cryptographique (DES, AES, RSA) et générateur de nb al.
  - Détecteur de sécurité, registres, timer, IO interface,…

Une carte Gemplus



Différentes cartes

- Algorithmes de chiffrement implémentés différents selon les cartes
  - Carte de téléphone : compteur et algorithmes propriétaires inconnus
  - Carte SIM de téléphone portable : un MAC pour authentification
  - …

- Règle générale cependant :
  - Possède tous :
    - Une fonction de hashage : SHA-1 (intégrité)
    - Un algorithme de chiffrement : 3DES, AES,… (confidentialité)
    - RSA (chiffrement asymétrique)
    - DSA ou ECDSA (signature électronique)

Utilisation de la cryptographie dans une carte de paiement :

- Principe de l'échange électronique :
  - Le code PIN entré par l'utilisateur permet de
    - L'authentifier
    - Libérer les droits de lecture contenus dans la carte
  - Le lecteur (de cartes) vérifie que la carte n'est pas blacklister
  - Le lecteur authentifie la carte (3-DES)
  - La carte authentifie le lecteur
  - Le lecteur enlève le montant de façon sûre (MAC) au compte
  - Et il augmente de façon sûre (MAC) le compte du commerçant

IPSec

IP-Sec : introduction

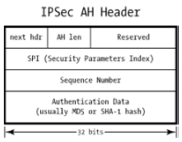
- Internet Security protocol, intégré à IPv6
- Objectifs : sécuriser les trames IP :
  - Confidentialité des données et protection partielle contre l'analyse du trafic
  - Authentification des données et contrôle d'accès continu
  - Protection contre le rejeu
- Principe :
  - ajout de champs d'authentification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

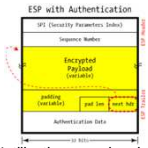
IP-Sec : les algos utilisés

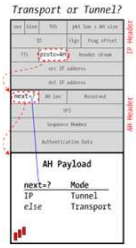
- IP-Sec s'appuie sur différents protocoles et algorithmes en fonction du niveau de sécurité souhaité :
  - **Authentification** par signature électronique à clé publique (RSA).
  - **Contrôle de l'intégrité** par fonction de hachage (MD5).
  - **Confidentialité** par l'intermédiaire d'algorithmes symétriques,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

IP-Sec

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



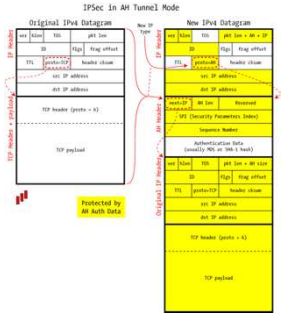




- Deux modes possibles d'utilisation avec les deux :
  - Transport
  - Tunnel

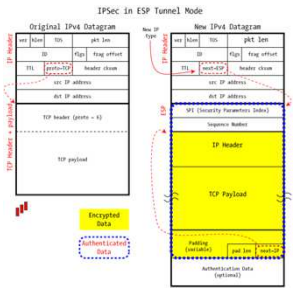
IP-Sec : mode tunnel avec AH

- 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)

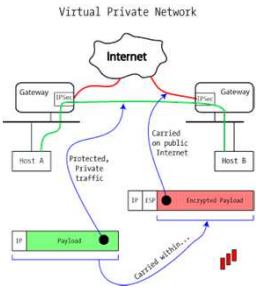
IP-Sec : les problèmes

- 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

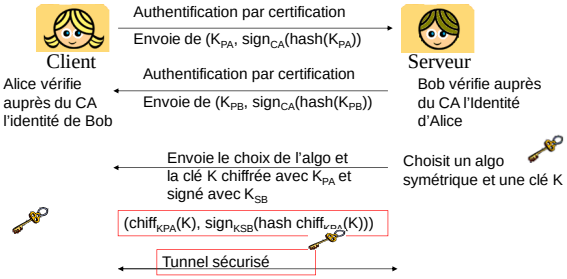
- **IPSec mode tunnel:** En mode tunnel, la session IPSec est établie entre deux passerelles IPSec, ou un host et une passerelle
  - Avantage: l'ensemble des communications traversant les passerelles VPN peuvent être sécurisées; pas de modification des hosts
  - Inconvénient: nécessite des passerelles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

SSL

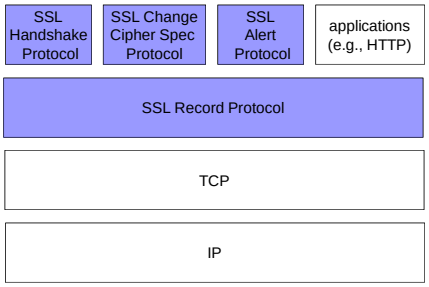
Schéma habituel d'utilisation



History of the protocol

- SSL 1.0
  - Internal Netscape design, early 1994?
  - Lost in the mists of time
- SSL 2.0
  - Published by Netscape, November 1994
  - Badly broken
- SSL 3.0
  - Designed by Netscape and Paul Kocher, November 1996
- TLS 1.0
  - Internet standard based on SSL 3.0, January 1999
  - Not interoperable with SSL 3.0

SSL architecture



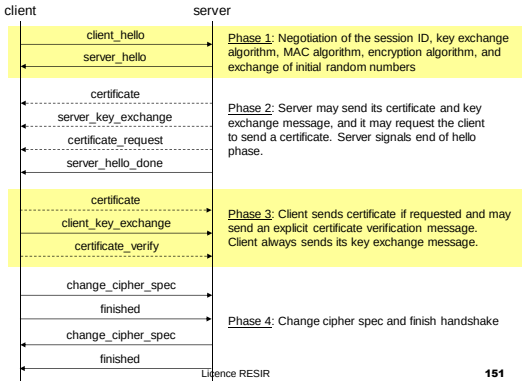
SSL components

- SSL Handshake Protocol
  - negotiation of security algorithms and parameters
  - key exchange
  - server authentication and optionally client authentication
- SSL Record Protocol
  - fragmentation
  - compression
  - message authentication and integrity protection
  - encryption
- SSL Alert Protocol
  - error messages (fatal alerts and warnings)
- SSL Change Cipher Spec Protocol
  - a single message that indicates the end of the SSL handshake

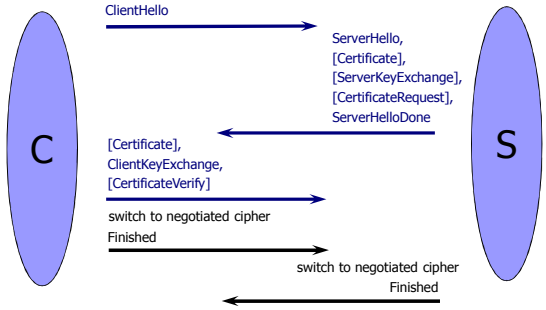
TLS Handshake Protocol

- Two parties: client and server
- Negotiate version of the protocol and the set of cryptographic algorithms to be used
  - Interoperability between different implementations of the protocol
- Authenticate client and server (optional)
  - Use digital certificates to learn each other's public keys and verify each other's identity
- Use public keys to establish a shared secret

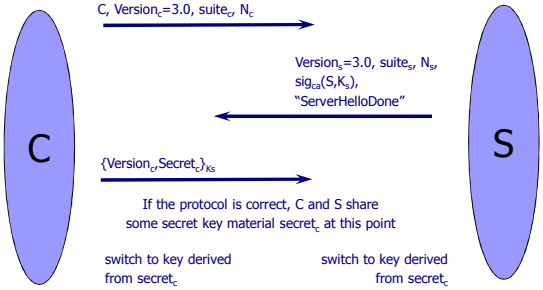
SSL Handshake Protocol – overview



Handshake Protocol Structure



"Core" SSL



Supported key exchange methods (1/2)

- RSA based (SSL_RSA_with...)
  - the secret key (pre-master secret) is encrypted with the server's public RSA key
  - the server's public key is made available to the client during the exchange
- fixed Diffie-Hellman (SSL_DH_RSA_with… or SSL_DH_DSS_with…)
  - the server has fix DH parameters contained in a certificate signed by a CA
  - the client may have fix DH parameters certified by a CA or it may send an unauthenticated one-time DH public value in the client_key_exchange message

Supported key exchange methods (2/2)

- ephemeral Diffie-Hellman (SSL_DHE_RSA_with… or SSL_DHE_DSS_with…)
  - both the server and the client generate one-time DH parameters
  - the server signs its DH parameters with its private RSA or DSS key
  - the client may authenticate itself (if requested by the server) by signing the hash of the handshake messages with its private RSA or DSS key
- anonymous Diffie-Hellman
  - both the server and the client generate one-time DH parameters
  - they send their parameters to the peer without authentication
- Fortezza
  - Fortezza proprietary key exchange scheme

Server certificate

- certificate
  - required for every key exchange method except for anonymous DH
  - contains one or a chain of X.509 certificates (up to a known root CA)
  - may contain
    - public RSA key suitable for encryption, or
    - public RSA or DSS key suitable for signing only, or
    - fix DH parameters

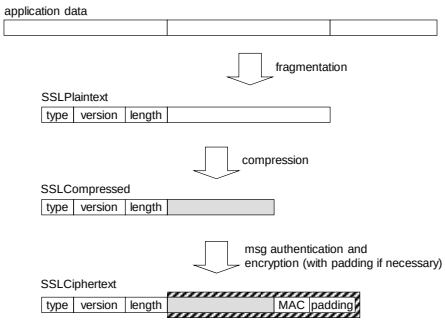
Certificate request

- certificate_request
  - sent if the client needs to authenticate itself
  - specifies which type of certificate is requested (rsa_sign, dss_sign, rsa_fixed_dh, dss_fixed_dh, …)

Client authentication

- certificate
  - sent only if requested by the server
  - may contain
    - public RSA or DSS key suitable for signing only, or
    - fix DH parameters

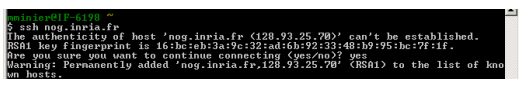
SSL Record Protocol –overview



SSH

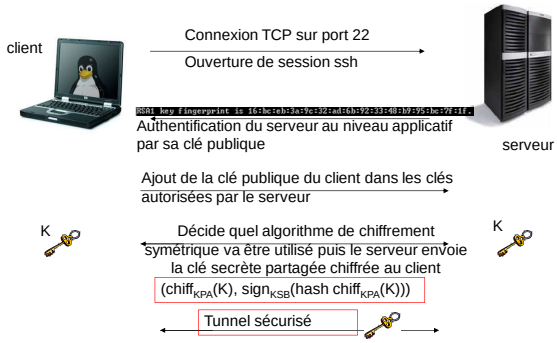
SSH : Secure Shell (1/4)

- Authentification
  - Coté client : par mot de passe ou par clé publique RSA ou DSA (ssh v2)
  - Coté serveur : Authentification par clef publique (RSA/DSA)
    - Transmise au client à la première session



    - Sauvegardée par le client

SSH et ses dérivées (2/4)



Les dérivées (3/4)

- Une fois le tunnel sécurisé établi :
  - Commandes normales en shell :copy,…
- Permet également la communication entre des serveurs
- Peut remplacer telnet, rlogin,…
- Extensions : sftp,…

SSH et ses dérivés (4/4)

- Problèmes d'attaques :
  - DNS spoofing
  - Attaque de type man in the middle qui se fait passer pour celui qu'il n'est pas
- Vient toujours du même problème :
  - La clé publique n'est pas garantie par un tiers de confiance !
  - Ou utilisation de mécanismes « hors bande », rencontre physique par exemple

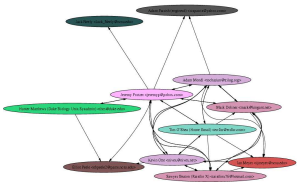
PGP

Premier Exemple : PGP

- PGP = Pretty good privacy sous licence GPL (jusqu'à la version 2.x)
- Cryptosystème hybride permettant l'échange de données chiffrées/authentifiées
  - Cryptographie symétrique pour chiffrer
  - Asymétrique pour signature et transport de clé de session

PGP et le « Web of trust »

- Authentification repose sur l'anneau de confiance : web of trust
  - L'authenticité d'une clé publique est prouvée de proche en proche
  - Adaptée au communautés pas à grande échelle !
  - Fondée sur la notion de COI « Community of interest »
- Exemple : le « Debian Keyring Web of Trust »



Principe : je signe avec ma clé secrète la clé publique des personnes dont je suis sûre

Distribution de paire de clés

- Généré par vous-même
- Ou serveur de clés et serveur référencent les clés
  - http://wwwkeys.pgp.net
  - Recherche sur une chaine de caractères, un nom,…
- Je donne ma clé publique à mes amis qui la signe si il me font confiance
  - Par exemple lors d'une « GnuPG Keysigning party »

PGP

- Pas d'autorité de certification
- Alice → Bob et Bob → Eve ⇒ Alice → Eve
- Fusion de plusieurs certificats
  - Réputation cachée
- Pours : pas de gestion centralisée
- Contres :
  - Initialisation / stockage
  - Transitivité de la confiance (pb si un noeud malveillant au milieu)

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

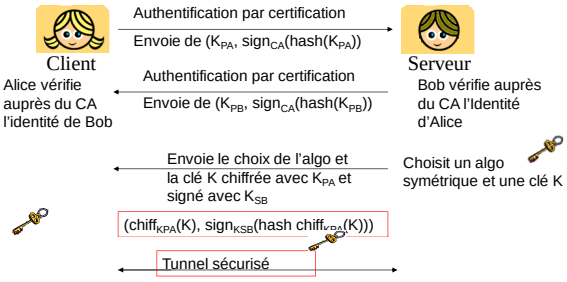
- Comme une carte d'identité national
  - Preuve de l'identité quand création de la carte
  - Unique, liée à une identité et non falsifiable
  - Déclaration en préfecture quand vol ou perte afin d'en obtenir une autre et pour ne pas qu'il y est une mauvaise utilisation de la disparue

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.anssi.gouv.fr/

Conclusion

Conclusion



Projets

Projets

- 4h ensemble
- 12h de travail personnel
- Soutenance d'une demi-heure