Contrôle d'accès & Authentification

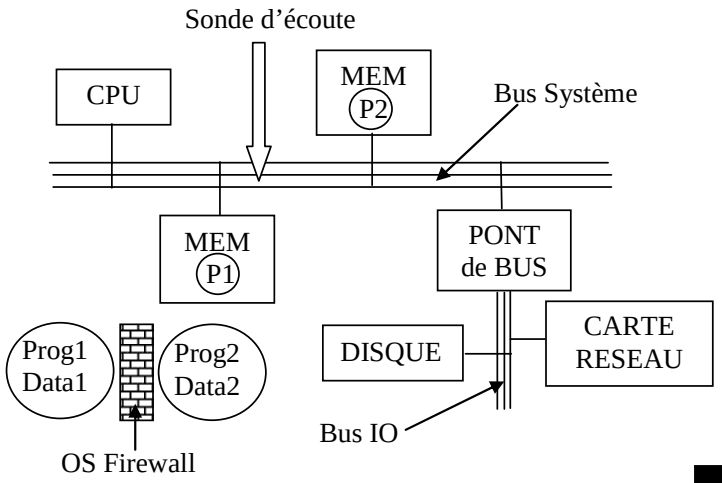
Pascal Urien
Télécom ParisTech
http://www.enst.fr/~urien/

Un rapide historique

Applications distribuées

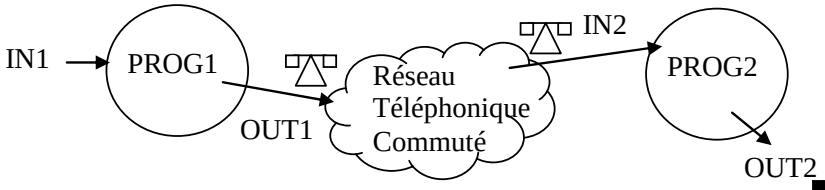
- **Une application distribuée est un ensemble d'entités logicielles, logiquement autonomes, qui produisent, consomment et échangent des informations**
  - OUTi = PROG(INi))
- **Dans un premier temps les composants logiciels des applications étaient logés dans un même système informatique, constituant de fait leur média de communication (parfois dénommé *gluware*).**
  - Le bus système permet le transfert des informations stockées en mémoire, les modules logiciels sont réalisés par des processus gérés par le système d'exploitation.
  - La sécurité est uniquement dépendante des caractéristiques du système d'exploitation, par exemple en terme de gestion des droits utilisateurs, ou d'isolement des processus.

GlueWare



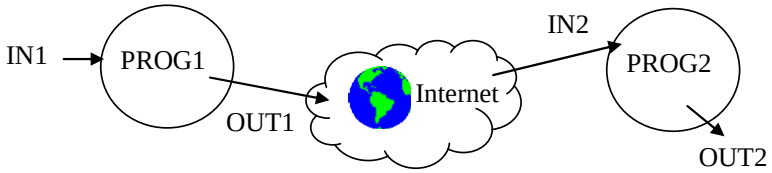
L'âge des MODEMs

- Dans une deuxième période l'application distribuée est répartie entre plusieurs systèmes informatiques reliés entre eux par des liens de communications supposés sures (c'est à dire qu'il est difficile d'enregistrer ou de modifier l'information transmise) tels que modems ou liaisons spécialisées (X25, RNIS …).
- Nous remarquerons à ce propos qu'il est possible de sécuriser une liaison de type point à point par un dispositif matériel de chiffrement.



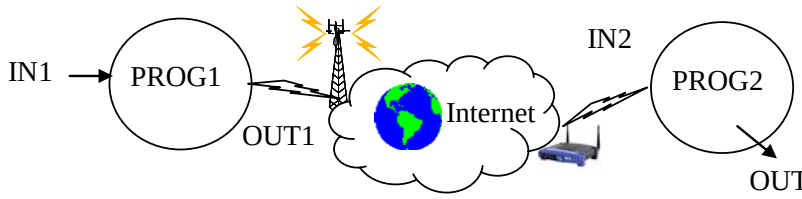
Internet Protocol

- Enfin l'émergence de la toile d'araignée mondiale a permis de concevoir des systèmes distribués à l'échelle planétaire, les composants logiciels sont répartis sur des systèmes informatiques hétéroclites, le réseau n'est pas sûr, le nombre d'utilisateurs est important.
- La sécurité devient un paramètre critique et tente de concilier des contraintes à priori antinomiques telles que, nécessité économique d'utiliser Internet, et impérative résistance à la piraterie informatique ou à l'espionnage.

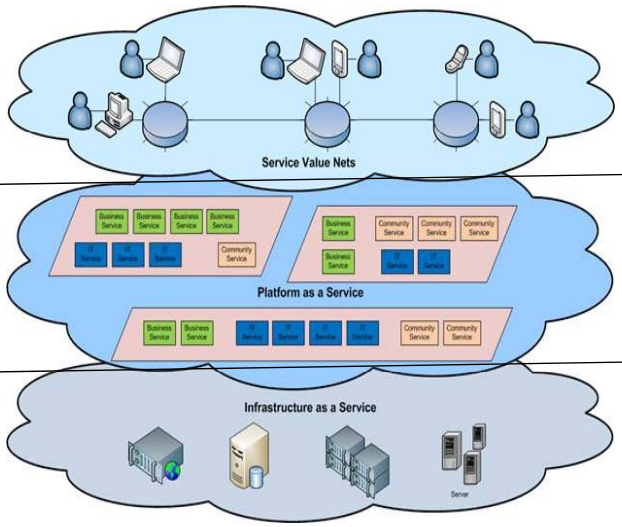


Ubiguitous Networks

- La dernière révolution des communications s'appuie sur les technologies de réseaux IP sans fil, tels que Wi-Fi ou WiMAX.
- Les liens filaires symboles d'une connectivité volontaire et contrôlée s'estompent, l'infrastructure du réseau devient diffuse et invisible. Un nouveau besoin de sécurité s'affirme, le contrôle des accès réseaux.



Le Cloud Computing



SaaS

PaaS

IaaS

Sécurité & Réseaux

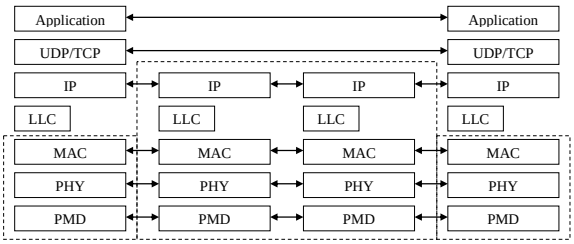
Principes de sécurité

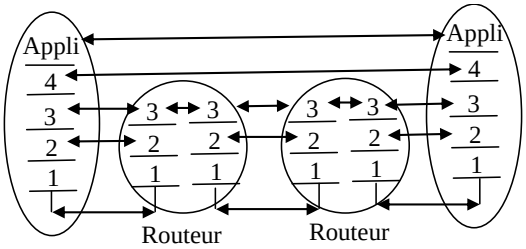
- L'identification (identity).
  - L'utilisateur d'un système ou de ressources diverses possède une identité (une sorte de clé primaire d'une base de données) qui détermine ses lettres de crédits (credential) et ses autorisations d'usage. Cette dernière peut être déclinée de multiples manières, compte utilisateur (login) d'un système d'exploitation ou techniques biométriques empreinte digitale, empreinte vocale, schéma rétinien…
- L'authentification (authentication).
  - Cette opération consiste à faire la preuve de son identité. Par exemple on peut utiliser un mot de passe, ou une méthode de défi basée sur une fonction cryptographique et un secret partagé. L'authentification est simple ou mutuelle selon les contraintes de l'environnement.
- La confidentialité (privacy).
  - C'est la garantie que les données échangées ne sont compréhensibles que pour les deux entités qui partagent un même secret souvent appelé association de sécurité (SA). Cette propriété implique la mise en oeuvre d'algorithmes de chiffrements soit en mode flux (octet par octet, comme par exemple dans RC4) soit en mode bloc (par exemple par série de 8 octets dans le cas du DES).
- L'intégrité des données (MAC, Message AuthentiCation).
  - Le chiffrement évite les écoutes indiscrètes, mais il ne protège pas contre la modification des informations par un intervenant mal intentionné. Des fonctions à sens unique (encore dénommées empreintes) telles que MD5 (16 octets) ou SHA1 (20 octets) réalisent ce service. Le MAC peut être associé à une clé secrète (HMAC(Message,clé), Keyed-Hashing for Message AuthentiCation).
- La non-répudiation.
  - Elle consiste à prouver l'origine des données. Généralement cette opération utilise une signature asymétrique en chiffrant l'empreinte du message avec la clé RSA privée de son auteur (RSA(Empreinte(Message))).
- On cite parfois un sixième attribut relatif à la sûreté de fonctionnement (disponibilité, résilience) du système.

De la confiance (TRUST)

- **La confiance est une relation sans propriétés particulières.**
  - *Réflexivité*, ai-je confiance en moi-même (pas dans tous domaines).
  - *Symétrie*, je fais confiance au pilote de l'avion ou au chirurgien, la réciproque n'est pas forcément vraie.
  - *Transitivité*, j'ai confiance dans le président, le président a confiance en la présidente, je n'ai pas obligatoirement confiance dans la présidente.
- **Les infrastructures PKI supposent une transitivité de la relation de confiance. Le client du réseau et un serveur d'authentification partagent une même autorité de certification (CA), qui crée une classe de confiance basée sur une relation R (R signifiant= «fait confiance à»).**
  - (Client R CA) ET (Serveur R CA) => (Client R Serveu

La sécurité appliquée aux réseaux

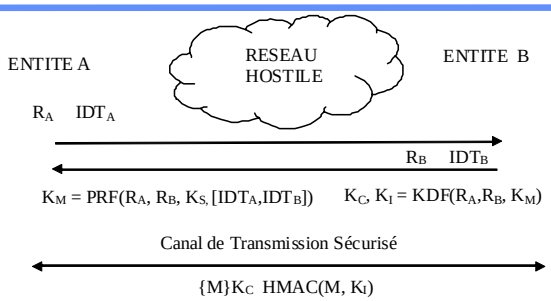




Comment sécuriser une pile réseau ?

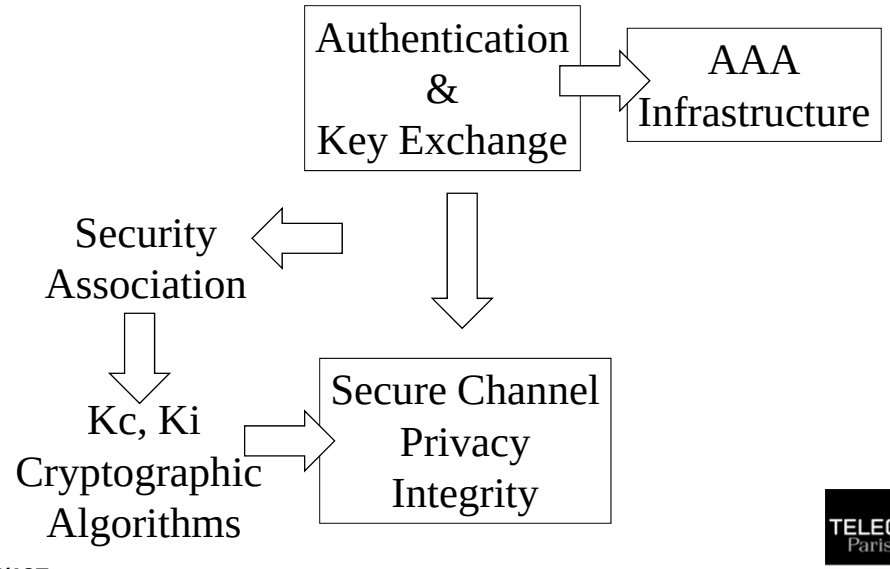
- **PHY- Le chiffrement au niveau physique sur des liaisons point à point.**
  - Par exemple cryptographie quantique (PMD), saut de fréquences pseudo aléatoire, ou chiffrement 3xDES du flux octets (une méthode couramment déployée par les banques). Dans ces différentes procédures les clés sont distribuées manuellement.
- **MAC- Confidentialité, intégrité de données, signature de trames MAC.**
  - C'est la technique choisie par les réseaux sans fil 802.11. La distribution des clés est réalisée dans un plan particulier (décrit par la norme IEEE 802.1x). Dans ce cas on introduit la notion de contrôle d'accès au réseau LAN, c'est à dire à la porte de communication avec la toile d'araignée mondiale. C'est une notion juridique importante, le but est d'interdire le transport des informations à des individus non authentifiés (et donc potentiellement criminels…)
- **TCP/IP- Confidentialité, intégrité de données, signature des paquets IP et/ou TCP.**
  - C'est typiquement la technologie IPSEC en mode tunnel. Un paquet IP chiffré et signé est encapsulé dans un paquet IP non protégé. En effet le routage à travers l'Internet implique l'analyse de l'en tête IP, par les passerelles traversées. IPSEC crée un tunnel sécurisé entre le réseau d'accès et le domaine du fournisseur de service. On peut déployer une gestion manuelle des clés ou des protocoles de distribution automatisés tels que ISAKMP. La philosophie de ce protocole s'appuie sur la libre utilisation du réseau d'accès ce qui n'est pas sans soulever des problèmes juridiques. Par exemple des criminels protègent leurs échanges de données, il est impossible aux réseaux traversés de détecter leur complicité dans le transport d'informations illégales.
- **ADDON- Insertion d'une couche de sécurité additive assurant la protection d'application telles que navigateurs WEB ou messageries électroniques.**
  - Par exemple le protocole SSL basé sur la cryptographie asymétrique réalise cette fonction. Généralement ce dernier conduit une simple authentification entre serveur et client. Il utilise un secret partagé (Master Secret) à partir duquel on dérive des clés de chiffrements utilisées par l'algorithme négocié entre les deux parties. Par exemple dans le cas d'une session entre un navigateur et un serveur bancaire, le client authentifie son service bancaire. Une fois le tunnel sécurisé établit le client s'authentifie à l'aide d'un login et d'un mot de passe. Il obtient alors une identité temporaire associée à un simple cookie.
- **APPLICATION- Gestion de la sécurité par l'application elle même.**
  - Ainsi le protocole S-MIME réalise la confidentialité, l'intégrité et la signature des contenus critiques d'un message électronique.

Création d'un canal sécurisé



- La procédure d'authentification d'une paire d'entités informatiques, parfois dénommée phase d'autorisation, consiste typiquement à échanger les identités (IDTA et IDTB) d'un couple d'interlocuteurs (appelés client/serveur ou initiateur/répondeur), deux nombres aléatoires (RA, RB) formant un identifiant unique de la session, puis d'effectuer un calcul.
- Ce dernier produit, à l'aide d'une valeur secrète (KS) un secret maître (KM), à partir duquel on déduit des clés de chiffrement (KC) et d'intégrité (KI) permettant de créer un canal sécurisé.
- Dans un contexte de cryptographie symétrique la clé KS est distribuée manuellement ; dans un contexte de cryptographie asymétrique la clé KS sera par exemple générée par A, mais chiffrée par la clé publique (e,n) de B (Ks^e mod n).
- La protection de l'identité est une préoccupation croissante avec l'émergence des technologies sans fil. Il existe divers mécanismes permettant d'obtenir cette propriété avec des degrés de confiance divers, par exemple grâce à la mise en œuvre de pseudonymes (tel que le TIMSI du GSM), du protocole de Diffie-Hellman, ou du chiffrement de certificats par la clé publique du serveur.

Mécanismes de base

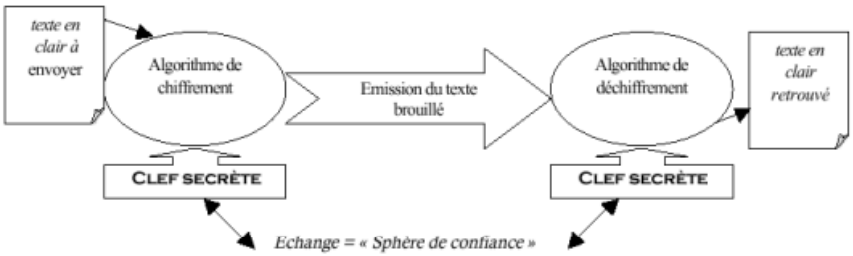


Quelles architectures ?

- **Clés symétriques distribuées manuellement**
  - Out Of Band
  - Pas de serveur d'authentification centralisé
- **Clés symétriques distribuées automatiquement**
  - Serveur d'authentification centralisé
- **Vecteurs d'authentification**
  - GSM, UMTS
  - Serveur d'authentification central ou réparti
- **Architecture basée sur des clés asymétriques**
  - Distribution de certificats et de clés RSA privées
  - Architecture répartie ou centralisée
  - Problème de la révocation

Bases Cryptographiques

Chiffrement symétrique 1/3



Une seule clé de chiffrement et déchiffrement

HAL

Chiffrement symétrique 2/3

- Les clés de chiffrement (*cipher keys*) et de déchiffrement (*uncipher)* sont identiques.
- Les méthodes de bases du chiffrement sont
  - La substitution : n! bijections de X[1,n] vers X[1,n]
    - Exemple le code de CESAR

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

  - La permutation (on dit encore transposition). Un ensemble de données en clair (b éléments) est découpé en b/p blocs de p éléments.
    - Exemple ensemble de 20 éléments divisé en 4 blocs de 5 éléments.

      x1 x2 x3 x4 x5
      x6 x7 x8 x9 x10
      x11 x12 x13 x14 x15
      X16 x17 x18 x19 x20

    - Les lignes et colonnes sont permutées selon un ordre connu (la clé de permutation)
- On distingue le chiffrement par flot (*stream cipher*) et le chiffrement par bloc (*bloc cipher*).
  - RC4 est un chiffrement par flot utilisé fréquemment par le protocole SSL. Il utilise des clés d'au plus 2048 bits et chiffre un octet (8 bits)
  - DES (Digital Encryption Algorithm) utilise des clés de 56 bits et chiffre des blocs de 64 bits
  - AES (Avanced Encryption Algorithm) utilise des clés de 128 bits et chiffre des blocs de 128 bits.

Chiffrement symétrique : notion d'entropie 3/3

- Claude Shannon a défini la notion d'entropie de l'information
  - $H = - \Sigma\, p(x) \log_2 p(x)$, soit $\log_2(n)$ pour n symboles équiprobables
- L'entropie conditionnelle de X sachant Y s'écrit
  - $H(X|Y) = - \Sigma\, P(X=x,Y=y) \log_2 P(X=x,Y=y)$
  - $H(X,Y) = H(X) + H(Y|X)$
- Un système cryptographique parfait au sens de Shannon est tel que
  - $H(M|C) = H(M)$, M le message en clair et C le message chiffré par une clé K.
- Par exemple une clé constituée par une suite d'octets aléatoires k1,k2,…..,ki réalise un système cryptographique parfait (Ci = ki exor Mi, Mi = ki exor Ci), c'est le code dit de *Vernam*.
- **En particulier si tous les messages en clair sont équiprobables la taille de la clé doit être au moins aussi grande que la taille des messages en clair.**

$$-\sum_{i=1}^{N} p_i \log_2(p_i)$$

Les chiffrements symétriques usuels

- **RC4, clés d'au plus 2048 bits**
  - Chiffrement par flux (octets)
- **3xDES, avec 2 clés (112 bits)**
  - Chiffrement de blocs de 64 bits
  - 2 clés K1, K2
  - y= E(K1,D(K2,E(K1,x))), soit chiffrement avec K1, déchiffrement avec K2, et chiffrement avec k1
- **AES avec une clé 128 bits**
  - Chiffrement de blocs de 128 bits
- **Le mode** *Cipher Block Chaining, CBC*
  - $IV_o = null$
  - $y_k = E(Key, IV_k \text{ exor } x_k)$
  - $IV_{k+1} = y_k$
- *CBC-MAC*
  - *Chiffrement en mode CBC*
  - *On ajoute au message un ensemble d'octets (padding) de telle sorte que la longueur obtenue soit un multiple de la taille du bloc. Le CBC-MAC est le dernier bloc calculé*
  - **ISO/IEC 9797**
    - *Method 1, padding avec une suite d'octets nuls*
    - *Method 2, padding avec un premier octet 0x80, complété par une suite d'octets nuls*
    - *Method 3, ajout d'un entête comportant la longueur, une attaque existe*

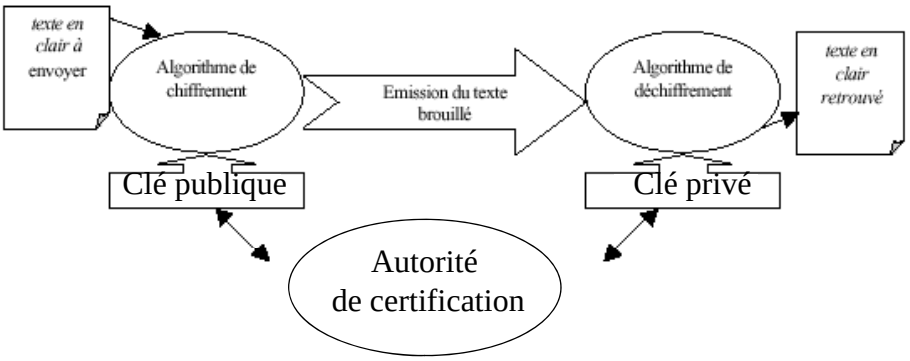
Fonctions de HASH (empreinte)

- **Une fonction d'empreinte (H) produit, à partir d'un message M une valeur pseudo aléatoire de p bits (soit 2^P empreintes). Les attaques sont classées en trois catégories**
  - Collision: trouver (M0,M) tel que H(M) = H(M0)
    - En raison du paradoxe des deux anniversaires, la probabilité d'une collision est de $1/2^{p/2}$
  - 1st pre-image, étant donné X, trouver M tel que H(M) = X
  - 2nd pre-image, étant donné M, trouver M0 tel que H(M0) = H(M)
  - Dans le cas d'un algorithme parfait, la complexité d'une collision est de $1/ 2^{P/2}$ et pour les *pre-image* $1/ 2^P$.

HMAC - Keyed-Hashing for Message Authentication – RFC 2104

- HMAC(K,m) = H((K ⊕ opad) ‖ H((K ⊕ ipad) ‖ m))
- L, la longueur d'un bloc de calcul de la fonction de hash H
- Si taille(K) > L, K= H(K)
- B = L- taille(K)
- ipad = l'octet 0x36 répété B fois
- opad = l'octet 0x5C répété B fois

Chiffrement Asymétrique



Au sujet des Groupes Abeliens finis

- **Un groupe Abelien (G,*) est un ensemble d'éléments tel que la loi * pour cet ensemble soit:**
  - définie pour tout couple (a,b), a*b Є G
  - commutative, a*b = b*a
  - associative, (a*b)*c = a *(b*c)
  - possède un élément neutre a*e = e*a = a
  - et que possède un élément inverse unique, $a*a^{-1} = a^{-1}*a = e$
- **Un groupe fini possède un nombre d'éléments fini.**
- **A l'aide du théorème de Bezout on démontre facilement que Z/pZ avec p premier est un groupe pour la loi x**
  - Z/pZ représente l'ensemble des restes de la division d'un nombre z par l'entier p.
  - $z = r \pmod p \Leftrightarrow z = qp + r$

Le nombre d'Euler

- **Le nombre d'Euler φ (n) est le nombre d'entiers premiers avec n**
  - φ (1)=1
  - Pour p premier, φ (p) = p-1
  - Pour p et q premiers, φ (p.q) = φ (p). φ (q)
- **Le nombre des entiers x tels que pgcd(a,x)=1 est égal à φ (a)**
  - (Z/nZ*,x) est le groupe des entiers inversible pour la loi x en modulo n
  - Le cardinal de ce groupe est égal à φ (n)

RSA 1/3

- **RSA**

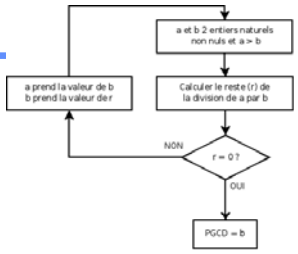
Soit n un produit de deux nombres premiers n = pq
Soit e un entier inversible modulo $\varphi(n)$,
c.a.d e premier avec $\varphi(n)$ =(q-1)(p-1)
$\exists$ d tel que e.d= 1 mod $\varphi(n)$

(e,n) clé publique, chiffrement C = M^e modulo n
(d,n) clé privée, déchiffrement M = C^d modulo n
M^{ed} = M modulo n

La solution de *a x = 1 mod n* peut être trouvée à l'aide de l'algorithme d'Euclide étendu.

Si, a et b ont un diviseur commun d (a >b) alors a-kb est divisible par d.
On choisit le plus grand k tel que a-kb = r (r>=0), soit a= kb + r.
Si r=0 alors a est divisible par b
Sinon on recommence l'algorithme avec b et r
Le PGCD est le dernier reste non nul.

L'algorithme d'Euclide RSA 2/3



- **Soit deux nombres a et b avec a>b**
  - a= bq + r, pgcd(a,b) = pgcd(b,r)
- **L'algorithme d'Euclide réalise le calcul de pgcd(a,b) en temps polynomial**
- **Deux nombres sont premiers entre entre eux si et seulement si pgcd(a,b)=1.**
- **Le théorème de Bezout se démontre grâce à l'algorithme d'Euclide**
  - Si a et b sont premiers entre eux il existe un couple unique d'entiers (u,v) tel que
    - au +bv = 1

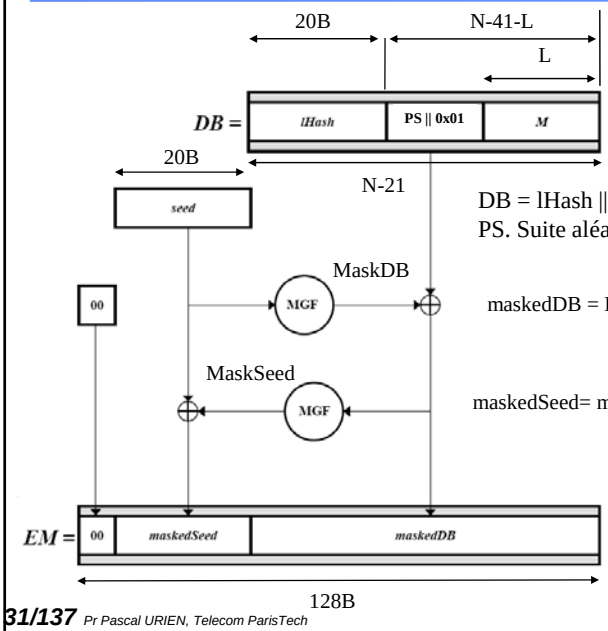
RSA 3/3

- **Exemple a=522, b=453**
  522=a, 453=b
  522-453 = 69
  453-6.69 = 39
  69-39=30
  39-30=9
  30-3.9=3
  3-3=0, 3 est le PGCD(522,453)
- *Exemple de calcul de clés RSA*
  p=47, q=59, n=pq = 2773, (p-1)(q-1)=2668
  17 est premier avec (p-1)(q-1)
  on cherche d 17 =1 mod 2668
  PGCD(17,2668)=1, 1 = a u + b v, v est la solution
  2668=a
  17=b
  2668-156.17 = 2668 – 2652 = 16 = a -156 b
  17-16 =1, b – (a – 156 b) = 1, 157 b – a = 1, d'ou d = 157

Le padding RSA PKCS#1-v1_5 (RFC 3347)

- **Le padding est nécessaire pour la sécurité de RSA**
  - Par exemple le chiffrement d'une même valeur x par trois clés RSA publiques (3, n) permet de retrouver la valeur x
    - $y1= x^3 \bmod n1$, $y2= x^3 \bmod n2$, $y3= x^3 \bmod n3$
    - Si n1, n2, n3 sont premiers entre eux, le théorème du résidu chinois permet de construire $y = x^3 \bmod (n1.n2.n3)$, puis de calculer la racine cubique de y égale à x (puisque x<n1, x<n2,x<n3
      - $n= n1.n2.n3$, $qi = (n/ni)^{-1}$ modulo ni, $y = \Sigma\ qi\ yi\ n/ni$ modulo n
- **Pour un chiffrement, $EM^e \bmod n$, N octets**
  - EM = 0x00 || 0x02 || PS || 0x00 || M
  - PS est un nombre aléatoire de taille N - longueur(M)-3 octets
- **Pour une signature, $EM^d \bmod n$, N octets**
  - EM = 0x00 || 0x01 || PS || 0x00 || M
  - PS est une série d'octets de taille N - longueur(M)-3 octets dont la valeur est 0xFF

RSAES-OAEP Encryption (RFC 3347)



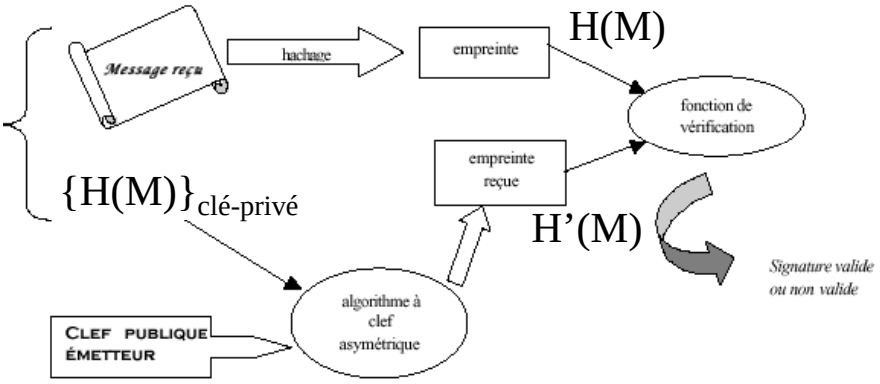
RSA 8.N bits

DB = lHash || PS || 0x01 || M.
PS. Suite aléatoire

maskedDB = DB + MGF(seed)

maskedSeed= maskedDB + MGF(maskedDB)

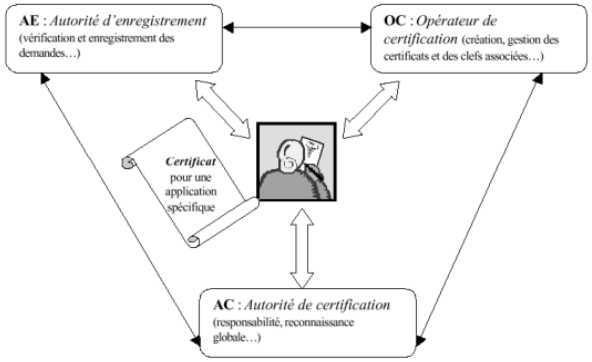
Signature



Signature = $\{H(M)\}_{\text{clé-privé}}$

Certificat

- C'est l'ensemble constitué par une suite de symbole (document M) et une signature.
- Le format de certificat le plus courant est X509 v2 ou v3. La syntaxe utilisée est l'ASN.1 (*Abstract Syntax Notation One*).



X509 v3

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signature            BIT STRING  }

TBSCertificate  ::=  SEQUENCE  {
        version         [0]  EXPLICIT Version DEFAULT v1,
        serialNumber         CertificateSerialNumber,
        signature            AlgorithmIdentifier,
        issuer               Name,
        validity             Validity,
        subject              Name,
        subjectPublicKeyInfo SubjectPublicKeyInfo,
        issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                             -- If present, version shall be v2 or v3
        subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                             -- If present, version shall be v2 or v3
        extensions      [3]  EXPLICIT Extensions OPTIONAL
                             -- If present, version shall be v3
        }
```

Public Key Infrastructure

- **Les principales fonctions réalisées par une architecture PKI pour la gestion des certificats se résument ainsi :**
  - Enregistrement de demande et vérification des critères pour attribution d'un certificat.
    - L'identité du demandeur est vérifiée ainsi que le fait qu'il soit bien en possession de la clef privée associée
  - Création des certificats
  - Diffusion des certificats entraînant la publication des clefs publiques
  - Archivage des certificats pour assurer la sécurité et la pérennité
  - Renouvellement des certificats en fin de période de validité
  - Suspension de certificats.
    - Elle peut être utile si le propriétaire estime ne pas avoir besoin temporairement de son certificat ; cependant cette fonction n'est pas aisée à mettre en œuvre ; elle est essentiellement administrative et il n'existe pas de standard d'implémentation
  - Révocation de certificats.
    - Sur date de péremption, perte, vol ou compromission de clefs
  - Création et publication (au sens gestion) des listes de révocation des certificats
    - Il y aura révocation du certificat dans les cas suivants : date de fin de validité atteinte, clef privée divulguée, perdue (donc impossibilité de lire les objets rendus confidentiels) ou compromise.
  - Délégation de pouvoir à d'autres entités reconnues de confiance.
    - Toute communauté peut créer sa propre infrastructure PKI, dans ce cas une étude de faisabilité est nécessaire en s'appuyant sur de nombreux critères.

Standards PKCS

- **PKCS** *« Public-Key Cryptography Standards »* **est un ensemble de standards pour la mise en place des IGC, coordonné par RSA ; ces standards définissent les formats des éléments de cryptographie :**
  - PKCS#1 : RSA Cryptography Specifications Version 2 *(RFC 2437)*
  - PKCS#2 : inclus dans PKCS#1
  - PKCS#3 : Diffie-Hellman Key Agreement Standard Version 1.4
  - PKCS#4 : inclus dans PKCS#1
  - PKCS#5 : Password-Based Cryptography Standard Version 2
  - PKCS#6 : Extended-Certificate Syntax Standard Version 1.5
  - PKCS#7 : Cryptographic Message Syntax Standard Version 1.5 *(RFC2315)*
  - PKCS#8 : Private-Key Information Syntax Standard Version 1.2
  - PKCS#9 : Selected Attribute Types Version 2.0
  - PKCS#10 : Certification Request Syntax Version 1.7 or Certificate Signing Request (CSR) *(RFC 2314)*
  - PKCS#11 : Cryptographic Token Interface Standard Version 2.10
  - PKCS#12 : Personnal Information Exchange Syntax Standard Version 1.0
  - PKCS#13 : Elliptic Curve Cryptography Standard Version 1.0
  - PKCS#14 : Pseudorandom Number Generation Standard Version 1.0
  - PKCS#15 : Cryptographic Token Information Format Standard Version 1.1

Diffie-Hellman

- Un générateur g de (Z*/pZ, x), avec p premier
  - $\forall x \in [1, p-1] \exists i \in [1, p-1]$ $g^i = x \pmod p$,
  - en particulier $g^{p-1}=1$
- Il existe au moins un g, qui est premier avec p-1, soit au moins $\varphi(p-1)$ solutions.
- Exemple p=5, g=3, g est premier avec (5-1=4)
  - $g^1 = 1.3 = 3$
  - $g^2 = 3.3 = 4$
  - $g^3 = 4.3 = 2$
  - $g^4 = 2.3 = 1$

Plus loin avec les Groupes finis

- (G,x) un groupe fini, cardinal G = |G|
- L'ordre d'un élément g (g ∈ G) est le plus petit entier e tel que $g^e=1$
- H est un sous groupe de G si
  - H est inclus dans G
  - (H,x) est un groupe
- Soit g un élément de G, l'ensemble <g>= { g^k: k ∈ Z} est un sous groupe de G
  - C'est le groupe engendré par g
  - Son cardinal est égal à l'ordre g
- G =<g> pour un certain g on dit que G est un groupe cyclique
- Si G est fini et cyclique il existe exactement , φ (|G|) générateurs de G.
- Théorème de Lagrange
  - Si G est un groupe fini le cardinal (ou l'ordre) de chaque sous groupe divise le cardinal de G
  - L'entier |G|/|H| s'appelle l'indice de H dans G (*cofactor*)
- Théorème de Sylow
  - G un groupe fini. Pour tout nombre premier p et tout nombre entier r tel que p^r divise l'ordre de G, il existe un sous groupe de G d'ordre p^r.
- Si G est un groupe fini, et g ∈ G, $g^{|G|}=1$
- On en déduit le petit théorème de Fermat,
  - si pgcd(a,m)=1, alors $a^{\varphi(n)} = 1 \bmod m$
  - En particulier $a^{\varphi(n)-1} a = 1 \bmod m$, ce qui donne l'inverse de en modul
  - $a^{-1} = a^{\varphi(n)-1} \bmod m$

Comprendre le théorème de Sylow dans (Z/pZ,+)

- **Dans (Z/pZ, +) la solution (x) de ax=b (a, b étant connu) n'est pas une procédure complexe**
  - une solution existe si pgcd(a,p) divise b
- **L'ordre r d'un élément a (tel que ar=0) s'obtient par la relation**
  - ordre(a) = ppcm(a,p) / p = a x / pgcd(a,p)
  - En particulier lorsque a est premier avec p, ordre(a)=p
- **Exemple dans (Z/36Z, +), 36 = 2^2 x 3^2**
  - Les ordres possibles sont 2,4,3,9,6,12,18,36

Les corps finis

- **Un corps K est un ensemble d'éléments munis de deux lois notées + et x, tel que**
  - (K,+) est un groupe abélien, élément neutre e
  - (K*, x) est un groupe abélien, K* représente l 'ensemble K sans le neutre e, élément neutre i
  - Distributivité de la deuxième loi (x) sur la première (+)
    - a*(b+c) = ab + cd
- **L'intérêt des corps est d'obtenir des propriétés algébriques permettant :**
  - La solution de systèmes d'équations linéaires
  - La définition de polynômes
- **(Z/pZ, +, x) avec p premier est un corps fini.**

Les polynômes

- **Dans un corps K on peut définir des polynômes P, de degré p**
  - $P(x) = a_0 + a_1 x + a_2 x^2 + \ldots a_p x^p$
  - (P,+,x) est un anneau
- **Soit P un polynôme de degré p et Q un polynôme de degré q, p ≥ q, on définit la division de P par Q**
  - P = Q A + R, le couple (A,R) de polynômes est unique, et degré(R) < p.
- **P est divisible par Q si et seulement (deg P ≥ deg B) et si le reste de la division de P par Q est nul.**
- **L'algorithme d'Euclide s'applique aux polynômes, et permet de calculer le pgcd de deux polynômes.**
- **Deux polynômes P et Q sont étrangers si et seulement si pgcd(P,Q) est un polynôme de degré 0.**
- **Le théorème de Bezout s'applique aux polynômes étrangers**
  - Si P et Q sont étrangers, il existe un couple unique (U,V) de polynômes étrangers, tels que, P.U + Q.V = 1

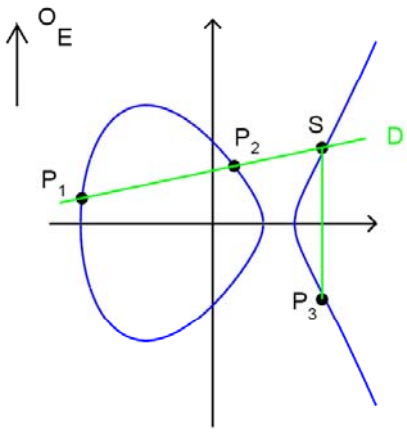
Les corps de Galois

- **Théorème: Soit *p* un nombre premier et *I(X)* un polynôme irréductible de degré *n≥1* dans *Z/pZ[X]*.**
  - Alors *(Z/pZ[X])/I , l'ensemble des restes des divisions par I(X)*) est un corps.
  - Il possède p^n éléments et il ne dépend pas de *I* (seulement du degré de *I*).
  - Ce corps est appelé corps de Galois à p^n éléments, noté $CG(p^n)$ (ou $GF(p^n)$ en anglais).
  - La caractéristique d'un corps (char(K)) est l'entier premier p
    - **Pour tout élément x de $CG(p^n)$, p.x= 0**
    - En particulier $(x+y)^{p\hat{}k} = x^{p\hat{}k} + y^{p\hat{}k}$ pour k entier positif
- **Théorème: Les seuls corps finis qui existent sont les corps de Galois.**

Courbes Elliptiques

- **Dans un corps K (fini) Fq,**
  - $q=p^m$ ou $q=p$, avec p premier, p étant différent de 2 ou 3 (la caractéristique du corps est différente de 2 ou 3)
  - $x,y,a,b \in K$, $y^2 = x^3 + ax + b$
  - $\Delta = 4a^3 + 27b^2 \neq 0$
- **ECC(Fq,a,b) = { P(x,y): $y^2 = x^3 + ax + b$} U {O}**
  - O = Point à l'infini
  - Pour tout P on impose , P+O = O+P = O
  - Pour P(x,y)#O, on définit -P=(x,-y), et on pose P + -P =O
- **Pour p=2**
  - ECC($F2^m$,a,b) = { P(x,y): $y^2 + xy = x^3 + ax^2 + b$} U {O}

ECC: Addition



Loi additive

- Soit deux points P1 et P2 de la courbe
  - Si P1=O ou P2=O, P1+P2=O
  - Sinon si P1= -P2 ou P2=-P1, P1+P2 = O
  - Sinon, P3(x3,y3)=P1(x1,y1)+P2(x2,y2)

$$x_3 = \begin{cases} \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 & \text{si } P_1 \neq P_2 \\ \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 & \text{si } P_1 = P_2 \end{cases}$$

$$y_3 = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}(x_1 - x_3) - y_1 & \text{si } P_1 \neq P_2 \\ \frac{3x_1^2 + a}{2y_1}(x_1 - x_3) - y_1 & \text{si } P_1 = P_2 \end{cases}$$

Groupe d'une courbe elliptique

- L'ensemble des points (x,y) de Fq, de la courbe elliptique forme un groupe E(Fq) dont le cardinal est noté #E(Fq)
- Théorème de Hasse

$$q + 1 - 2\sqrt{q} \leq \#E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q}$$

- #E(Fq) = q + 1−t, avec |t| ≤ 2√q. La quantité t est appelée trace de Frobenius.
- Un point de E(Fq) est représenté par $\log_2 q + 1$ bits
  - $\log_2 q$ bits pour x
  - 1 bit pour le choix de y

Avantage des Courbes Elliptiques

security (bits)	block cipher	$E/\mathbb{F}_p$	$E/\mathbb{F}_{2^m}$	RSA
112	3-DES	224	233	2048
128	AES small	256	283	3072
192	AES medium	384	409	8192
256	AES large	512	571	14720

Diffie Hellman pour ECC

- G un sous groupe de ECC(Fq,a,b)
- G est cyclique d'ordre n, G=<P>, le point P est un générateur de G
- En notation additive
  - aP=P+P+…+P
  - bP=P+P+…+P
  - Secret partagé Diffie Hellman
    - DH=a (bP) = b(aP) = abP

Signature ECDSA

- **G un sous groupe de ECC(Fq,a,b)**
- **#ECC= n = index . q = cofactor . order**
  - G est cyclique d'ordre premier q, G=<P>, le point P est un générateur de G
- **La clé privé est a Є [0, q-1], on calcule aP (clé publique)**
- **La clé éphémère est k Є [0, q-1], on calcule kP**
- **On note kP=(u,v), x = u mod q**
- **La signature d'un message m, est le couple (x,y)**
  - x = u mod q (ou entier r)
  - $y = k^{-1}(H(m)+ax) \bmod q$ (la signature s)
    - La fonction H est généralement sha1
- **L'opération de vérification s'écrit**
  - $(i,j)= (H(m)y^{-1} P + x\, y^{-1}(aP)) \bmod q$
  - x = i mod q ?
- **Clé privé (P,q,a,k), Clé publique(P,q,aP,kP)**
- **RFC 3278**
  - "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)"
  - ECDSA-Sig-Value ::= SEQUENCE { r INTEGER, s INTEGER }

Quelques normes utiles

- **Norme ANSI X9.62**
  - Représentation d'un point au format compressé ou non compressé
  - Exemple, $N= \log_{256} q$ (pour un corps Fq)
    - Type, 1 octet (04=uncompressed, 03=compressed)
    - Valeur de x (N octets), Valeur de y (N octets)
- **IEEE Std 1363**
  - " IEEE Standard Specifications for Public-Key Cryptography"
  - Terminologie pour les courbes elliptiques
  - Key Agreement (DH), DL/ECKAS-DH1
    - Sha1(x|y), x N octets, y N octets, $N= \log_{256} q$ (pour un corps Fq)
- **SEC 2**
  - "Recommended Elliptic Curve Domain Parameters, Certicom Research"
  - Recommandation de courbes elliptiques particulières
- **FIPS PUB 186-2**
  - **DIGITAL SIGNATURE STANDARD (DSS)**
  - "Recommandation de courbes elliptiques particulières"
- **RFC 3278**
  - "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)"

Exemple Courbe Certicom sect113r1

- Dans $F2^m$, avec m=113, représentation des coordonnées (x,y) 15 octets
- Courbe: $y^2 + xy = x^3 + ax^2 + b$
  - a= 003088250CA6E7C7FE649CE85820F7
  - b= 00E8BEE4D3E2260744188BE0E9C723
- Polynôme générateur: $x^{113}+x^9 + 1$
  - 02000000000000000000000000201
- Générateur (forme non compressé)
  - 04 009D73616F35F4AB1407D73562C10F 00A52830277958EE84D1315ED31886
- Ordre
  - 0100000000000000D9CCEC8A39E56F
- Cofacteur
  - 2
- Exemple de clé
  - Privé a= 000D2634C36BDE27916D7C590136CD
  - Publique aP= 04 01A352487A98884A2A35EEDBB4A93B 0052525EFAC8DA9B62A56D40BBCBEE
- Exemple de signature ECCDSA
  - 30 22
  - 02 0F 0020634AAF9F9B385A6CD10086377E
  - 02 0F 00E6855729E55AAB86D69CE2646415
- Exemple de calcul ECCDH
  - 20 octets: 82461C62A3BC762DF2F3270BDD6DC9CC58FD9E17

Pairage bilineaire

- Soit G1, G2, et Gr des groupes finis d'ordre r, avec r premier
- Un pairage bilinéaire e (*bilinear pairing, bilinear map*) est une fonction telle que
  - e: G1xG2 -> Gr
  - e est non dégénérée, e(P,Q) # 1
  - e est bilinéaire $e(aP, bQ) = e(P,Q)^{ab}$

Groupe de r-torsion

- **E une courbe elliptique définie sur un corps K (Fp) de caractéristique q ($p=q^m$)**
  - Les points de E peuvent être définis (il existe un isomorphisme) sur des corps plus grands Fq^k , ou encore la clôture algébrique de K .
- **Pour r entier, le sous groupe de points de r-torsion, défini pour une courbe elliptique E sur un corps K, est noté**
  - $E(K)[r] = \{ P \in E \mid rP = O\}$
- **Pour r premier avec q (soit pgcd(r,q)=1)**
  - E(K)[r] est isomorphe à Z/rZ x Z/rZ
  - E(K)[r] contient r^2 points
- **Si r divise p-1 (q^m - 1), le corps K (Fp) contient les racines $r^{ième}$ de l'unité**

Embedding Degree

- **Soit E une courbe elliptique définie sur un corps K (Fp), avec $p = q^m$**
- **Soit G un sous groupe cyclique de E(K)**
  - Il existe k tel que G soit isomorphe à un sous groupe de Fp^k
  - La plus petite valeur de k est appelé le *embedding degree*.
    - Soit n le cardinal de G (#G)
    - n divise $p^k - 1$, $n \mid p^k - 1$
- **De manière équivalente k est le plus petit entier, tel que Fp^k contienne le groupe μ_n des racines $n^{ième}$ de l'unité dans $\overline{Fp}$**

Couplage de Weil

- Couplage de Weil
  - Corps K =Fq
  - e: E(K) [r] x E(K) [r] -> Um
    - $e(aP,bQ) = e(P,Q)^{ab}$
  - Um est le groupe des racines $r^{ième}$ de l'unité dans Fq^k
    - $Um = \{ x \in Fq^k \mid x^r = 1 \}$
    - r divise le cardinal de E(K) (# E(K))
    - r divise $q^k - 1$
  - Le premier algorithme de couplage de Weil a été proposé en 1986 par *V. Miller*, "Short Programs for Functions on Curves"

Exemple de couplage de Weil

- ECC(Fq): $y^2 = x^3 + x$, corps Fq, q premier avec q = 3 modulo 4
- G1 est un sous groupe de ECC(Fq), G2 est un sous groupe de Fq^2
- Il y a q+1 points sur la courbe #ECC(Fq) = q+1
- r = ordre de G1 = facteur premier de q+1
- h = cofactor = #ECC(Fq) / r
  - r divise q+1
  - r divise $q^2 - 1$ puisque $q^2 - 1 = (q-1)(q+1)$
- q=8780710799663312522437781984754049815806883199414208211028653399266475630880222957078625179422662221423155858769582317459277713367317481 324925129998224791.
- h=120160122648911460793888213667405342048029544012513118229196151310472072893597045311028448021839065377867765
- r=730750818665451621361119245571504901405976559617
- P=764213932795790385166146156484628185710738246136731223594607305863197148904107335230752866953232919510098156557991388877251113225844051396939078151410688 4,841053126166803064145349163706237513167951671763744094302723035409824870295101995804868584972949481861515630634339691266774480234634049031935396

Couplage de Tate

- **E une courbe elliptique sur un corps K (Fp), de caractéristique q (p= q^m)**
- **E(K)[r] un groupe de r-torsion avec r premier avec q**
- **Soit k le plus petit entier tel que r divise p^k-1**
  - k est le « **embedding degree** » de la courbe elliptique E relativement à r
- **e: E(Fp) [r] x E(Fp^k) [r] -> Um**
  - Um est le groupe des racines $r^{ième}$ de l'unité dans Fp^k
    - Um = { x Є Fp^k | x^r = 1 }
- **e(aP,bQ)= e(P,Q)^{ab}**

Quelques Applications Emergentes

- **DH Tripartie**
  - Trois clés publiques aP, bP, cP
  - Trois clés privées a,b,c
  - e(aP,bP)^c = e(bP,cP)^a = e(bP,cP)^a
- **IBE (Identity Based Encryption) version de base**
  - P un point d'ordre r
  - Clé maitre (privée) s Є [0, r-1]
  - Clé publique sP
  - h: {identités} -> Z/rZ
  - h': Fq^k -> Z/nZ
  - r un nombre aléatoire dans [0, r-1]
  - Chiffrement à l'aide de la clé publique sP, et d'un nombre aléatoire r
    - Secret = e(h(ID)P, sP)^r,= e(P,P)$^{s\ r\ h(ID)}$ C = M exor Secret
  - Transmission (ID, h(ID)P, rP, C)
  - Déchiffrement S= e(h(ID)P,rP)^s = e(P,P)$^{s\ r\ h(ID)}$, M = C exor S

Kerberos

Idée Générale

- **Kerberos est un protocole développé par le MIT**
  - Les deux versions majeures sont v4 et v5
    - La version 4 s'appuie sur l'algorithme DES
    - La version 5 supporte 3xDES et AES
  - C'est un standard IETF, RFC 1510 (kerberos v5, 1993)
- **Paradigmes**
  - Utilisation de tickets établissant une preuve d'identité entre deux entités (un utilisateur et un service)
  - Génération de clés de session
- **Eléments**
  - Utilisateur
  - KDC, Key distribution Center
    - AS, Authentication Server
    - TGS, Ticket Granting Service
    - Base de données des clients et des clés
      - Key Version Number (kvno) est un index de clé ou de mot de passe
  - Serveur d'applications
- **Structures des messages**
  - Les messages sont codés selon la syntaxe ASN.1

Un peu de vocabulaire

- **Realm**
  - Un nom de domaine lié (example.com) à une autorité d'authentification
  - Un utilisateur appartient à un domaine si il partage un mot de passe ou une clé cryptographique avec ce dernier.
- **Principal**
  - Clé d'identification dans la base de donné
    - *component1/component2/.../componentN@REALM*
  - Exemples
    - user@example.com
    - Service/Hostname@REALM
      - *imap/mbox.example.com@EXAMPLE.COM*
- Ticket
  - Un ticket est délivré par le serveur d'authentification, il comporte
    - Une zone d'information chiffrée avec la clé de l'utilisateur
      - *Date Validité du ticket*
      - *Clé de session du service*
    - Une zone d'information chiffrée avec la clé du service
      - **Date Validité**
      - **Clé de session du service**

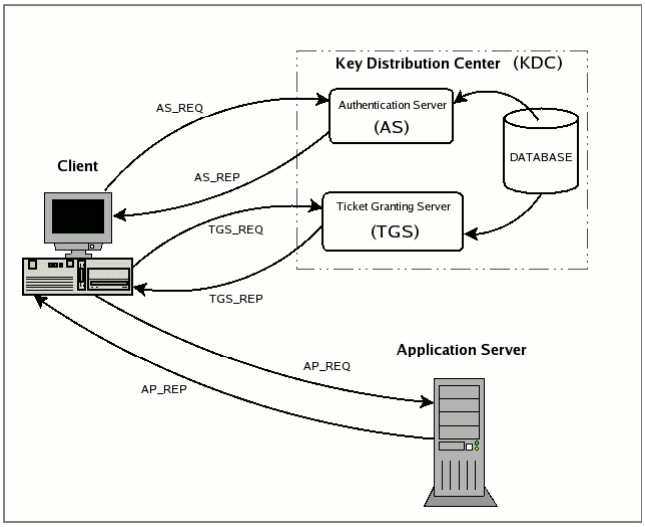
KDC et Clé de Session

- **Composants du KDC**
  - La base de données stocke toutes les informations associées à un principal
    - Mots de passe, clé, etc…
  - Le serveur d'authentification
    - Il réalise l'authentification d'un utilisateur basé sur son mot de passe
    - Il délivre un ticket d'authentification, *Ticket Granting Ticket*, ou TGT, dont le *principal* est krbtgt/REALM@REALM
  - Le Ticket Granting Server (TGS)
    - Il délivre des tickets de services à un utilisateur authentifié, c'est-à-dire muni d'un TGT.
- **La clé de session (Session Key)**
  - Pour un utilisateur c'est une clé cryptographique (DES, 3xDES, AES) déduite de son mot de passe
  - Pour un service, c'est une clé générée par le KDC

Au sujet des authenticator

- Un *authenticator* est un authentifiant
  - Il est réalisé grâce au chiffrement avec la clé de session d'un ensemble de paramètres comportant l'identité de l'utilisateur et la date
- Un *authenticator* associé à un ticket de service évite la duplication illicite de ticket
  - Le ticket de service est chiffré avec la clé de session
  - La présence d'un *authenticator* « frais » prouve la connaissance de la clé de session

Architecture Kerberos



AS_REQ, AS_REP

- **Authentication Server Request (AS_REQ)**
  - AS_REQ = ($Principal_{Client}$, $Principal_{Service}$, IP_list , Lifetime)
    - $Principal_{Client}$ = user@REALM
    - $Principal_{Service}$= krbtgt/REALM@REALM
- **Authentication Server Reply (AS_REP)**
  - TGT = ($Principal_{Client}$, krbtgt/REALM@REALM , IP_list , Timestamp , Lifetime , SK_{TGS})
  - AS_REP = { $Principal_{Service}$, Timestamp , Lifetime , SK_{TGS} }K_{User} { TGT }K_{TGS}

TGS-REQ, TGS_REP, AP_REQ, AP_REP

- **Ticket Granting Server Request (TGS_REQ)**
  - Authenticator = { $Principal_{Client}$, Timestamp }SK_{TGS}
  - TGS_REQ = ($Principal_{Service}$, Lifetime , Authenticator) { TGT }K_{TGS}
- **Ticket Granting Server Replay (TGS_REP)**
  - $T_{Service}$ = ($Principal_{Client}$, $Principal_{Service}$, IP_list , Timestamp , Lifetime , $SK_{Service}$)
  - TGS_REP = { $Principal_{Service}$, Timestamp , Lifetime , $SK_{Service}$ }SK_{TGS} { $T_{Service}$ }$K_{service}$
- **Application Request (AP_REQ)**
  - Authenticator = { $Principal_{Client}$, Timestamp }$SK_{Service}$
  - AP_REQ = Authenticator { $T_{Service}$ } $K_{service}$
- **Application Response (AP_REP)**
  - AP_REP = { $T_{Service}$ +1} $K_{service}$

Définitions ASN.1

- **Ticket ::= [APPLICATION 1] SEQUENCE {**
  - tkt-vno[0] INTEGER,
  - realm[1] Realm,
  - sname[2] PrincipalName,
  - enc-part[3] EncryptedData }
- **EncTicketPart ::= [APPLICATION 3] SEQUENCE {**
  - flags[0] TicketFlags,
  - key[1] EncryptionKey,
  - crealm[2] Realm,
  - cname[3] PrincipalName,
  - transited[4] TransitedEncoding,
  - authtime[5] KerberosTime,
  - starttime[6] KerberosTime OPTIONAL,
  - endtime[7] KerberosTime,
  - renew-till[8] KerberosTime OPTIONAL,
  - caddr[9] HostAddresses
  - OPTIONAL, authorization-data[10] AuthorizationData OPTIONAL }
- **AuthorizationData ::= SEQUENCE OF SEQUENCE {**
  - ad-type[0] INTEGER,
  - ad-data[1] OCTET STRING }

Définitions ASN.1

- **Authenticator ::= [APPLICATION 2] SEQUENCE {**
  - authenticator-vno[0] INTEGER,
  - crealm[1] Realm,
  - cname[2] PrincipalName,
  - cksum[3] Checksum OPTIONAL,
  - cusec[4] INTEGER,
  - ctime[5] KerberosTime,
  - subkey[6] EncryptionKey OPTIONAL,
  - seq-number[7] INTEGER OPTIONAL,
  - authorization-data[8] AuthorizationData OPTIONAL }

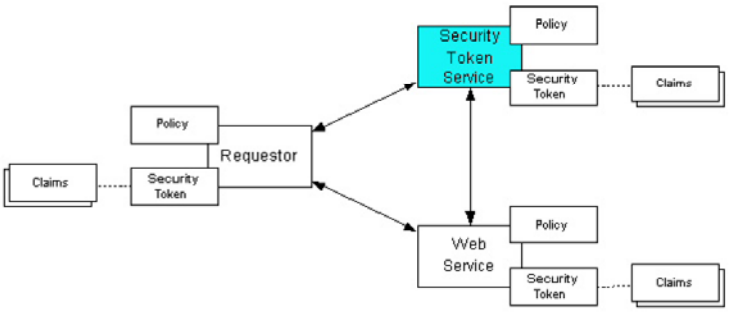
Définitions ASN.1

- **AP-REQ ::= [APPLICATION 14] SEQUENCE {**
  - pvno[0] INTEGER,
  - msg-type[1] INTEGER,
  - ap-options[2] APOptions,
  - ticket[3] Ticket,
  - authenticator[4] EncryptedData }
- **APOptions ::= BIT STRING {**
  - reserved(0),
  - use-session-key(1),
  - mutual-required(2) }
- **AP-REP ::= [APPLICATION 15] SEQUENCE {**
  - pvno[0] INTEGER,
  - msg-type[1] INTEGER,
  - enc-part[2] EncryptedData }
- **EncAPRepPart ::= [APPLICATION 27] SEQUENCE {**
  - ctime[0] KerberosTime,
  - cusec[1] INTEGER,
  - subkey[2] EncryptionKey OPTIONAL,
  - seq-number[3] INTEGER OPTIONAL }

Jetons d'authentification

WS_TRUST: Web Services Trust Language

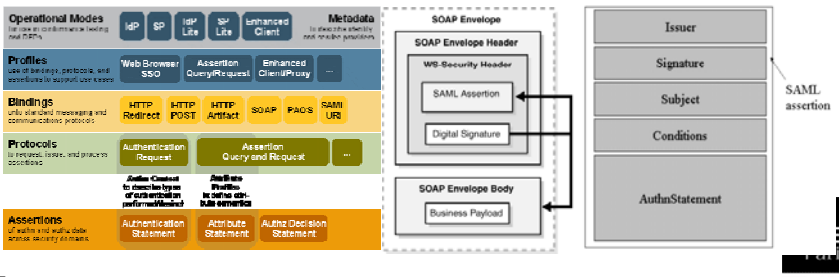
- *Claim* (droit): une demande relative à un client, un service ou tout autre ressource (nom, identité, clés, privilèges…)
- *Security Token*: un ensemble de claims
- *Signed Security Token*, un jeton sécurisé muni d'une signature
- *Proof-of-Possession Token*: la preuve (POP) de possession d'un jeton sécurisé, typiquement à l'aide d'une clé cryptographique
- *Security Token Service* (STS): un service WEB qui délivre des jetons sécurisés



71/13

Security Assertion Markup Language - SAML

- Une syntaxe basée sur XML, dédié à l'authentification de l'utilisateur d'un service et des attributs associés.
- SAML est utilisé par les projets Liberty Alliance, Shibboleth (Internet 2) et WS_Security (OASIS Web Services Security)
- SAML introduit les notions d'assertion, de « *protocols binding* », et de profile
  - Une assertion est un ensemble de données généré par une autorité SAML
    - Authentification, une preuve d'identité délivrée par un Identity Provider
    - Attributs, un ensemble d'attribut liés à un sujet
    - Décision d'autorisation, un droit pour accéder à une ressource
  - Un ensemble de protocoles de type requêtes/réponse réalisant, entre autres, authentification et délivrance d'assertions
  - Bindings, transport des messages SAML par des protocoles tels que SOAP
  - Profiles, définition de l'usage de SAML pour des apllications particulières (Web SSO, etc…)

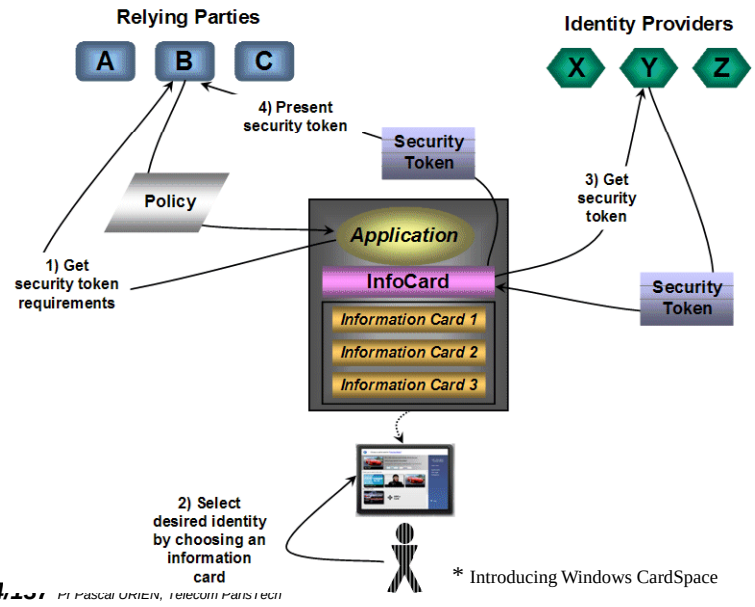


72/137 Pr Pascal URIEN, Telecom ParisTech

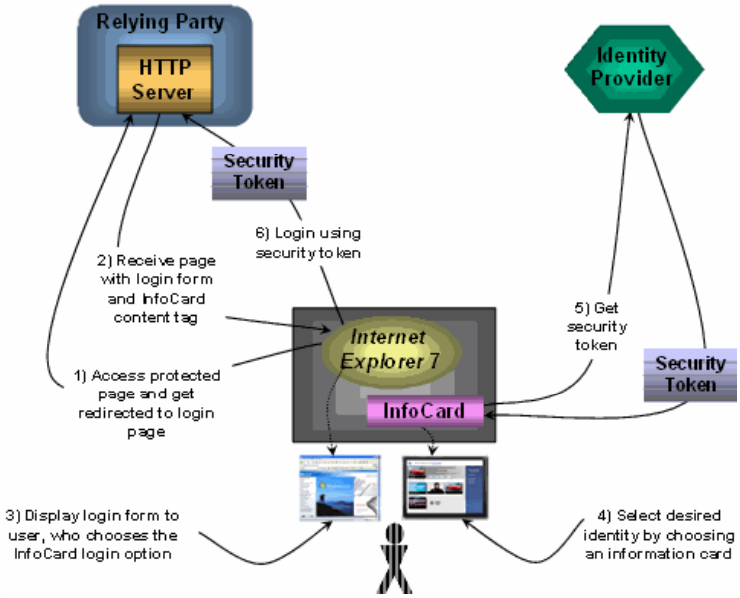
Information card (Microsoft)

- An *information card* represents a digital identity of a user issued by an identity provider.
- Multiple digital identities for a user from the same identity provider are represented by different information cards.
- Users may obtain an information card from an identity provider, and may have a collection of information cards from various identity providers.
- An information card is simply an artifact that contains metadata and represents the token-issuance relationship between an identity provider and a user.
- It serves the very important purpose of transforming something abstract like digital identity into something concrete and tangible for users to work with in their digital interactions.
- Furthermore, being concrete entities, they are portable and can be carried around by a user to be used from any computer or device through which Web services are accessed.

*A Technical Reference for the Information Card Profile V1.0



* Introducing Windows CardSpace



* Introducing Windows CardSpace

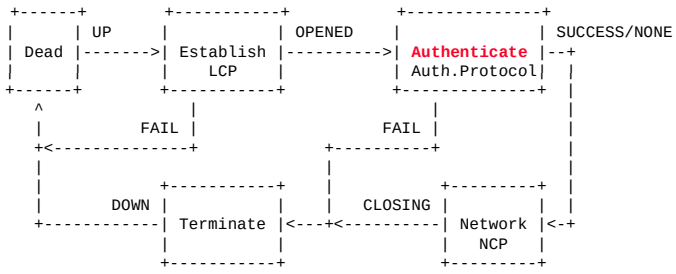
Example de binding XML

```
<ic:InformationCard xml:lang="xs:language" ...>
<ic:InformationCardReference> … </ic:InformationCardReference>
<ic:CardName> xs:string </ic:CardName>?
<ic:CardImage MimeType="xs:string"> xs:base64Binary </ic:CardImage>
<ic:Issuer> xs:anyURI </ic:Issuer>
<ic:TimeIssued> xs:dateTime </ic:TimeIssued>
<ic:TimeExpires> xs:dateTime </ic:TimeExpires>
<ic:TokenServiceList> ... </ic:TokenServiceList>
<ic:SupportedTokenTypeList> ... </ic:SupportedTokenTypeList>
<ic:SupportedClaimTypeList> ... </ic:SupportedClaimTypeList>
<ic:RequireAppliesTo ...> ... </ic:RequireAppliesTo> ?
<ic:PrivacyNotice ...> ... </ic:PrivacyNotice> ?
 ...
</ic:InformationCard>
```

PPP – MSCHAPv2

Au sujet de PPP

- PPP (RFC 1661, 1994) est LE protocole (niveau 2) pour les MODEMs
- Dans PPP le contrôle d'accès est réalisé avant l'allocation d'une adresse IP.



Flag 0x7E	Address 0xFF	Control 03	Protocol 2 bytes	information 1500 octets max	CRC 2 bytes	Flag 0x7E

- **Protocol Field Value**
  - 0x0021 : IP
  - 0xC021 : Link Control Protocol (LCP)
  - 0x8021 : Network Control Protocol (NCP)
  - 0xC023 : Password Authentication Protocol (PAP)
  - 0xC025 : Link Quality Report (LQR)
  - 0xC223 : Challenge Handshake Authentication Protocol (CHAP)

MOT de passe

MOST POPULAR PASSWORDS

Nearly one million RockYou users chose these passwords to protect their accounts.

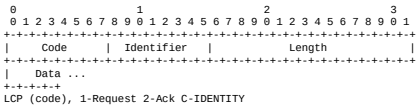
1. **123456**	17. **michael**
2. **12345**	18. **ashley**
3. **123456789**	19. **654321**
4. **password**	20. **qwerty**
5. **iloveyou**	21. **iloveu**
6. **princess**	22. **michelle**
7. **rockyou**	23. **111111**
8. **1234567**	24. **0**
9. **12345678**	25. **tigger**
10. **abc123**	26. **password1**
11. **nicole**	27. **sunshine**
12. **daniel**	28. **chocolate**
13. **babygirl**	29. **anthony**
14. **monkey**	30. **angel**
15. **jessica**	31. **FRIENDS**
16. **lovely**	32. **soccer**

Source: Imperva

Un exemple d'authentification PPP, CHAP

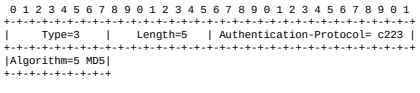
- RFC1994 - PPP Challenge Handshake Authentication Protocol_ CHAP

LCP Coding

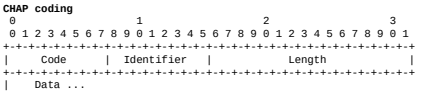


LCP (code), 1-Request 2-Ack C-IDENTITY

LCP Option=3, Authentication Request

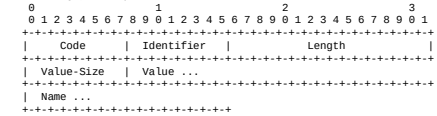


CHAP coding

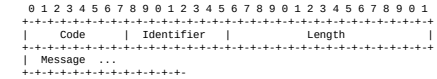


Code

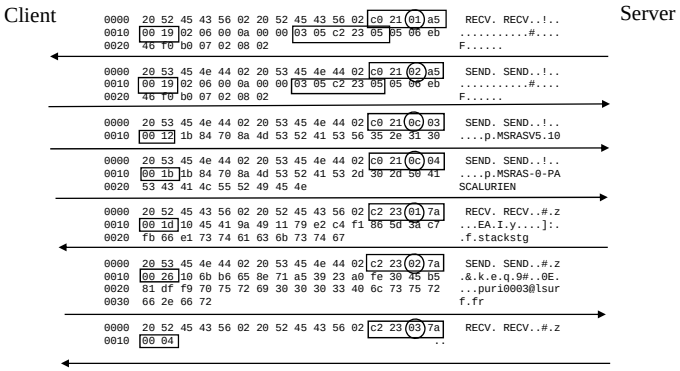
1- Challenge, 2-Response



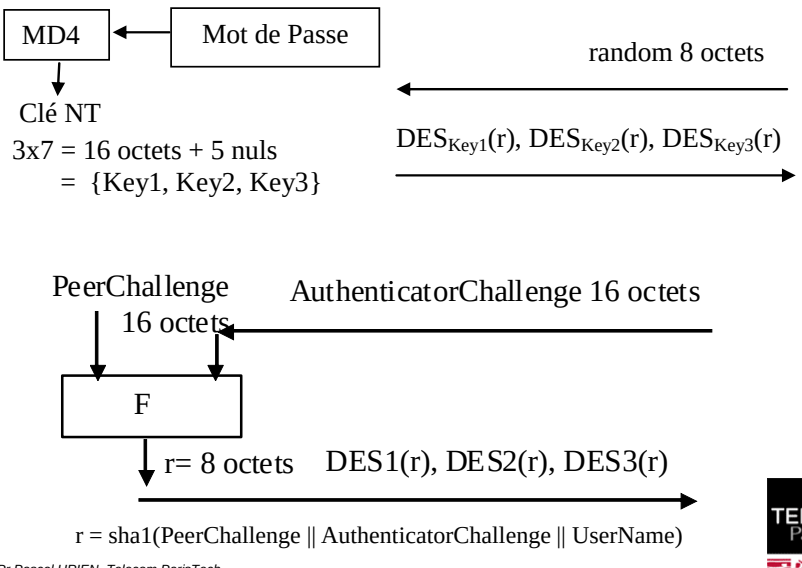
3-Success, 4-Failure



Une trace CHAP



Clé NT et MSCHAPv2 (RFC 2759, 2000)



MSCHAP-V2: Changement de mot de passe

- Ce message est forgé par l'utilisateur en réponse à un message d'erreur lors d'une tentative d'authentification (ERROR_PASSWORD_EXPIRED)
- Change Password Packet
  - 1 octet : Code=7
  - 1 octet : Identifier
  - 2 octets : Length
  - 516 octets : Encrypted-Password
    - Le nouveau mot de passe est chiffré avec l'algorithme RC4, dont la clé est déduite de l'ancienne valeur de la clé NT
  - 16 octets : Encrypted-Hash
    - L'ancienne clé NT (16 octets) chiffrée avec la nouvelle clé NT (DES1, DES2)
  - 16 octets : Peer-Challenge
  - 8 octets : Reserved
  - 24 octets : NT-Response
  - 2-octet : Flags

SSL/TLS & HTTP

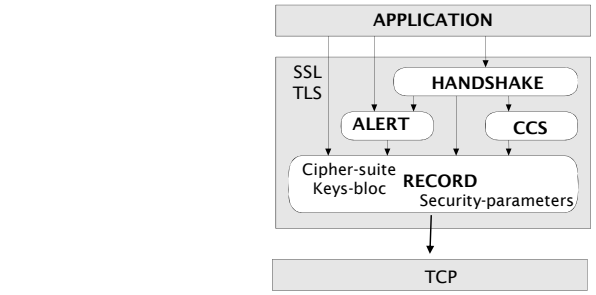
SSL/TLS Historique

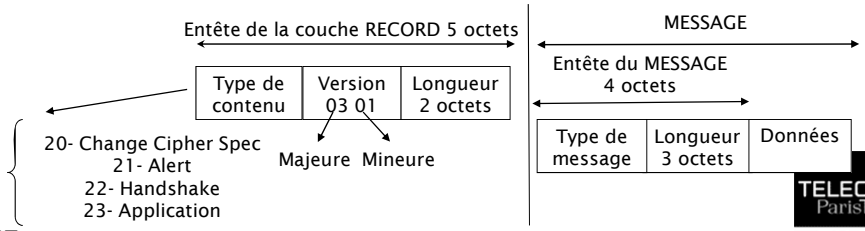
- **SSL défini par *netscape* et intégré au browser**
  - Première version de SSL testé en interne
  - Première version de SSL diffusé : V2 (1994)
  - Version finale, V3
- **Standard IETF, Transport Layer Security (TLS)**
  - Version 1.0 RFC 2246, 1.1 RFC 2346, 1.2 RFC 5246

SSL : Services

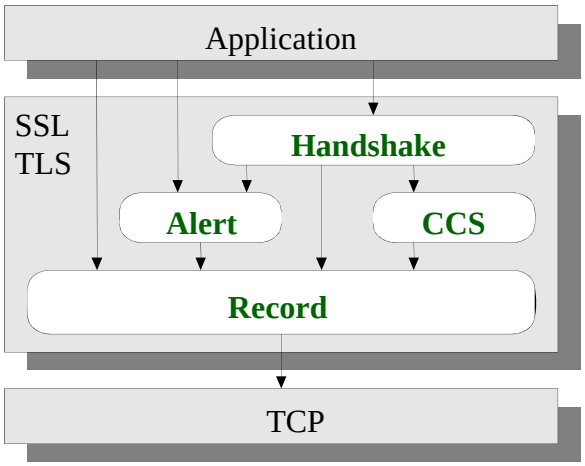
- **Authentification**
  - Serveur (obligatoire), client (optionnel)
  - Utilisation de certificat X509 V3, au cours d'une session full
- **Confidentialité**
  - Algorithme de chiffrement symétrique négocié, clé à l'établissement de la session
  - Intégrité, fonction de hachage avec clé secrète, généré à l'établissement de la session: HMAC(clé secrète, Message)
- **Non Rejeu**
  - Numéro de séquence

TLS: Entités logicielles & Binary Encoding Rules

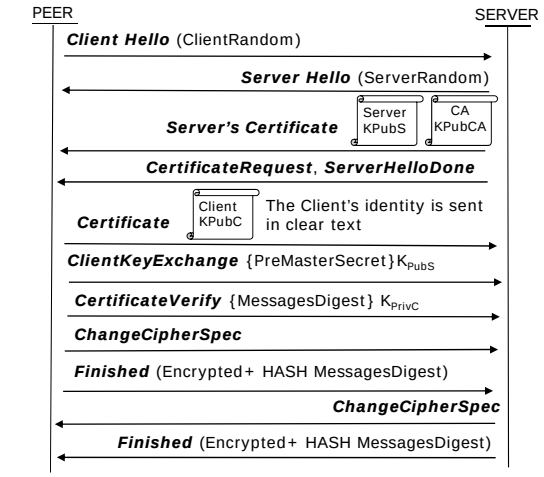




SSL/TLS : Protocoles



SSL/TLS, Dialogue de base



MasterSecret= PRF(ClientRandom, ServerRandom, PreMasterSecret,...)

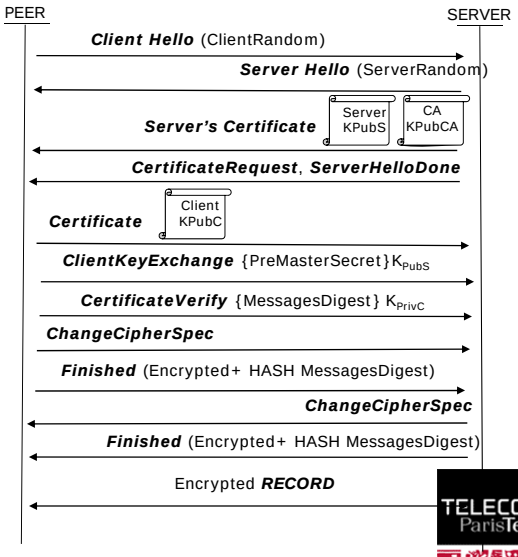
Keys = PRF(ClientRandom, ServerRandom, MasterSecret,...)

TLS, *full mode* standard

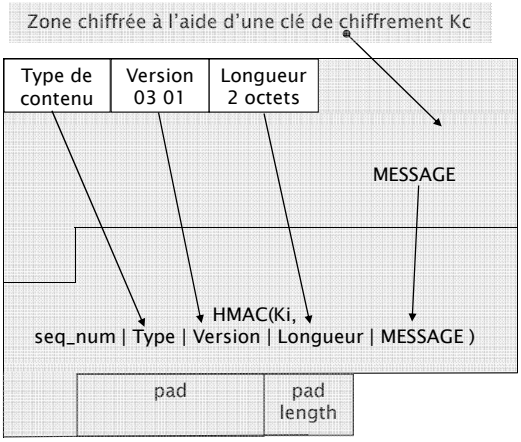
Mode calcul du pre_master_secret
-Un nombre aléatoire
-Un secret généré par un échange de DiffieHellman

master_secret =
PRF(pre_master_secret,
"master secret",
client_random | server_random)

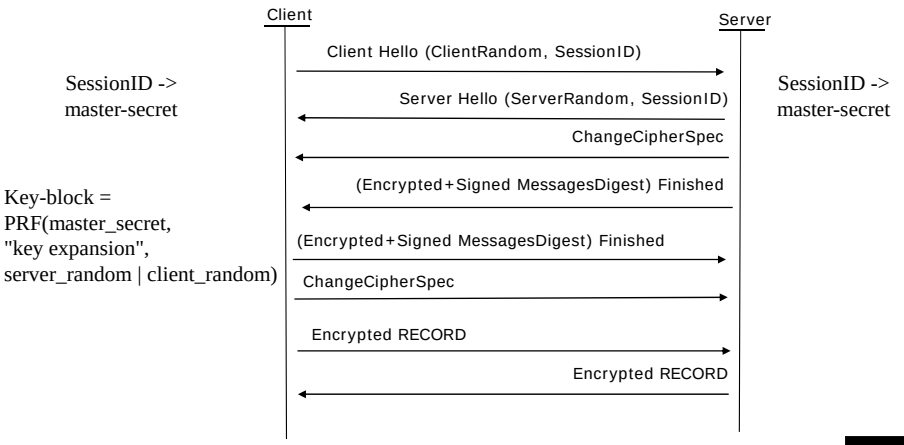
key-block =
PRF(master_secret,
"key expansion",
server_random | client_random)



Record Layer, en mode chiffré



TLS en mode *session resume*

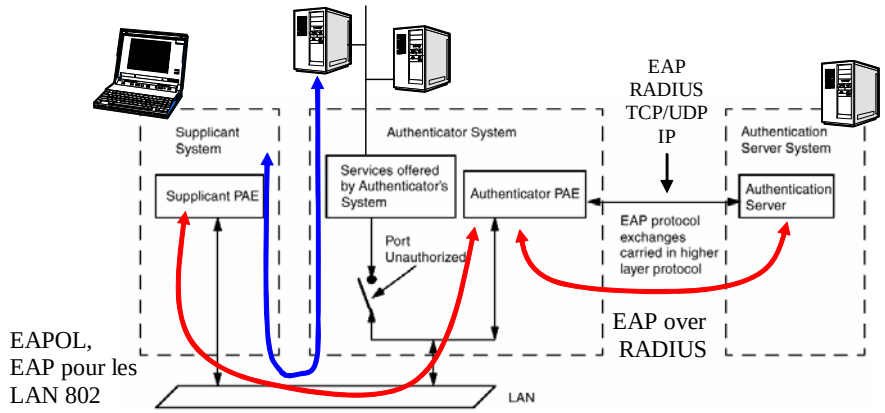


Authentification HTTP

- **HTTP digest, RFC 2617**
  - HTTP/1.1 401 Unauthorized WWW-Authenticate: Digest realm="testrealm@host.com", qop="auth,auth-int", nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093", opaque="5ccc069c403ebaf9f0171e9517f40e41"
  - Authorization: Digest username="Mufasa", realm="testrealm@host.com", nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093", uri="/dir/index.html", qop=auth, nc=00000001, cnonce="0a4f113b", response="6629fae49393a05397450978507c4ef1", opaque="5ccc069c403ebaf9f0171e9517f40e41"
  - **response= hash(secret || ":" || nonce)**
- **Scenario multimodale**
  - Collecte d'un formulaire HTTP collectant un login et un mot de passe
  - Gestion de session HTTPS authentifié à l'aide d'un cookie
    - Set-Cookie: valeur_du_cookie; path=/; expires=Wednesday, 09-Nov-99 23:12:40 GMT

IEEE 802.1x

Architecture d 'authentification 802.1x. 2/2



Network Port Authentication - 802.1x.

- Les trames émises par une station non authentifiée sont filtrées par le système d 'authentification.
- Les éléments de la procédure d 'authentification sont échangés via par le protocole EAP (*Extended Authentication Protocol*) .
- EAP est transporté par des trames 802 (EAP encapsulation over LAN) entre station et système d 'authentification.
- Le processus d 'authentification est conduit avec un serveur distant (et non par un AP).
  - Architecture centralisée.
- EAP est transporté par le protocole RADIUS (*Remote Access Dialing User Service*) entre système d 'authentification et serveur d 'authentification distant.

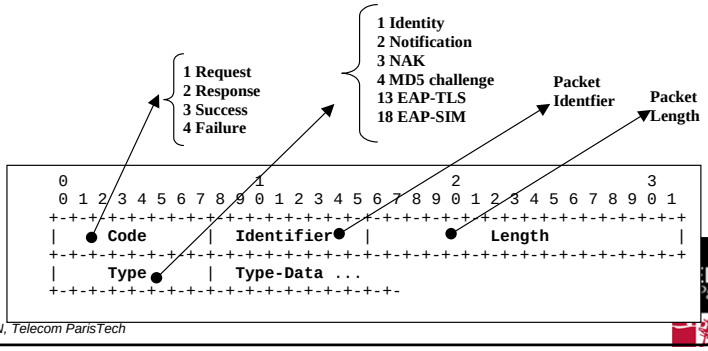
Le modèle 802.1x

- 1. L'identité du client (EAP_ID) détermine un serveur d'authentification (RADIUS). Elle est transmise au serveur RADIUS (RS), via le point d 'accès (AP).
- 2. Le processus d 'authentification se déroule entre le client (*supplicant*) et le serveur radius (RS). Le point d 'accès (*Authenticator*) se comporte comme un relais entre ces deux entités.
- 3. A la fin du processus d'authentification une clé unicast (ou clé maître MSK) est calculée par le client et le RS.
- 4. La clé MSK est transmise (chiffrée) par RS vers AP, à l 'aide du protocole RADIUS.
- 5. AP calcule alors une clé globale (WEP), il chiffre cette valeur par la clé SK, et la transmet au client (via trame EAPOL-Key).

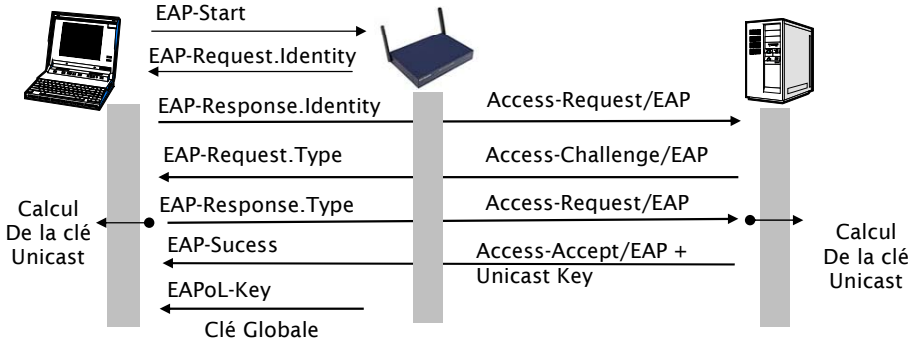
Le modèle EAP

Le protocole EAP.

- EAP est conçu pour transporter des scénarios d authentification.
  - Quatre types de messages, requêtes, réponses, succès, échec
- EAP, RFC 3748, "Extensible Authentication Protocol, (EAP)", June 2004.
  - *EAP-TLS*, RFC 2716, "PPP EAP TLS Authentication Protocol“, 1999.
  - *EAP-SIM*, RFC 4186, " Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM) ", 2006
  - *EAP-AKA*, RFC 4187, " Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA) “, 2006



802.1x

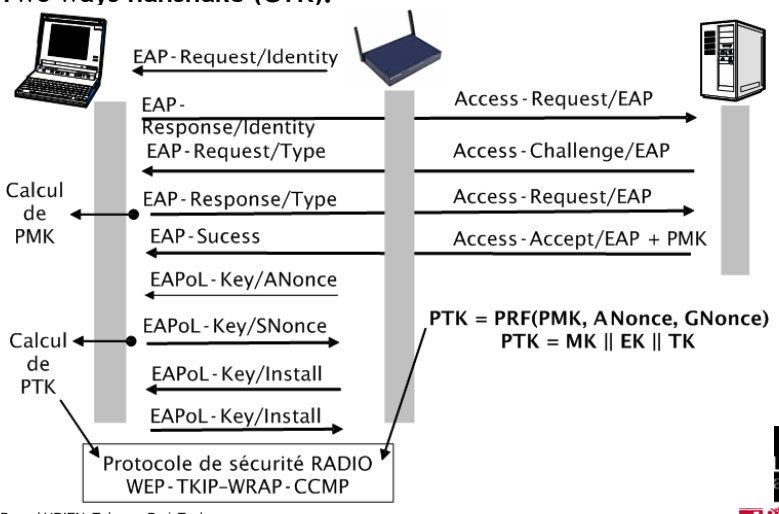


EAPoL key Descriptor

Descriptor Type – 1 octet	
Key Information – 2 octets	Key Length – 2 octets
Key Replay Counter – 8 octets	
Key Nonce – 32 octets	
EAPOL-Key IV – 16 octets	
Key RSC – 8 octets	
STA MAC Address – 6 octets	
GTK Length -2 octets	
Key MIC – 16 octets	
Key Data Length – 2 octets	Key Data – n octets

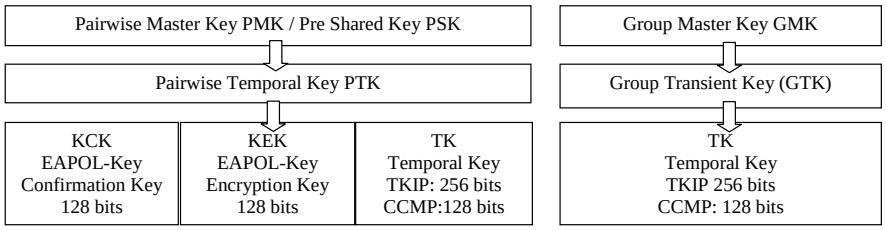
IEEE 802.11i : Distribution des clés

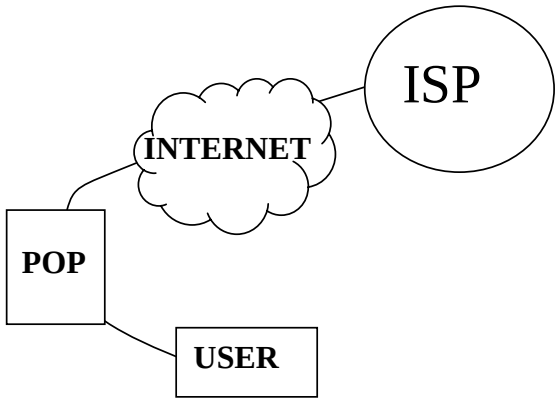
- Four ways handshake (PTK).
- Two ways hanshake (GTK).



802.11 i: Hiérarchie des clés

- PMK est déduite de l'authentification EAP.
- PSK est une alternative à PMK.
- GMK est une clé maître de groupe.





RADIUS

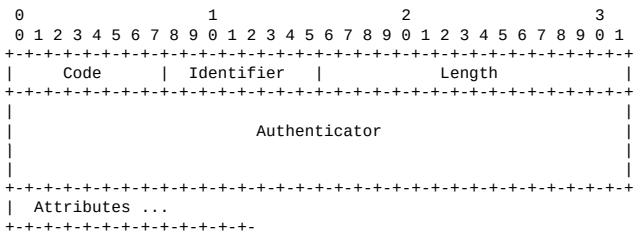
Le protocole RADIUS

- Permet d 'échanger des services entre fournisseurs de service.
- Network Access Server (NAS), est un serveur réalisant l 'authentification d 'un utilisateur désirant accéder au réseau (connexion PPP, accès sans fils…).
- NAS se comporte comme le client d 'un serveur d 'authentification RADIUS qui stocke les paramètres d 'authentification de l'utilisateur et ses droits.
- Les messages entre NAS et serveur RADIUS sont signés à l 'aide d 'un secret partagé et d 'une empreinte MD5.
- Le protocole RADIUS est également utilisé pour la facturation.

Sécurité Radius

- Le NAS génère des requêtes *Access-Request*, associées à un nombre aléatoire de 16 octets (le champ *Authenticator*). La réponse du serveur d'authentification est l'un des trois messages suivants
  - *Access-Challenge*
  - *Access-Reject*
  - *Access-Success.*
- Elle est signée par un nombre *Response Authenticator* (16 octets), une empreinte MD5 calculée à partir des données de la réponse, du champ *Authenticator* importé de la requête, et d'un *secret partagé* .
- De surcroît un paquet RADIUS comporte un attribut de signature (le *Message-Authenticator* **#80**), qui conformément à la RFC 2104, est déduit du secret partagé et du contenu du message.

Format des paquets RADIUS



- *Code*
  - 1 Access-Request
  - 2 Access-Accept
  - 3 Access-Reject
  - 4 Accounting-Request
  - 5 Accounting-Response
  - 11 Access-Challenge
- *Identifier*
  - Identifiant d'une requête et de la réponse associée.
- *Length*
  - Longueur totale du paquet en tête incluse (à partir du champ code).
- *Authenticator*
  - Un champ de 16 octets. C'est un nombre aléatoire dans de la cas d'un message access-request.
  - Pour les paquets access-accept, access-reject, accept-challenge, accounting-response c'est l'empreinte MD5 du message (en tête incluse, à partir de code) concaténé aux valeurs RequestAuthenticator et secret partagé.
  - ResponseAuth = MD5(Code||ID||Length||RequestAuth||Attributes||Secret)
- *Attributes*
  - Type, un octet, , l'identifiant d'un attribut (0..255)
  - Length, un octet, la longueur, champ type inclus (2,…255)
  - Value, la valeur de l'attribut

IKE / IPSEC

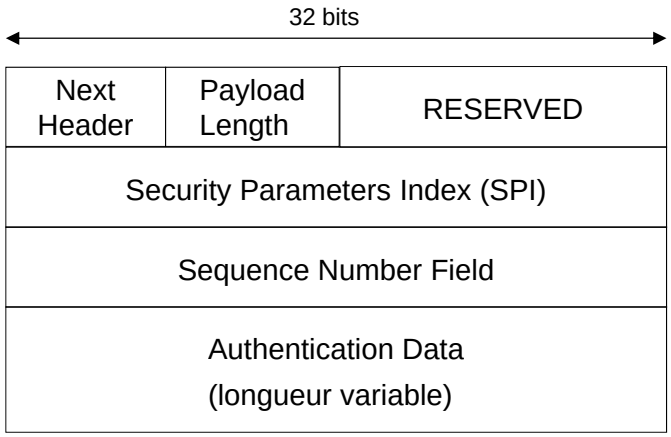
IPSEC: AH et ESP

- Deux en têtes spécifiques sont utilisés, AH (IP Authentification Header) et ESP (IP Encapsulating Security Payload).
- AH garantit l'intégrité et l'authentification des datagrammes IP, mais n'assure pas la confidentialité des données.
- ESP est utilisé pour fournir l'intégrité, l'authentification et la confidentialité des datagrammes IP.

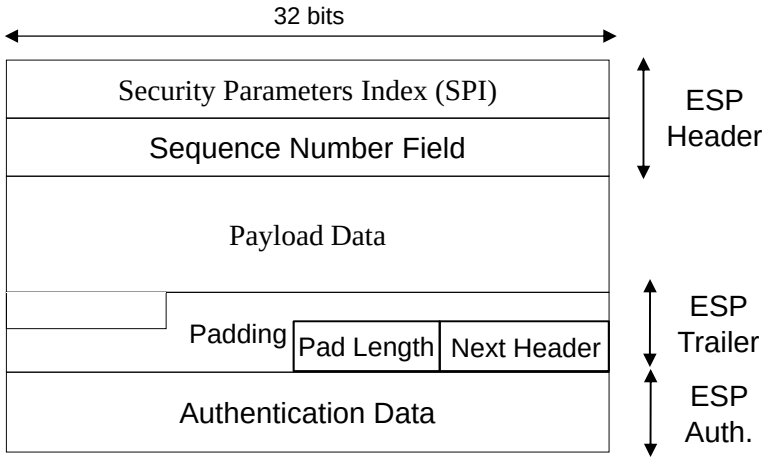
Security Association

- Ce concept est fondamental à la fois pour AH et ESP. La combinaison d'un SPI (*Security Parameter Index*) et d'une adresse de destination identifie de manière unique un SA particulier.
- Une association de sécurité inclue usuellement les paramètres suivant :
  - Un algorithme d'authentification (utilisé pour AH).
  - La (les) clé(s) utilisée(s) par l'algorithme d'authentification.
  - L'algorithme de chiffrement utilisé par ESP.
  - La (les) clé(s) utilisée(s) par l'algorithme de chiffrement.
  - Divers paramètres utiles à l'algorithme de chiffrement.
  - L'algorithme d'authentification utilisé avec ESP (s'il existe)
  - Les clés utilisées avec l'algorithme d'authentification d'ESP (si nécessaire).
  - La durée de vie de la clé.
  - La durée de vie du SA.
  - La ou les adresses de source du SA
  - Le niveau de sécurité (Secret, non classé ...)
- Le système hôte qui émet l'information sélectionne un SA en fonction du destinataire. L'association de sécurité est de manière générale *mono directionnelle*.

Authentication Header

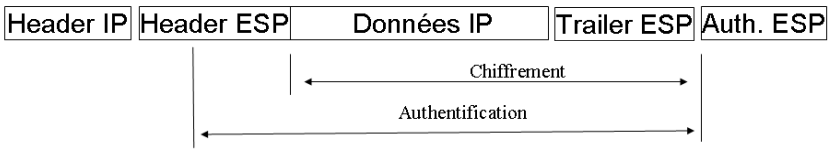


Encapsulating Security Payload

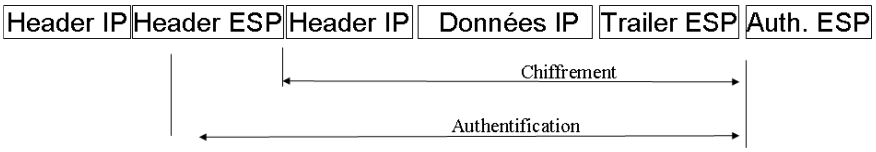


IPSEC: Mode Transport et Mode Tunnel

Mode transport



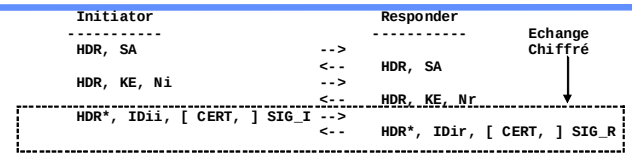
Mode tunnel



Au sujet de IKEv1

- Internet Key Exchange
- RFC 2409, 1998
- IKE PHASE 1 réalise une association de sécurité ISAKMP entre deux systèmes, qui protège les échanges de IKE phase 2
  - 4 modes, Main Mode, Agressive Mode, Quick Mode, New Group Mode
  - Plusieurs protocoles d'échanges de clés
    - Asymétriques, OAKLEY et SKEME
    - Symétrique (Pre-Shared-Key)
- IKE PHASE 2 réalise une association de sécurité pour des sessions IPSEC

IKEv1, Pre-Shared-Keys, Main Mode



For pre-shared keys:

SKEYID = prf(pre-shared-key, Ni_b | Nr_b)

The result of either Main Mode or Aggressive Mode is three groups of authenticated keying material:

SKEYID_d = prf(SKEYID, g^xy | CKY-I | CKY-R | 0)
SKEYID_a = prf(SKEYID, SKEYID_d | g^xy | CKY-I | CKY-R | 1)
SKEYID_e = prf(SKEYID, SKEYID_a | g^xy | CKY-I | CKY-R | 2)

and agreed upon policy to protect further communications. The values of 0, 1, and 2 above are represented by a single octet. The key used for encryption is derived from SKEYID_e in an algorithm-specific manner.

To authenticate either exchange the initiator of the protocol generates HASH_I(SIG_I) and the responder generates HASH_R(SIG_R)

where:

HASH_I = prf(SKEYID, g^xi | g^xr | CKY-I | CKY-R | SAi_b | IDii_b)
HASH_R = prf(SKEYID, g^xr | g^xi | CKY-R | CKY-I | SAi_b | IDir_b)

SAi_b is the entire body of the SA payload (minus the ISAKMP generic header), all proposals and all transforms offered by the Initiator.
CKY-I and CKY-R are the Initiator's cookie and the Responder's cookie, respectively, from the ISAKMP header.
g^xi and g^xr are the Diffie-Hellman public values of the initiator and responder respectively.

IKeV1, Phase II, Pre-Shared-Key, Quick Mode

Initiator — Responder

HDR*, HASH(1), SA, Ni [, KE] [, IDci, IDcr] -->
<-- HDR*, HASH(2), SA, Nr [, KE] [, IDci, IDcr]
HDR*, HASH(3) -->

HASH(1) = prf(SKEYID_a, M-ID | SA | Ni [| KE] [| IDci | IDcr])
HASH(2) = prf(SKEYID_a, M-ID | Ni_b | SA | Nr [| KE] [| IDci | IDcr])
HASH(3) = prf(SKEYID_a, 0 | M-ID | Ni_b | Nr_b)

KEYMAT = prf(SKEYID_d, protocol | SPI | Ni_b | Nr_b)

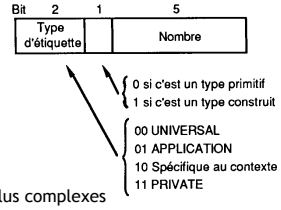
IDci, IDcr, identités, les adresses IP en fait.
M-ID, identifiant du message, extrait de l'en tête ISAKMP

Dessine moi un arbre

XML et ASN.1

Abstract Syntax Notation One - ASN.1

- *L'Abstract Syntax Notation* **One normalisé par l'ISO est une syntaxe de transfert de données et comporte les éléments suivants**
- *Les types primitifs*
  - BOOLEAN (1) vrai-faux
  - INTEGER (2) entier de longueur arbitraire
  - BIT STRING (3) liste de bits
  - OCTET STRING (4) liste d'octets
  - NULL (5)
  - ANY ensemble de tout type
  - OBJECT IDENTIFIER (5) nom d'objet



- *Les constructeurs*
  - Les types primitifs peuvent être combinés pour construire des types plus complexes
  - SEQUENCE (16) collection ordonnée d'éléments de types divers
  - SEQUENCE OF (16) collection ordonnée d'éléments de même type
  - SET (17) collection désordonnée d'éléments de divers type
  - SET OF (17) collection désordonnée d'éléments de même type.
  - CHOICE choix d'un type parmi une liste donnée.
- *Les étiquettes (tags)*
  - Les types sont dotés d'une étiquette (tag).Un tag comporte deux parties :
  - La classe : UNIVERSAL (00) APPLICATION (01) PRIVATE (11) CONTEXT-SPECIFIC (10)
  - un nombre entier
  - lorsque le tag apparaît sans classe ([nombre], [3]) la classe par défaut est CONTEXT-SPECIFIC
- **Exemples**
  - [UNIVERSAL 1] BOOLEAN
  - [UNIVERSAL 2] INTEGER
  - Le mot clé IMPLICITE placé avant une étiquette permet de supprimer le type des informations (exemple [PRIVATE 1] IMPLICITE INTEGER). Cette fonctionnalité est utilisée pour réduire la tai informations transférées.

ASN.1 - BER

- **Syntaxe**
  - Le symbole ::= décrit une règle de production
  - Le symbole | sépare les alternatives
  - Exemples :
    - Id_Object ::= OBJECT IDENTIFIER
    - Chiffre ::= 0 | 2 | 3
    - Class ::= UNIVERSAL | APPLICATION | PRIVATE | VIDE
- **Binary Encoding Rules (BER)**
  - Chaque valeur transmise contient 4 champs
    - l'identificateur (type ou étiquette).
    - la longueur en octets du champ de données
    - le champ de données
    - le fanion de fin de données si la longueur est inconnue.
  - Ce type de codage est dit TLV (*Type Longueur Valeur*).
- **Encodage de la Class**
  - 00 universal - 01 application - 10 context-specific - 11 private
- **P/C**
  - 0 primitif - 1 construit
- ***Encodage de l'identificateur***
  - pour un nombre <= 30
    - b8 b7 b6 b5 b4 b3 b2 b1
    - Class P/C nombre
  - pour un nombre > 30
    - b8 b7 b6 b5 b4 b3 b2 b1
    - Class P/C 1 1 1 1 1
    - b8 b7 b6 b5 b4 b3 b2 b1
    - 1 nombre
    - 0 nombre fin
- ***Encodage de la longueur***
  - Forme courte b8=0
  - Forme longue b8=1, b7...b1= longueur N en octets de la longueur + N octets.
  - Forme indéfinie 10000000..valeurs.. 00000000 00000000

Exemple

RequestPDU ::= [PRIVATE 3] IMPLICIT SEQUENCE{
responseRequired BOOLEAN,
requestID [PRIVATE 14] IMPLICIT INTEGER
parameters SEQUENCE {
request BIT STRING {startup(0),shutdown(1),status(2),echo(3)},
priority INTEGER DEFALUT 0},
additionalInformation OCTET STRING OPTIONNAL}

echo requestPDU ::= {
responseRequired TRUE, requestID 1324,
parameters {request(status,echo), priority 3},
additionalInformation '10101100'B}

E3 12
01 01 FF
CE 02 05 2C
30 06
03 01 02
02 01 03
04 01 AC

Document XML - Extensible Markup Language

- **Un document xml comporte deux parties**
  - un prologue
  - Un arbre d'éléments
    - Chaque élément est délimité par une balise de début et une balise de fin
    - Une balise de début peut comporter des attributs
- **Un document bien formé est décrit par une DTD (*Document Type Definition*)**
  - Structure de l'arbre
  - Attributs des éléments
- **Un espace de noms XML (xmlns)est identifié par une URI, et permet d'importer dans un document XML des éléments et des attributs**

Biométrie et carte à puce

Biométrie

- **La biométrie peut être utilisée pour le contrôle d'accès**
  - Généralement en complément d'une autre méthode, telle que une carte (à puce) d'accès
  - Le format CBEFF - *Common Biometric Exchange Formats Framework* - (norme NIST, NISTIR-6529A) décrit le transport d'empreintes biométriques
  - Par exemple l'empreinte digitale est représentée par une suite de détails caractéristiques (les minuties), dont les positions sont exprimées sous forme d'un liste de points (xi,yi)
- **Les caractéristiques d'un système biométrique peuvent être représentées par les probabilités de 4 événements**
  - p(x, accepté), p(x, refusé), p(~x, accepté), p(~x, refusé)
  - Les systèmes sont ajustés de tel sorte que le taux de faux positif soit de l'ordre du taux de faux négatif
    - p(x, refusé)=p(~x, accepté)
- **Un système biométrique suppose la vérification de l'hypothèse de la présence réelle du sujet**
  - *Liveness detection*

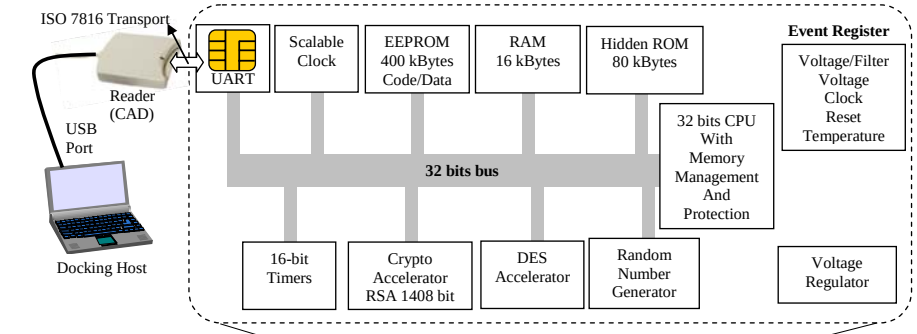
La carte à puce

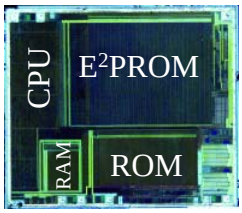
- **La carte à puce est un SPOM**
  - *Self Programmable One Chip Microcomputer*, inventé en 1980
    - 4 milliards de cartes/chips produits en 2008
  - Tamper resistant device
    - La sécurité est assurée à l'aide de contre-mesures physiques et logiques
  - Taille typique de la puce, 5mm x 5mm
- **Tailles mémoires**

ROM	28 - 256 Kbytes	Area Factor 1
E²PROM	64 - 128 Kbytes	Area Factor 4
RAM	4 - 8 Kbytes	Area Factor 16

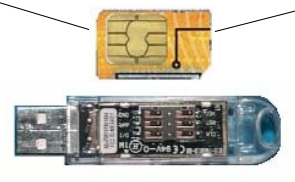
- **CPU**
  - Processeurs classiques de 8 bits, 1 - 3 MIPS (Clock 3.3 MHz)
  - Processeurs RISC 32 bits
- **Port de communication**
  - Lien série ISO7816 de 9600 à 230,400 bauds
  - USB (ISO7816-12), 10 Mbit/s
- **Binary Encoding rules**
  - Un entête de cinq octets
    - CLA INS P1 P2 P3
  - Une charge optionnelle de P3 (LC) octets
  - Une réponse optionnelle de P3 (LE) octets,
    - Qui se termine par deux octets de status SW

La carte à puce





1–5 $



One Time Password

One Time Password (OTP)

- Un OTP est un mot de passe éphémère
- Deux principes de générations
  - h(secret || horloge), exemple SecureID
  - h(secret || compteur), exemple HOTP
    - RFC 4226, "HOTP: An HMAC-Based One-Time Password Algorithm"
    - HMAC-SHA-1(clé,compteur)

SecurID: Principe

PIN — Token Code

Authentification forte à deux facteurs

Mot de passe dynamique, changeant toutes les minutes

10 digits ~ 33 bits

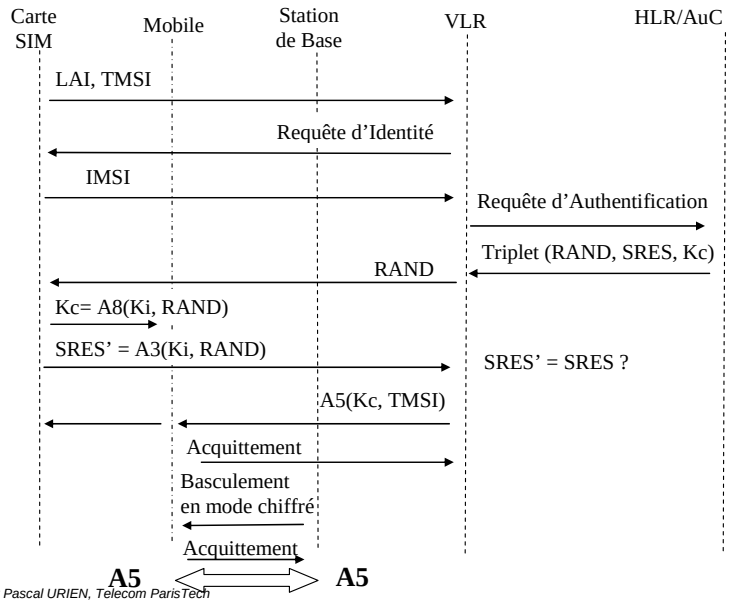
$2^{80} = 2^{10x8}$ = 24 digits =123456789012345678901234

Technologie

- **Une fonction de hash propriétaire, h=*ASHF***
  - TokenCode = h(SecretKey || TimeStamp)
    - SecretKey, 64 bits
    - TimeStamp, 32 bits
- **Architecture propriétaire**
  - Protocole ACE.
  - SOB, Secured By Obscurity.
- **Des failles publiées.**
  - "Fast Software-Based Attacks on SecurID", Scott Contini and Yiqun Lisa Yin, 2003
  - *Attaques publiées en 2^{40}*

Authentification GSM

Contrôle d'accès du GSM



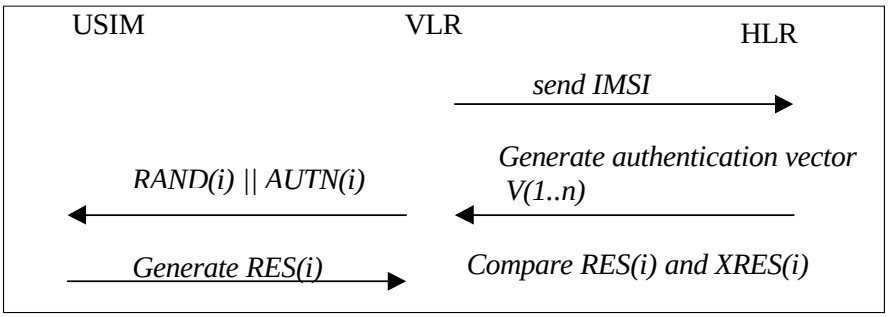
Authentification UMTS

Contrôle d'accès du GSM

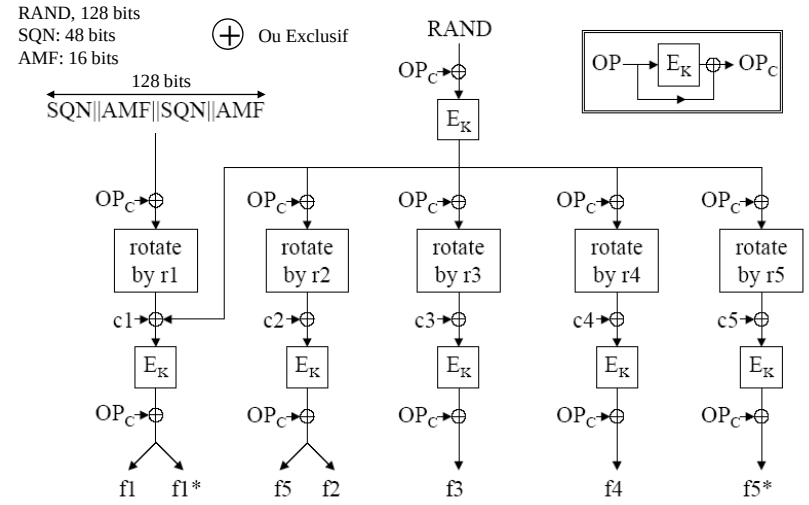
- La norme GSM 03.20 décrit de manière concise les principes de mise en œuvre des éléments de sécurité dans un environnent GSM .Une cellule ou un ensemble de cellules sont identifiées par une étiquette LAI (Location Area Identity).
- Un abonné dispose d'un couple de valeurs LAI-TMSI stocké dans son module SIM . Le mobile transmet au VLR les indications LAI et TMSI ; ce dernier tente de retrouver l'IMSI identifiant de manière univoque un abonné. En cas d'échec de cette opération, il délivre au mobile une requête d'identification qui récupère en clair l'IMSI mémorisé dans la carte SIM.
- A ce stade le VLR connaît l'IMSI de l'abonné. Il transmet au HLR (qui est généralement regroupé avec le bloc AuC) une demande d'authentification. Si le compte utilisateur est valide dans la base de donnée de l'opérateur, l'AuC produit une suite de valeurs connue sous l'appellation « triplet du GSM », trois nombres notés RAND, SRES et Kc.
- RAND est un nombre aléatoire de 16 octets, SRES (Signed RESponse, mot à mot réponse signée) est calculé par l'algorithme A3 associé à la clé Ki, soit SRES= A3(Ki, RAND).
- Kc est la clé utilisée pour le chiffrement des communications, elle est déduite de l'algorithme A8 associée à la clé Ki, soit Kc = A8(Ki, RAND).
- Ce mécanisme est une caractéristique originale du GSM ; en effet le VLR obtient de la part du HLR un ou plusieurs triplets d'authentification. Le VLR transmet au mobile un défi RAND. La carte SIM exécute alors la fonction A3 dont le résultat (SRES'=A3(Ki, RAND)) est renvoyé au HLR qui vérifie alors l'égalité entre SRES et SRES'.
- Le VLR choisit un nouveau TMSI, réalise son chiffrement avec l'algorithme A5 muni de la clé Kc, puis transmet ce paramètre au mobile qui effectue l'opération de déchiffrement.
- C'est la station de base dont dépend le mobile, qui décide du basculement mode chiffré (basé sur l'algorithme A5 muni de la clé Kc) des communications.

Authentification UMTS - Principes

- **Mutuelle Authentification**
  - Authentication and Key Agreement (AKA)
  - Cipher key (CK) and Integrity key (IK)



Authentification UMTS - MILENAGE

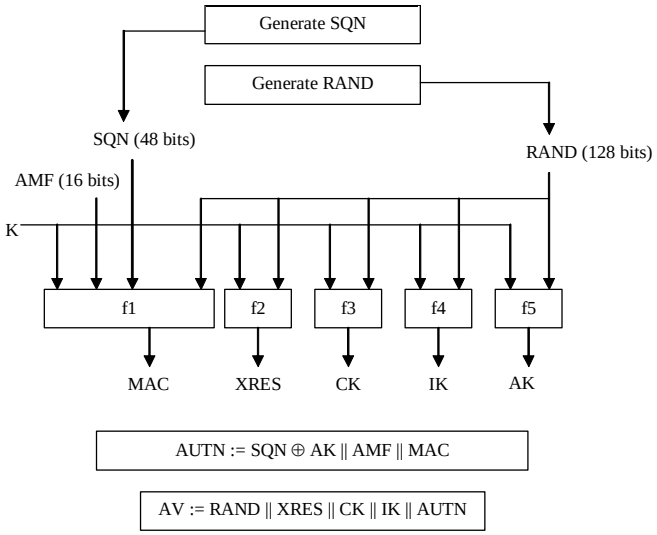


ri= 64,0,32,64,96 (rotation gauche

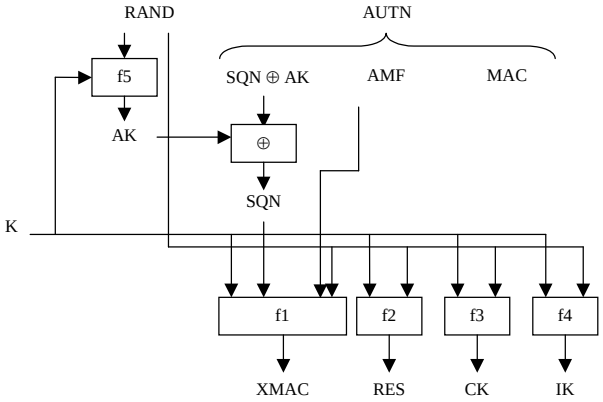
ci= 0,1,2,4,8

OP: clé OPérateur (128 bits) Ek: AES + clé 128bits

Authentification UMTS - Authentication Vector



Authentification UMTS - USIM



Verify MAC = XMAC

Verify that SQN is in the correct range