Cours de sécurité

PROTOCOLES DE SECURITE

Gérard Florin / Stéphane Natkin

- CNAM / Laboratoire CEDRIC -

Protocoles de sécurité

Plan du chapitre

Introduction
Confidentialité
Intégrité
Authentification
Protection (contrôle d'accès)
Conclusion

Protocoles de sécurité

Confidentialité

Solutions de base

Echange de clés

Diffie-Helman

La mise en oeuvre des chiffrements par blocs

Protocoles de confidentialité

Introduction – Solutions de base

Cryptographie symétrique (à clés secrètes)
Cryptographie asymétrique (à clés publiques)
Solution mixte

Confidentialité : solutions de base

- **Confidentialité :** donner un accès en lecture à des données uniquement aux entités autorisées.
- **Repose sur l'existence d'un bon chiffre.**

- **Cryptographie à clé privée (Ek,Dk) k clé secrète**
  - Pour un message M, chiffrement Ek(M).
  - Seuls les détenteurs de k savent chiffrer ou déchiffrer.
- **Cryptographie à clé publique Ek , clé privée Dk‘**
  - Pour un message M on envoie Ek(M).
  - Tout le monde connaît Ek => tout le monde peut chiffrer.
  - Seul le détenteur de k’, peut déchiffrer avec Dk’.

- **Problème de distribution des clés.**
- **Problème de mise en œuvre des chiffres sur les blocs successifs.**

Protocoles de confidentialité

Protocoles d'échange de clés

Solutions à clés secrètes et clés publiques
Solution de Diffie-Hellman

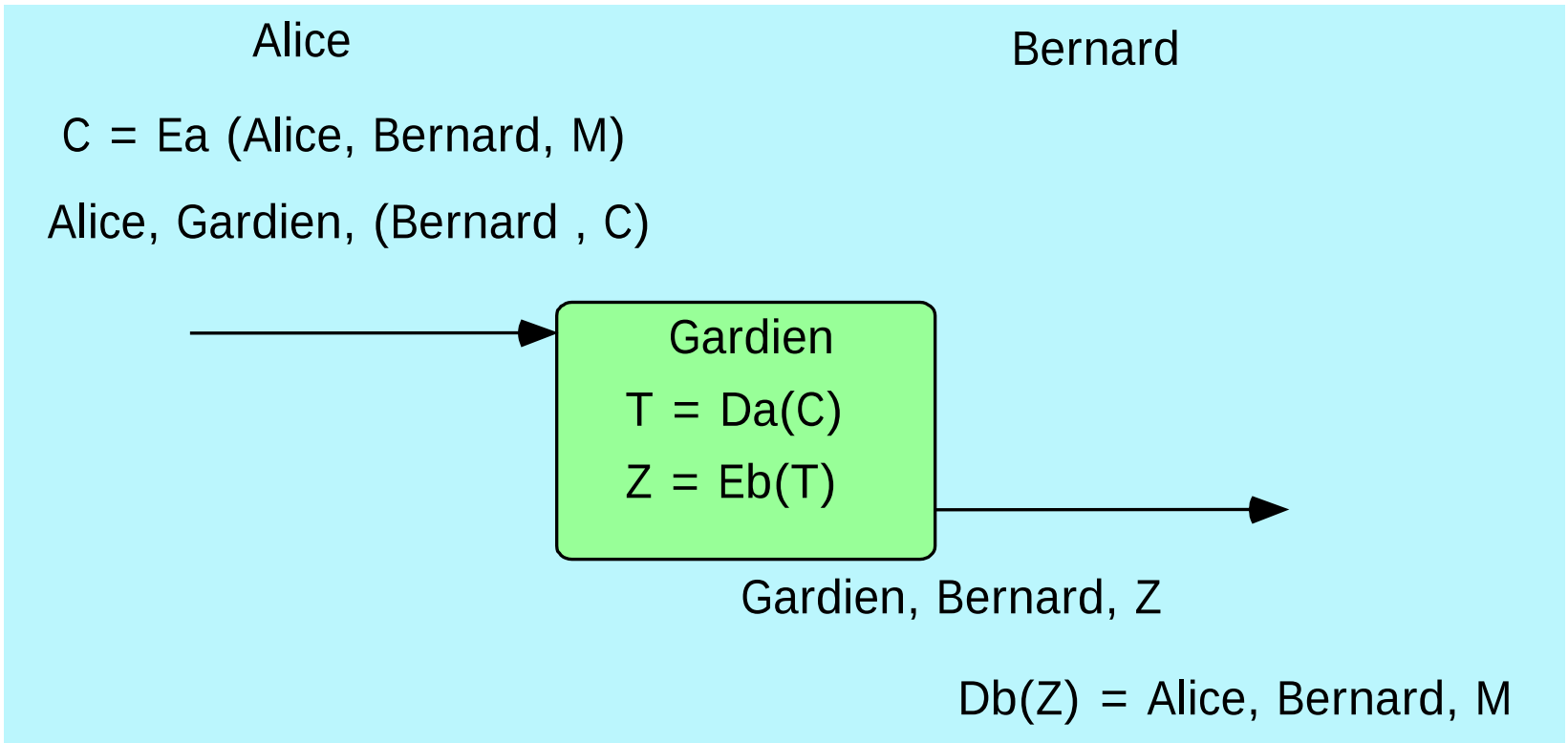
Confidentialité : Combinaison des chiffres à clés publiques et secrètes

- **Problèmes de performances:**
  - La cryptographie à clé publique est **beaucoup trop lente**.
  - Personne (ou presque) **ne chiffre en confidentialité en clés publiques.**

- **Solutions mixtes** :
  - **Echanger** confidentiellement des **clés de session** en début de session
    - **En cryptographie à clé secrète.**
    - **En cryptographie à clé publique.**
    - **Par le protocole de Diffie-Hellman.**
  - **Communiquer** ensuite rapidement confidentiellement au moyen de la clé de session en utilisant la cryptographie à **clé secrète.**

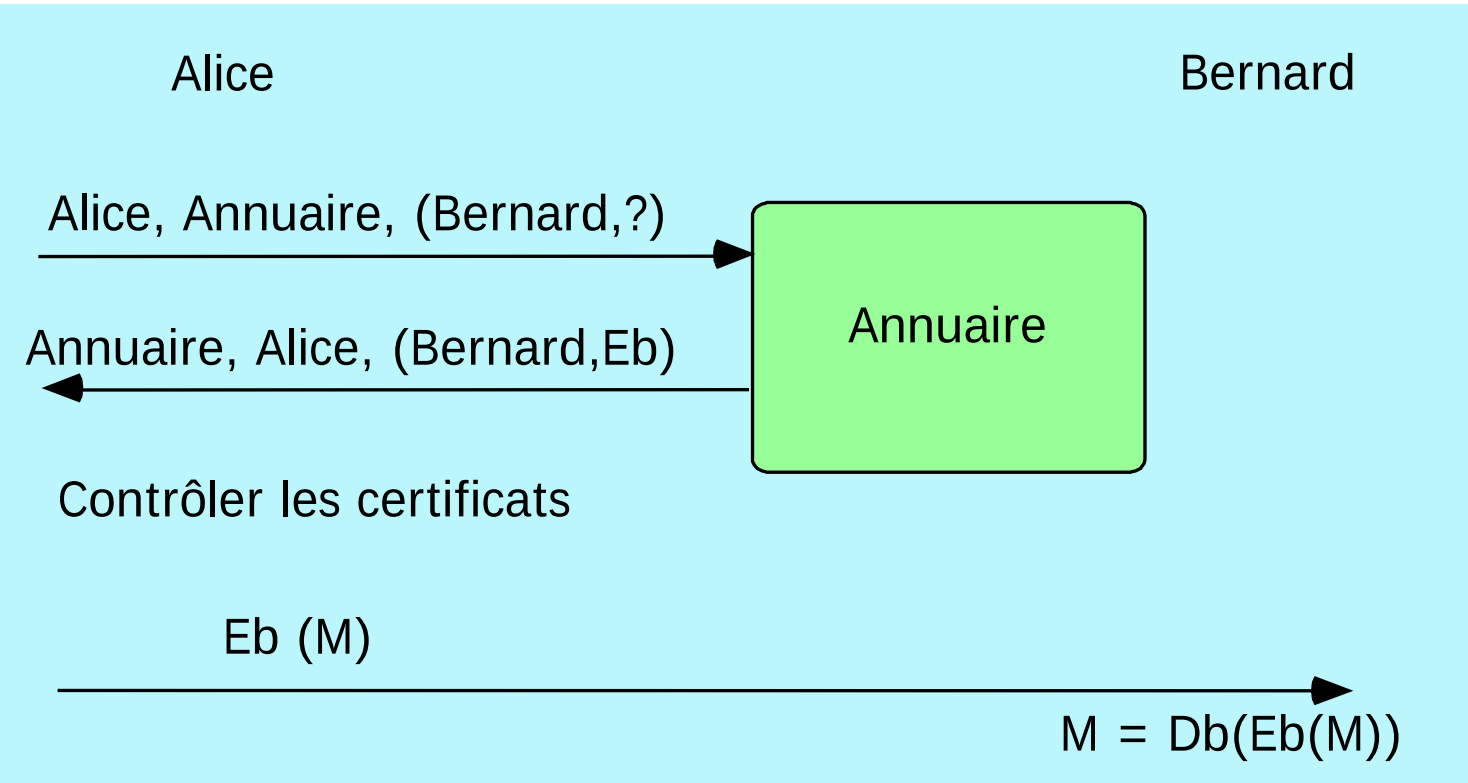
Confidentialité : Chiffre symétrique

- **Confidentialité générale=> applicable à l'échange de clé de session**
- **Cas d'entités nombreuses sans relations préalables.**
- **Alice veut transmettre à Bernard** sans connaître la clé de Bernard.
- **Utilisation d'un gardien de clés** qui connaît les clés secrètes de Alice et de Bernard (pour protéger la divulgation des clés).



Confidentialité : Chiffre asymétrique

- **Confidentialité générale => applicable à l'échange de clé de session**
- **Utilisation d'un annuaire** : Alice peut obtenir la clé publique de Bernard en interrogeant l'annuaire.
- **Notion de certificat** : une structure de données stockée dans l'annuaire contenant une clé publique et d'autres informations (contrôle d'intégrité).



Notion de certificat : Exemple de certificat X509

Certificate ::= SIGNED {
SEQUENCE{ **version [0]** **Version DEFAULT v1,**
serialNumber **CertificateSerialNumber,**
signature **AlgorithmIdentifier,**
issuer **Name,**
validity **Validity,**
subject **Name,**
subjectPublicKeyInfo **SubjectPublicKeyInfo,**
issuerUniqueIdentifier [1] IMPLICIT UniqueIdentifier OPTIONAL
}
}

Version ::= INTEGER { v1(0), v2(1), v3(2) }
CertificateSerialNumber ::= INTEGER
AlgorithmIdentifier ::= SEQUENCE {
algorithm ALGORITHM.&id({SupportedAlgorithms}),
parameters ALGORITHM.&Type ({SupportedAlgorithms}{@algorithm}) OPTIONAL }
SupportedAlgorithms ALGORITHM ::= { ... }
Validity ::= SEQUENCE { notBefore UTCTime, notAfter UTCTime }
SubjectPublicKeyInfo ::= SEQUENCE { algorithm AlgorithmIdentifier, subjectPublicKey BIT STRING }
Time ::= CHOICE { utcTime UTCTime, generalizedTime GeneralizedTime }
SIGNED { ToBeSigned } ::= SEQUENCE { **toBeSigned ToBeSigned,**
encrypted ENCRYPTED { HASHED {ToBeSigned }}}

L'échange de clés de Diffie-Hellman : Introduction

- **1) Diffie-Hellman** : une première tentative pour trouver un chiffre à clés publiques.
- **2) En fait un protocole uniquement utilisable pour le partage d'un secret** : pour l'échange d'une clé secrète.
- **3) Entre deux entités qui ne se connaissent pas.**
- **4) Communicant** : au moyen d'un réseau non sécurisé.
- **5) Solution utilisée en pratique:** de façon assez significative.

Confidentialité : Diffie-Hellman
Le protocole de base (1976)

Partie publique : p , g

- **Choix d'un grand nombre entier premier p :** on travaille dans le groupe multiplicatif des entiers inférieurs à p (Z_p^*)
- **Choix d'un nombre g :** générateur des entiers inférieurs à p (g est un élément primitif dans Z_p^*)

  $\{g^0, g^1, g^2, \ldots, g^{P-1} \pmod p\} = \{0..p-1\}$

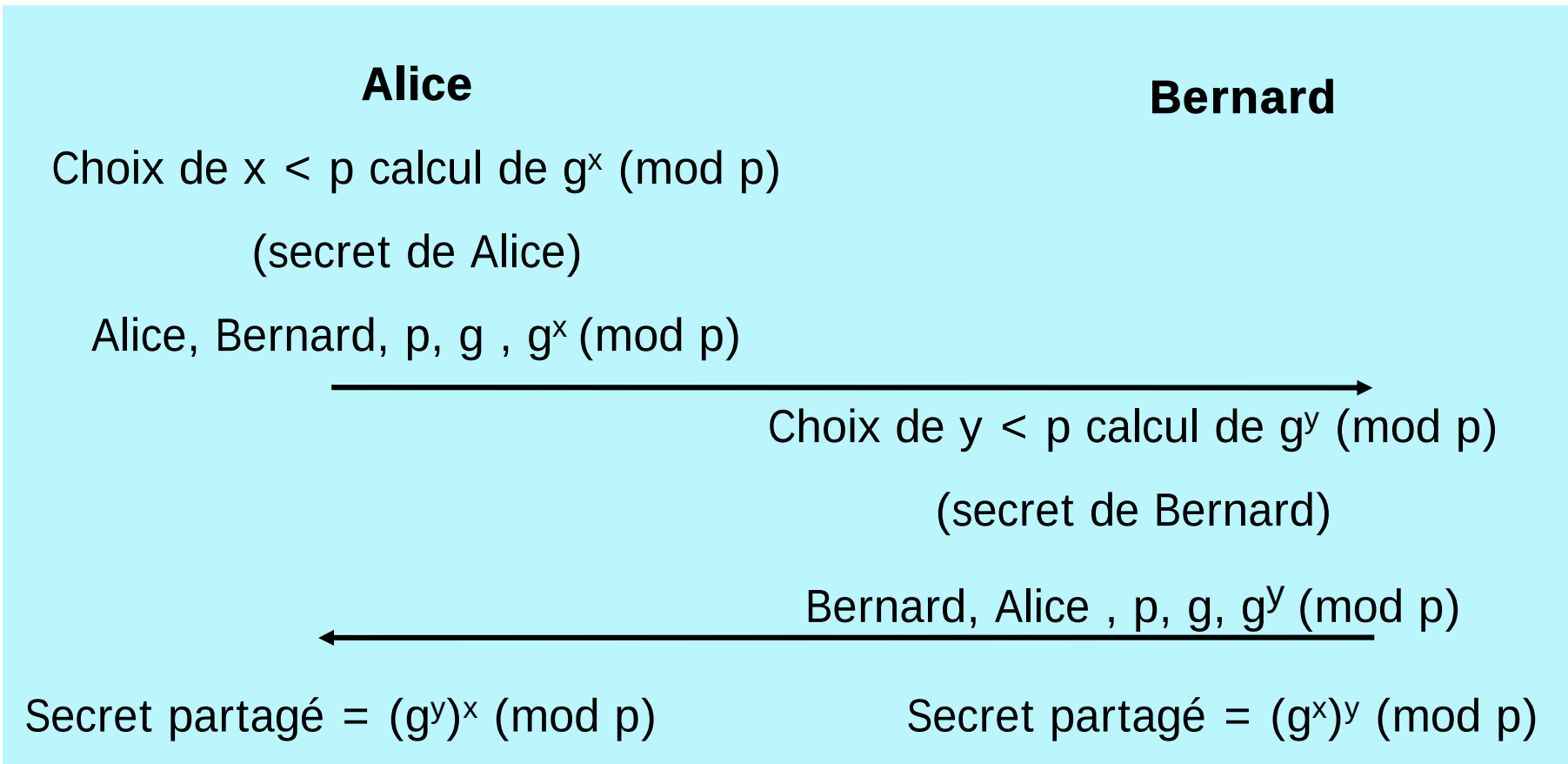
Partie secrète : x , y

- **Pour Alice :** choix d'un nombre aléatoire $x < p$.
- **Pour Bernard :** choix d'un nombre aléatoire $y < p$.

- **Résistance aux écoutes** : élévation à la puissance modulo p.
- **Complexité de la solution du problème du logarithme discret** : dans les entiers.

Confidentialité : Diffie-Hellman
Les échanges du protocole de base

Un protocole en deux messages



Conclusion : Diffie-Hellman

- **1) Principal problème l'attaque du milieu :** Eve peut se faire passer pour Alice auprès de Bernard et pour Bernard auprès de Alice parce qu'il n'y a aucun moyen d'authentification de prévu.
  - **Seule solution** : rajouter une structure de données pour valider l'intégrité des messages de Alice et de Bernard (une signature ou un MAC).
  - **Utilisation indispensable d'une clé** : pour authentifier Alice et Bernard.
  - **Echanger des clés confidentiellement** : nécessite donc quand même la possession d'une clé pour assurer l'intégrité.
- **2) Différents autres problèmes de mise en œuvre :**
  - **Choix de p** : nombre premier sur de grande taille. On suggère la forme 2q+1 avec q premier (à vérifier que p est bon).
  - **Choix de g** : génère bien tous les entiers (à vérifier aussi, sinon on peut simplifier considérablement les efforts d'un attaquant).
  - **Utiliser un sous groupe plus petit $0<x<q$** : pour simplifier les calculs d'exponentiation (bien choisir q).
- **3) Utilisation :** IPSEC/IKE Internet Key Exchange Protocol (RFC 2409)

Protocoles de confidentialité

Mise en oeuvre des chiffrements à clés secrètes par blocs

Bourrage.
Le mode basique par blocs : ECB ‘Electronic Code Book’.
Chaînage de blocs chiffrés : CBC ‘Cipher Block Chaining’.
Les vecteurs d’initialisation.
Génération de flots de clés: OFB ‘Output Feedback Block’.
Génération de flots de clés en mode compteur:
CFB ‘Counter Feedback Block’.

Chiffrement par blocs : Bourrage ou Complémentation ('Padding')

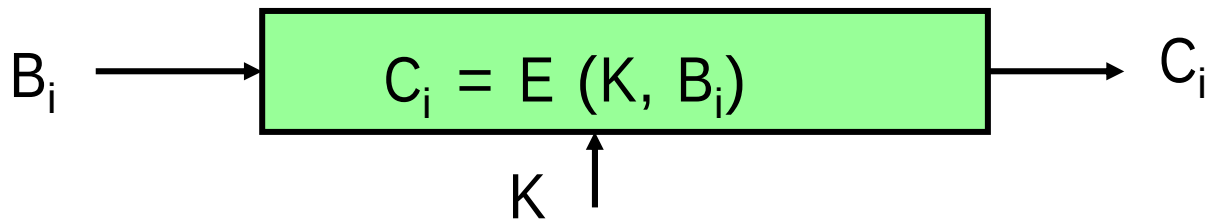
- **Chiffrement par blocs de b octets**
  - **Problème** : si les messages à transmettre M ne sont pas de longueur l(M) fixe et multiple de b (k.b)
  - **Solution :** Compléter les informations à k.b **par du bourrage**
  - **Selon une méthode réversible** : permettant d'enlever le bourrage.
- **Solution réversible:** rallonge **obligatoirement** chaque message
  - **Structuration par octets**: au moins un octet en plus.

Chiffrement par blocs : Bourrage Quelques solutions

- **Solutions pour des messages structurés par octets.**
- **Solution 1 :**
  - **Rajouter un octet de valeur connue fixe** : exemple 255.
  - **Rajouter k.b–(l(M)+1) octets** de bourrage de valeur fixe : exemple 0
- **Solution 2** :
  - **Solution pour des longueurs de bourrage en octets <255.**
  - **Déterminer le nombre d'octets n** de bourrage (avec minimum 1).
  - **Rajouter n octets de bourrage de valeur égale à n**.
- **Solution 3** :
  - **l(M) nombre d'octets** significatifs du message **codé sur m octets :** exemple 8 octets (64 bits).
  - **Rajouter** k.b - l(M) - m octets de bourrage puis la **longueur sur m octets** des données significatives.

Chiffrement par blocs : Le mode ECB ('Electronic Code Book')

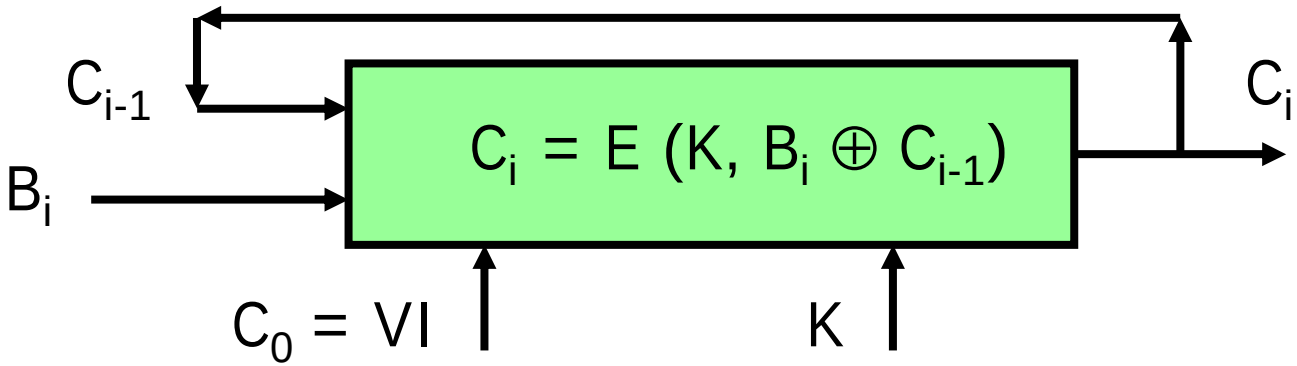
- **Application basique du chiffrement par blocs** : chiffrer séparément chaque bloc.
  - $M = B_1, B_2, \ldots, B_i, \ldots B_{Nblocs}$ $\quad C_i = E(K, B_i)$ i=1, ..., Nblocs



- **Problème général d'un chiffre :** laisser filtrer le moins possible d'informations.
- **Problème de cette solution:** des blocs identiques du message en clair sont chiffrés de la même façon
  - **On peut inférer des informations sur la nature du message en clair : Entêtes fixes, postface fixe** (bourrage de structure connue) Chaînes de caractères **répétées dans un texte** (balises, selon le sujet).
  - **On peut changer l'ordre des blocs.**
- **Mode basique** de chiffrement par blocs: **non recommandé en sécurité.**

Chiffrement par blocs : Le chaînage de blocs CBC 'Cipher Block Chaining'

- **Idée principale : diversifier le résultat de chiffrement de chaque bloc en le faisant dépendre du bloc précédent**.
  - $M = B_1, B_2, \ldots, B_i, \ldots B_{Nblocs}$
  - $C_i = E(K, B_i \oplus Ci\text{-}1)$ i=1, ..., Nblocs



- **Nécessité de choisir** une valeur initiale VI (un premier bloc chiffré, **V**ecteur d'**I**nitialisation) **qui amorce le processus**.
- **Exemples** : DES-CBC , 3DES-CBC, AES-128-CBC (en IPSEC)

Chiffrement par blocs en mode CBC : Choix du vecteur d'initialisation

1) Vecteur d'initialisation constant

- **Problème sur le premier bloc** dont la valeur en clair est souvent fixe et dont on va donc connaître le chiffre.

2) Vecteur d'initialisation numéro de séquence.

- **VI un compteur** : le numéro de séquence du message.
- **Si l'on part toujours de 0** : encore des faiblesses.
- **Sur n bits :** on repasse par les mêmes valeurs.

3) Vecteur d'initialisation aléatoire.

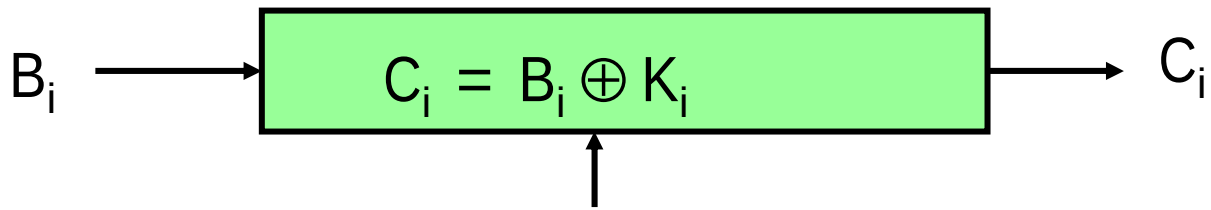
- VI donné par un bon générateur de nombres aléatoires.
- Il faut communiquer VI en clair avec le premier bloc qui est donc différent : $VI=C_0$, $C_1 = E(K, B_1 \oplus C_0)$
- Une solution qui gène considérablement un attaquant.

Exemple de vecteur d'initialisation basé sur un nonce (B. Schneier)

- **VI est un nonce** '**N**umber Used **Once'** : un nombre aléatoire **(imprévisible)** qui n'est jamais utilisé **deux fois.**
- **Etape 1 : Fabrication d'une estampille**
  - **Identifiant unique de message** => Estampilles de Lamport
  - **Numéroter chaque message** à chiffrer par blocs (par exemple sur 32 bits comme en TCP)
  - **Associer au numéro de message l'identifiant unique de l'émetteur** pour former l'estampille.
- **Etape 2** : **Nonce obtenu par chiffrement de l'estampille**
  - VI (vecteur d'initialisation) = Nonce = E(K, Estampille)
  - S'il y a des problème de collisions avec E(K, ...) => utiliser un hachage
- **Règle de sécurité**: faire circuler l'estampille en clair, ne pas faire circuler le nonce en clair (le vecteur d'initialisation).
- **Solution considérée comme sécuritaire.**

Chiffrement par bloc: Chiffrement à flot de clés OFB 'Output Feedback Block'

- **Idée générale**: utiliser un chiffre pour **générer un flot de clés** successives formant une **séquence pseudo aléatoire qu'on utilise comme masque (ex RC4)**.

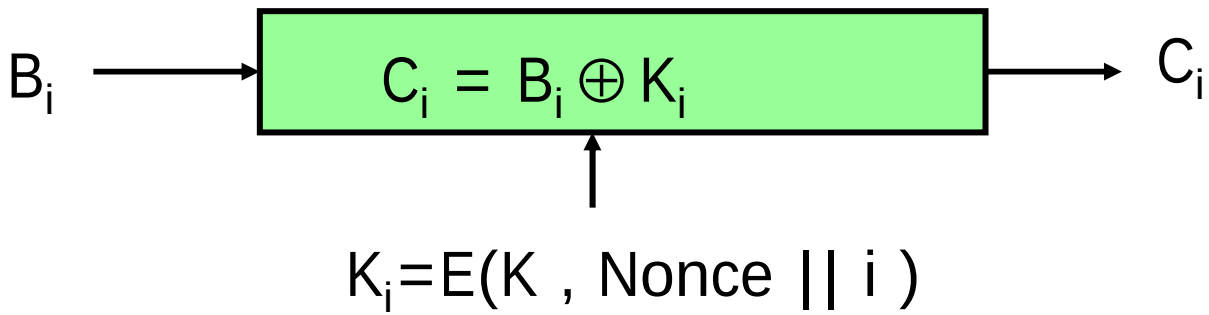


$K_0 = VI$, ..., $K_i = E(K , K_{i-1})$, ... $K_n = E(K , K_{n-1})$

- **On doit choisir un VI selon les mêmes possibilités qu'en mode CBC** : avec une préférence pour un VI défini à partir d'un nonce (le plus aléatoire possible et non réutilisé).
- **Variante de OFB le mode CFB 'Cipher Feedback Block')** : c'est la précédente donnée chiffrée qui sert à générer la clé suivante: $K_i = E(K , C_{i-1})$.

Chiffrement à flot en mode compteur CFB 'Counter Feedback Block'

- **Une autre variante du mode OFB :** chaque clé de bloc est construite à partir d'un nonce et du numéro du bloc :



- **Nonce :** donne une valeur unique pour tous les blocs d'un message.
- **Indice i du bloc :** donne une valeur unique pour chaque blocs (|| symbole pour la concaténation).
- **Méthode jugée très sécuritaire (recommandée)**: selon les qualités du chiffre E les clés Ki sont pseudo aléatoires et imprévisibles à cause de E, de K et du nonce.

Protocoles de sécurité

Protocoles d'intégrité

Introduction: définition de base
Signatures numériques
Codes d'authentification de messages (MAC)

Problème d'intégrité : Définition

- **1) Intégrité : contrôler l'accès en écriture**
  - Autoriser seulement certaines entités à écrire.
- **2) Donnée dont on peut vérifier l'intégrité**: donnée possédant un code de détection de modification.
  - Système d'intégrité : deux facettes.
  - Fonction de construction du code de détection.
  - Fonction de vérification du code.
- **3) Terminologie fréquente :** donnée signée.
- **4) Fonctionnement conjoint** : intégrité et authentification
  - Un document est intègre parce que son code de détection de modification a été calculé par une entité de confiance.

Problème d'intégrité : Différences de nature Confidentialité/Intégrité

- **Confidentialité** :
  - **Basée sur la génération de données chiffrées par les usagers autorisés** détenteurs d'un secret:
    - **Faire disparaître les redondances** du texte en clair: idée de diffusion
    - **Pas de corrélations possibles** entre le clair et le chiffré: idée de confusion
    - Mais on peut toujours fabriquer une suite binaire qui a la longueur d'un message chiffré et qui peut être pris pour un message chiffré.
- **Intégrité :**
  - **Ajouter des redondances** à un message en clair pour vérifier que le clair n'a pas été modifié.
    - **Faire apparaître des redondances** pour vérifier l'intégrité
    - **Etablir des corrélations** entre le code de détection de modification et le clair.
    - On peut toujours accéder au contenu du message en clair.

Protocoles d'intégrité

Les signatures numériques

Introduction

La signature RSA

La signature DSA

Notion de signature numérique : Définition et propriétés principales

Signature numérique : une structure de données associée à un document pour le garantir en intégrité mais aussi en authentification et en non répudiation.

- **P1) Une signature concerne un seul document** :
  - Une signature **n'est pas réutilisable**
- **P2) Intégrité 1 :** Un document signé ne peut être ensuite modifié
  - Même très partiellement.
- **P3) Intégrité 2 :** Une signature est vérifiable par n'importe qui
  - Sans qu'on ait à demander d'autorisation.
- **P4) Authentification :** Une signature authentifie le signataire:
  - Seul le signataire **peut avoir signé**.
  - Une signature **ne peut-être imitée**
- **P5) Non répudiation:** Une signature ne peut-être reniée.
  - C'est une preuve que le signataire a délibérément signé le document.
- **P6) Performances :** Devrait être facile à calculer et à vérifier
  - En fait assez souvent de mauvaises performances.

Signatures numériques : Système de signature numérique

- **1) Signataires** (en nombre limité) **= >** existence d'un système de chiffrement avec clés, associé à la signature numérique.
  - **Conseil** : ne pas utiliser les mêmes clés pour le chiffrement en confidentialité et pour la vérification d'authentification (affaiblissement d'un mécanisme par l'autre).
- **2) Algorithme de signature :** Fonction de construction de la structure de données qui garantit l'intégrité d'un document et authentifie l'émetteur.
  - (M , K) → S = sig (K , M)
- **3) Algorithme de vérification de signature** : calcul de prédicat de vérification de l'intégrité de la donnée et de l'authentification de l'émetteur.
  - ver (M , S) → vrai ou faux

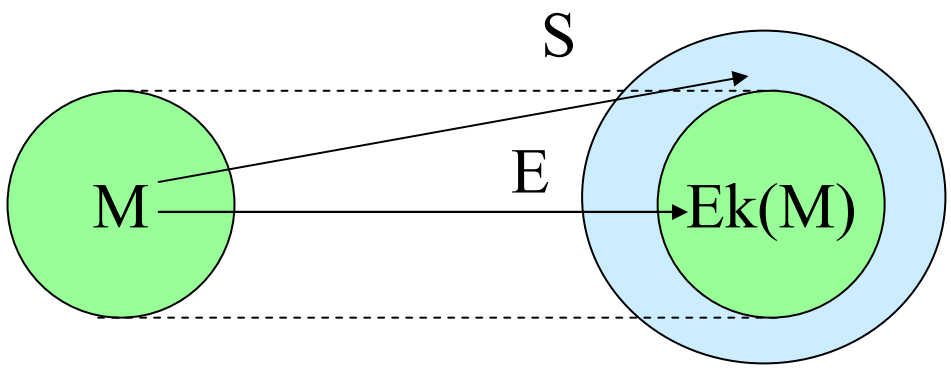
    ver (M , S) : S = sig (K , M)

Signatures numériques : Utilisation des chiffres à clés publiques

- **1) Signature numérique :** utilisation d'un chiffre asymétrique (à clé publique k chiffrement Ek et clé privée k' déchiffrement Dk').
- **2) Idée de base :** signer avec la clé privée, vérifier avec la clé publique.
  - sig(M) := Dk'(M, redondance(M), ...)
  - ver(sig(M)) : Ek(sig(M)) = M , redondance(M), ...
- **3) Propriétés :** si le chiffre à clé publique et la fonction de redondance sont sécuritaires on peut satisfaire les propriétés d'une signature numérique.
  - En particulier: **seul l'émetteur peut signer (Dk').**
  - Tout le monde **peut vérifier une signature (Ek)**.

Signatures numériques : Signatures avec recouvrement de message

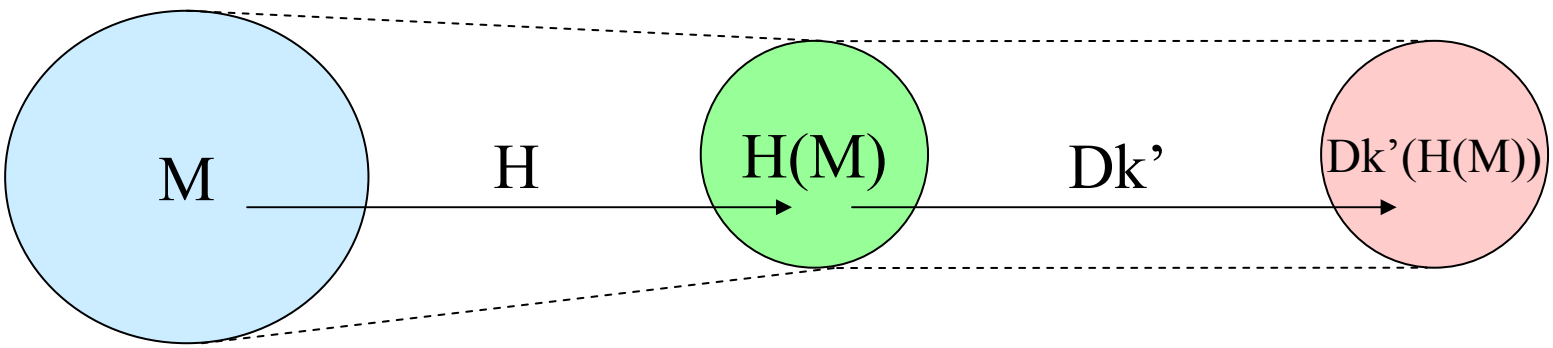
- **1) Solution de la signature 'avec recouvrement'**: chiffrer tout le message => Pour mémoire car très rare.
- **2) Ajouter des informations formant redondance:** pour s'apercevoir des modifications éventuelles.
  - sig(M) : transmettre Ek(M) et S = f(Ek(M))
  - ver(sig(M)): S=f(Ek(M))
  - Récupérer M à partir de Ek(M) : Dk'(Ek(M))



- **3) Problème**: La signature avec recouvrement est **coûteuse** car elle suppose le **chiffrement complet de M**.

Signatures numériques : Signatures avec appendice (‘appendix’)

- **Solution de la signature ‘avec appendice’**: chiffrer un hachage du message.
- **Solution** préconisée dans tous les protocoles cryptographiques
- **Utilisation d’un chiffre à clés publiques et d’une fonction de hachage H:** résumé du message pour s'apercevoir des modifications éventuelles.
  - sig(M) : on transmet M et appendice = Dk’(H(M))
  - ver(sig(M)): H(M) = Ek(appendice)= Ek(Dk’(H(M)))



Protocoles de signatures numériques

La signature RSA
(Rivest , Shamir , Adleman)

Signatures numériques RSA : Introduction

- **Utilisation du chiffre RSA:** en signature.
- **Utilisation en mode appendice:** ne pas chiffrer en RSA tout le message M pour le signer.
  - Trop lent
  - N'ajouterait pas de redondances.
- **Utilisation d'un hachage M → H(M).**
  - **Le plus souvent MD5 ou SHA-1.**
  - **Préconisation de sécurité :** actuellement SHA-256 (le moins lent et le moins encombrant des hachages sécuritaires).
- **Message signé de Alice à Bernard**
  - **Alice clé publique E_A (e_A, n_A) ; clé privée D_A (d_A, n_A)**
  - **sig (M) $S := D_A(H(M)) := (H(M))^{dA} \pmod{n_A}$**
  - **ver (S) $H(M) = E_A(S) = (S)^{eA} \pmod{n_A}$**
- **Signature:** de très loin la plus utilisée.

Signatures numériques : Le protocole de signature RSA

Alice

Bernard

$M \rightarrow MDC := H(M)$

$MDC := MDC||Bourrage$

$Sig := D_A (MDC)$

Alice, Bernard, M, Sig →

← Bernard, Annuaire , (Alice , E_A?)

Annuaire

Annuaire, Bernard, Alice , E_A) →

Vérifier E_A

$M \rightarrow MDC := H(M)$

$MDC := MDC || Bourrage$

$MDC := E_A (Sig)$

Signature numérique RSA : Problème de bourrage

- **H(M) plus petit que la taille des blocs chiffrés en RSA :** fonctions de hachage de 128 à 512 bits, RSA blocs de 320 à 2048 bits selon valeur de n=p.q
- **Solution 1** : Complémentation par concaténation d'une **information fixe** => **H(M) || B.**
- **Solution 2** : Complémentation par concaténation d'un **nombre aléatoire** => **H(M) || GNPA(x).**

  Plus sécuritaire car la partie aléatoire est inconnue
- **Solution 3** : Complémentation **par extension de H(M) par une fonction d'extension pseudo aléatoire H(M) → F(H(M))** F résultat entre 0 et n=pq aléatoire (proposition B Schneier).

Protocoles de signatures numériques

La signature DSA
‘Digital Signature Algorithm’

Signatures numériques DSA : Introduction

- **Origine: NIST** (National Institute Of Standards and Technology) **et NSA** (National Security Agency) **en 1991.**
- **Normalisé dans le cadre du standard DSS :** Digital Signature Standard (1994).
  - **DSS : une signature en mode appendice de 320 bits (2x160)**
  - **Utilisation de SHA-1 comme fonction de hachage (160 bits).**
  - **Utilisation de DSA comme signature.**
- **Solution très voisine de la signature numérique définie par El Gamal** : modifications légères.
  - Utilise le chiffre à clé publique défini par El Gamal.

Signatures numériques : Signature El Gamal

- **Chiffre à clés publiques El Gamal** : basé sur la difficulté du logarithme discret.
- **Choix d'un jeu de clés :**
  - **Clé publique : p , g , y**
    - **p grand nombre premier**
    - **g inférieur à p**
    - **$y = g^x \pmod p$**
  - **Clé secrète : x , p**
- **Solution originale** de signature:
  - **On ne chiffre pas** directement le hachage du message.
  - **On fait un calcul** spécifique de signature.

Signature El Gamal : Génération de la signature

- **H(M) :** le hachage d'un message (le résumé pour la signature).
- **Choisir un nombre aléatoire k** : k et p-1 sont premiers entr'eux (k est inférieur ou égal à p-2 et ne divise pas p-1).
- **Calculer :** $a = g^k \pmod p$
- **Trouver b tel que** : identité de Bezout

$$x\,a + k\,b = H(M) \pmod{p-1}$$

- **Signature de M** : le couple a , b.
- **Ne peut se calculer** : que si l'on connaît x la clé privée et k valeur aléatoire qui doit rester secrète.
- **Si l'on signe deux fois la même chose** : la signature change à cause de k (qui change).

Signature El Gamal : Vérification de la signature

- **Connaissant : H(M) , (a,b)** (la signature) **et p,g,y (**la clé publique)**.**
- **Calculer :** $g^{H(M)} \pmod p$ et $y^a a^b \pmod p$.
- **Si** les deux valeurs sont égales M n'a pas été modifié.

Preuve: D'après la construction de la signature (a et b):

$$H(M) = x\,a + k\,b \pmod{p-1}$$

Soit encore : $H(M) + (p-1)\,r = x\,a + k\,b$

On a donc : $g^{H(M)}\, g^{(p-1)r} \pmod p = g^{xa}\, g^{kb} \pmod p$

Avec : $g^{(p-1)} \pmod p = 1$ (Fermat)

Et par choix: $y = g^x \pmod p$, $a = g^k \pmod p$

Donc : $g^{H(M)} \pmod p = y^a\, a^b \pmod p$

Signatures numériques : DSA Conclusion

- **Modifications DSA par rapport à El Gamal**
  - **Réduction de la taille des signatures** : en travaillant sur un sous groupe 0..q sur 160 bits (nombres inférieurs à p grand nombre premier 1024 bits nécessaire pour la sécurité du chiffre El Gamal).
  - **Petite modification** des formules de calcul.
- **Solution qui a été très utilisée :** surtout aux USA.
- **Inconvénient** : Solution très lente.
- **Avantages :**
  - **Sécurité :**
    - Solution à clé publique reposant sur la difficulté du logarithme discret,
    - Un nombre aléatoire k change les signatures pour deux messages identiques.
  - **Génère une signature de longueur raisonnable** : 320 bits (en relation avec SHA-1 160 bits => deux fois cette taille)
  - **Solution retenue aussi car n'utilisant pas le RSA et solution uniquement dédiée à la signature** : la solution ne devait pas pouvoir être utilisée en chiffrement ...

Protocoles d'intégrité

Les codes d'authentification de message (CAM)
MAC 'Message Authentication Code'

Introduction
Mac de base
Mac double
CBC-MAC
HMAC

Codes d'authentification de messages (MAC) : Introduction (1)

- **1) Objectif :** générer à l'émission une structure de donnée MAC permettant de vérifier l'intégrité d'un message avec une **solution utilisant un secret partagé** (une clé secrète K).
- **2) En émission** : calculer MAC(K,M) ; envoyer M , MAC(K,M)
  - **Mac (K,M) incorpore K et M** dans une structure de donnée de taille fixe généralement en utilisant H une fonction de hachage (MD5, SHA)
  - **Exemple type** : MAC(K,M) = H(K||M)) ou || est la concaténation.
  - **D'où une autre terminologie** : en anglais KHF 'Keyed Hash Function', en français fonction de hachage avec clé.
- **3) En réception:** recevoir M,a ; vérifier a =MAC(K,M)

Codes d'authentification de messages (MAC) : Introduction (2)

- **4) Etude de la sécurité**
  - **MAC à sens unique** : pour ne pas pouvoir remonter à K.
  - **MAC sans collisions** : pour ne pas pouvoir violer l'intégrité.
- **5) Etude des performances**
  - **Bons temps de calcul** : gagner du temps sur la signature numérique.
  - **Volume peu important** : ne pas encombrer les messages (parfois on ne prend qu'un sous ensemble des bits calculés par une fonction MAC).

Codes d'authentification de messages (MAC) : Propriétés

Propriétés de la signature qui sont conservées

- **P1) Un MAC concerne un document :** (selon collisions).
- **P2) Intégrité :** un document avec MAC peut être très difficilement modifié (selon collisions).
- **P6) Performances :** un MAC est facile à calculer et à vérifier.

Propriétés de la signature qui sont abandonnées

- **P3) Intégrité :** un MAC n'est pas vérifiable par n'importe qui
  - **Seuls les détenteurs de la clé K peuvent vérifier le MAC.**
- **P4) , P5) Authentification , non répudiation**
  - Si Alice et Bernard qui communiquent en partageant K ne sont pas d'accord sur le message M on ne peut les départager

Codes d'authentification de messages : MAC de base (simple)

- **Construire un MAC en utilisant directement une fonction de hachage H:** nombreuses variantes de solutions.
  - Selon la fonction H utilisée : MD5, SHA-1, SHA-256 (meilleure est la fonction H meilleur est le MAC).
  - Selon l'utilisation de H dans le cadre du MAC (simple, doubles ...)
- **Solution 1**: H appliqué à la concaténation || de K et M
  - MAC (K, M) = H(K||M)
  - **Attaque d'extension de longueur** : si H est itératif (toujours) on peut rajouter des informations en fin de M et calculer un MAC correct.
  - **Exemples dans cette classe** : SHA-1 (K||M) , SHA-256(K||M);
- **Solution 2**: H en mode simple sur la concaténation de M et K
  - MAC (K, M) = H(M||K)
  - La clé dans le dernier bloc est plus facilement attaquable (nombre d'essais)

Codes d'authentification de messages : MAC double

- **Solution** : Utiliser un hachage H en mode double (d) sur la concaténation de K et de M.
- **MAC (K, M)** = H_d (K || M) = H(H (K || M))
- **Pas d'extension** de longueur possible.
- **Solution plus lente** qu'un Mac simple mais solution considérée comme sécuritaire
- **Exemple dans cette classe :**

  SHA_d-256 (K||M) = SHA-256 (SHA-256 (K || M))

Codes d'authentification de messages : CBC-MAC

- **1) CBC-MAC** '**C**ipher **B**lock **C**haining – **MAC'**:
  - Construire un MAC à partir d'un chiffrement par blocs.
  - Chiffre en mode chaînage : voisin d'une fonction de hachage itérée.
  - Avantage : on réutilise le programme d'un chiffre existant.
- **2) Fonctionnement :**
  - Choix d'un chiffre par blocs E(K, B) et d'une clé secrète K
  - Découpage par blocs M = B_1, B_2 , …. ,B_i , ….. B_{Nblocs}
  - Choix d'un vecteur d'initialisation H_0 = VI
  - Itérations : $H_i = E(K, B_i \oplus H_{i-1})$ i=1, …., Nblocs
  - CBC-MAC(K, M) = H_{Nblocs} Le dernier résultat des itérations.
- **CBC-MAC :** solution de mise en œuvre délicate
  - Selon les propriétés de collisions du résultat final.
  - **Variantes utilisées** : CBC-MAC-DES, CBC-MAC-AES.

Codes d'authentification de messages : HMAC Définition

- **HMAC :** 'Keyed-**H**ashing **M**essage **A**uthentication **C**ode'
- **IETF :** RFC 2104 (1997)
- **HMAC (K, M)** = H [(K ⊕ Ipad) || H((K ⊕ Opad) || M))]
  - K est une clé secrète de longueur n octets.
  - B est la longueur du hachage généré par H (16, 20, 32 oct)
  - Ipad (0x36 répété B fois) et Opad (0x5C répété B fois) sont des constantes qui servent à compléter la clé K si elle n'a pas la taille du bloc utilisé par le hachage (bourrage).
  - || définit la concaténation.
- **Exemples de HMAC** : HMAC-MD5 ; HMAC-SHA-1 ; HMAC-SHA-256

Codes d'authentification de messages : HMAC Conclusion

- **Avantages des HMAC :**
  - **HMAC est rapide**
    - K XOR opad et K XOR ipad peuvent être calculés au préalable et utilisés comme initialisation pour des algorithmes de hachage itératifs MD5.
    - HMAC est a peu près équivalent à un calcul simple de hachage.
  - **HMAC résiste aux extensions de longueur** de M.
  - **K clé secrète n'est pas présente à la fin**.
  - **Les techniques d'attaques actuelles de collisions sur les hachages** ne fonctionnent pas avec HMAC.
    - Avec HMAC il n'est pas nécessaire d'utiliser un hachage H_d
- **HMAC : solution recommandée**
  - **Déjà ancienne et déjà attaquée avec une bonne résistance.**
- **Solution très utilisée :** déployée dans de nombreux protocoles de sécurité IPSEC, SSL, DNSSEC.

Codes d'authentification de messages (MAC) : Conclusion

- **1) MAC : une utilisation très répandue** dans de nombreux protocoles de sécurité en mode message.
- **2) Très nombreuses autres solutions non citées ici :** peu utilisées industriellement.
- **3) En général :** Bonnes performances en temps de calcul.
- **4) En général :** Niveau de sécurité satisfaisant.
- **5) Solutions recommandées actuellement:**
  - MAC doubles
  - HMAC (peut-être la meilleure solution actuelle disponible).
- **6) Pour un bon MAC :** utiliser une bonne fonction de hachage.

  Exemple de MAC recommandé => HMAC-SHA-256

Protocoles d'intégrité

L'intégrité d'un flot de messages : le rejeu

Introduction

Solutions

Intégrité d'un flot de messages : Introduction

- **1) Problème général :** respecter les relations de causalité.
- **2) Problème du rejeu :** un intrus écoute un message correct et le relance ultérieurement pour en tirer un avantage ou désorganiser le protocole.
  - **Message d**'une connexion ancienne ou de la même connexion
  - => Destruction silencieuse du message rejoué.
- **3) Retransmission** : un émetteur retransmet ultérieurement un message considéré par lui comme non délivré (rejeu légal).
  - => Destruction silencieuse normale.
- **4) Principe de Horton:**
  - **Contrôler en intégrité la charge utile d'un message M** : les bits de données.
  - **Mais aussi les autres informations utiles à l'interprétation correcte de cette charge utile** : ici typiquement les données qui identifient un message dans un flot (données protocolaires).
    - Identifiant émetteur, destinataire, référence de connexion, numéro de séquence dans une connexion ....

Intégrité d'un flot : Solutions au problème du rejeu (1)

- **Base des solutions :** utiliser un identifiant unique pour chaque message.
- **Nonce :** un identifiant utilisé seulement une fois et contrôlé ('**n**umber used **once**').
- **Solution 1) Nonce basé sur un numéro de séquence**
  - Identifier les messages **par des numéros de séquence**.
  - Identifier de même les **connexions successives**.
  - Utiliser de **grands espaces** de numéros (64 bits au moins)
  - **Gestion des numéros très stricte** : démarrage.
  - **Vérification en intégrité des numéros:** signature, MAC.
  - **Rejeu excessivement difficile** : possibilité de succès uniquement lié aux collisions.

Intégrité d'un flot : Solutions au problème du rejeu (2)

- **Solution 2) Nonce basé sur le temps physique**
  - Utiliser un estampillage par **l'horloge de l'émetteur**.
  - Cette datation nécessite un protocole de **synchronisation d'horloge** (émetteur/récepteur) lui-même de sécurité.
  - **Vérification en intégrité** de la date: permet au récepteur de vérifier la cohérence de la date et évite ainsi le rejeu.
- **Solution 3) Nonce basé sur un nombre aléatoire**
  - **Nombre aléatoire imprévisible** pour chaque message : très faible probabilité de tirer deux fois le même nombre.
  - **Doit être échangé** dans les deux sens et vérifié (intégrité)
  - **Adapté à un échange client serveur** : sporadique.
  - **Peu adapté à un flot :** coût de vérification du nombre.

Protocoles de sécurité

Protection
(Contrôle d'accès)

Introduction

Gestion des droits (politique de sécurité).

Politique discrétionnaire dans les systèmes : anneaux et domaines

Protection : Introduction
Le problème du contrôle d'accès (1)

- **1) Définition:** le domaine des systèmes d'exploitation et des réseaux qui traite du contrôle de l'accès à des composants matériels ou logiciels.
- **2) Terminologie :** Protection, Autorisation, Contrôle d'accès, '**Authorization**', 'Access Control' (AAA).
- **3) Notion d'interface d'accès** => protection des interfaces.
- **4) Objectifs :** limiter la propagation des erreurs non intentionnelles, contrôler l'accès pour empêcher les intrusions.
- **5) Domaine développé avec les premiers systèmes d'exploitation:** début des années 1960.

Protection : Introduction
Le problème du contrôle d'accès(2)

- **6) Moyen :** un ensemble de mécanismes permettant de vérifier qu'un usager disposant d'un ensemble de droits agit en respectant ces droits.
- **7) Politique de sécurité :** définition des droits d'un usager et de leurs évolutions possibles.
- **8) Au départ :** vérification des droits de lecture ou d'écriture sur une donnée (mémoire ou fichier).
- **9) Extension à tous les types d'actions :** droits concernant n'importe quelle action exécutable (instruction, procédure).

Protection/Contrôle d'accès : Sujets , objets et groupes

- **1) Sujet :** Entité informatique qui réalise des actions devant être contrôlées.
  - **Exemples :** utilisateurs, processus, objets, composants, ...
- **2) Rôle : Groupe de sujets** recevant un traitement identique du point de vue d'une politique de sécurité.
  - **Exemples :** acheteur, vendeur, banque, **....**
- **3) Objet :** Entités informatique qui reçoit et exécute des requêtes devant être contrôlées.
  - **Exemples :** pages ou segments de mémoire, fichiers, processus, objets, composants.
- **4) Groupes d'objets :** Traités de manière identique du point de vue d'une politique de sécurité
  - **Exemple :** ensemble de fichiers d'une application.

Protection/Contrôle d'accès : Notion de capacité ('capability')

- **1) Abstraitement** : un droit d'exécution d'un sujet sur un objet.
- **2) Plus précisément** : l'affirmation d'un prédicat d'autorisation d'exécution d'une action dans le futur.
- **3) Concrètement** : une structure de données qui matérialise ce droit.
- **4) Informations dans une capacité**
  - **Au minimum un couple** : désignation de l'objet , désignation du droit (si on range la capacité dans le descriptif d'un sujet).
  - **Nombreuses autres informations possibles** : date d'invalidation, signature ...

Protection/Contrôle d'accès : Exemples de capacité ('capability')

- **1) Droits dans les systèmes de stockage (mémoire, fichier, base de données):** droit de lecture, d'écriture, création, destruction.
- **2) Droits réseaux :** droit d'émettre ou de recevoir des messages sur une adresse ....
- **3) Droits sur le noyau du système:** droit d'utiliser un périphérique (imprimer), de créer ou détruire un processus ...
- **4) Droits dans la politique de sécurité** : droit de transférer un droit à un autre sujet, droit de se donner un droit quel qu'il soit.

Protection/Contrôle d'accès : Notion d'assertions

- **1) Assertion :** généralisation de la notion de capacité à l'affirmation de n'importe quelle formule de logique (généralement de logique temporelle ou de connaissance)
- **2) Notion d'affirmateur d'une assertion :** le sujet qui est la source d'une assertion.
- **3) Informations dans une assertion**
  - **Nom** de l'assertion,
  - **Spécification** de l'assertion : éventuellement,
  - **Parties concernées** par l'assertion : affirmateur de l'assertion, sujets/objets concernés, ...
  - **Attributs divers :** date d'invalidation, signature ...
- **4) Exemple:** SAML Security Assertion Markup Language.

Protection/Contrôle d'accès : Exemple d'assertions

- **1) Signature :** Affirmation, l'opération écrire de cette donnée à été réalisée par moi (connaissance d'une opération écrire dans le passé) .
- **2) Assertion d'authentification** : Affirmation, un serveur d'authentification a authentifié tel sujet (connaissance d'une authentification passé).
- **3) Capacité (droit pour une action) :** Affirmation : l'action A est autorisée pour le sujet S (formule de logique temporelle du futur).
- **4) Assertion de capacité pour une suite d'actions :** les actions A et B sont liées et autorisées uniquement dans l'ordre A avant B (logique temporelle du futur).

Protection/Contrôle d'accès : Matrice de contrôle d'accès

- **Politique de sécurité** (au sens traditionnel) : l'ensemble des règles qui gouvernent la création, l'évolution et la destruction des droits (des capacités) et leur utilisation.
- **Matrice de contrôle (Lampson 1971)** : matrice qui définit à chaque instant les droits de chaque sujet sur chaque objet (généralement sujets en ligne et objets en colonnes)

	Seg M1	Seg M2	Fich F1	Fich F2
P1	R,W,E	R,W		R
P2	R,W,E	R	R,W	R
S1	R,W,E		R,W,E Créer Détruire	Propriétaire
S2		R,W,E		

Protection/Contrôle d'accès : Évolution de la matrice des droits

- **La matrice des droits évolue en fonction des événements suivants :**
  - Création et destruction des sujets et des objets.
  - Création et destruction des droits pour soi même.
  - Propagation et révocation des droits des autres usagers.
- **Exemple :**
  - Dans le système VMS les droits d'un processus fils sont hérités du père et partagés entre les fils.
  - Pour le processus racine ce sont les droits de l'utilisateur créateur.

Protection/Contrôle d'accès : Spécification d'une politique

- **Une politique de sécurité définit essentiellement les règles d'évolution** de la matrice des droits.

- **Définition d'une politique de sécurité** : analogue d'une spécification comportementale d'un composant
  - Des règles qui gèrent les évolutions de la matrice des droits.
  - Des propriétés de sécurité qui doivent être vérifiées à chaque instant (invariants de sécurité) par la matrice de contrôle des droits.

- **Possibilité de mise en oeuvre de toutes les techniques classiques de validation logicielle pour valider une politique de sécurité.**
  - **Preuve formelle :** analogue à la preuve de programme.
  - **Vérification opérationnelle (test)**
    - Les évolutions prévues préservent les invariants (l'implantation de la politique n'autorise que les transitions valides).
    - Le contrôle d'accès est toujours exécuté.

Protection/Contrôle d'accès : Deux principes fondamentaux

Le confinement

- Les objets sont maintenus dans des **domaines de protection** étanches pour empêcher qu'une entité n'interfère avec une autre à la suite d'une action involontaire ou volontaire.
- **Exemple** : accès à l'espace mémoire d'un autre usager.

Le moindre privilège

- Pour qu'un système fonctionne en sécurité il faut donner à ses utilisateurs **exactement les droits dont il ont besoin** pour s'exécuter : **ni plus ni moins**.
- **Si l'on donne plus de droits** on risque de voir ces droits utilisés anormalement : involontairement ou volontairement.
- **Exemple** : ne pas travailler en standard sur un système en mode administrateur.

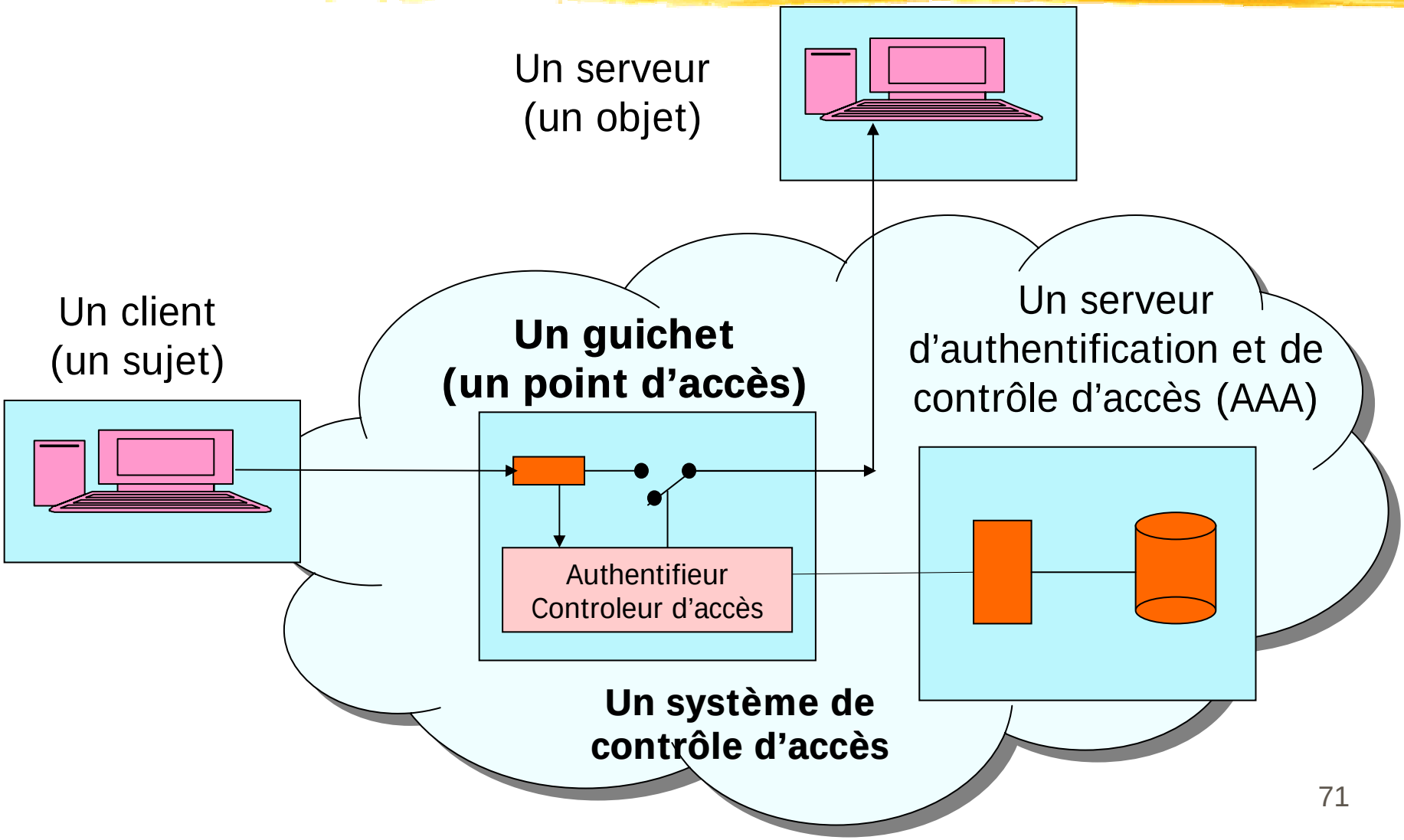
Protection/Contrôle d'accès : Politiques discrétionnaires et obligatoires

- **Politiques discrétionnaires**
  - Une politique est **discrétionnaire** si la propagation des droits sur un objet est entièrement contrôlée par les sujets.
  - **Exemple :** Pour chaque objet un droit propriétaire ("owner") permet à un sujet de propager des droits sur cet objet => Droits d'accès aux fichiers UNIX.
- **Politiques obligatoires**
  - La politique est **obligatoire** si le processus de propagation est défini par un ensemble de règles qu'un sujet ne peut modifier.
  - **Exemple :** Politique militaire de sécurité.

Protection/Contrôle d'accès : Politiques nominatives et de rôles

- **Politique nominative**
  - Les droits sont attribués **individuellement** à chaque sujet.
  - **Exemple :** Droits d'accès aux utilisateurs pour les fichiers UNIX.
- **Politique de rôle**
  - Les droits sont attribués à un sujet automatiquement **par son appartenance à un groupe**.
  - **Exemple** : Droits d'accès des groupes d'utilisateur dans le système de fichier UNIX.

Protection/Contrôle d'accès : Notion de guichet



Protection/Contrôle d'accès : Détails concernant les guichets

- **Guichet :** un nom générique pour un point de passage obligé qui permet la vérification des droits.
  - Tout accès à un objet se fait via un guichet
  - Un guichet rassemble pour une interface les aspects de sécurité relatifs à la protection (au contrôle d'accès).
- **Une entité se présente au guichet :**
  - Authentification de l'entité.
  - Authentification du guichet (risque de déguisement)
  - Présentation d'une capacité pour réaliser une opération.
- **Exemples de guichet**
  - **Point d'entrée dans l'exécutif** de requêtes d'un système d'exploitation.
  - **Serveur d'authentification et de contrôle d'accès** dans un système d'authentification unique (SSO 'Single Sign On')
  - **Mur pare-feux** (anti feux 'firewall')
  - **Aspect de sécurité** (programmation par aspects).

Protection/Contrôle d'accès : Construction des guichets

Principe de méfiance mutuelle: le client se méfie du serveur et le serveur se méfie du client.

- **A) Garantir les transferts de données entre le client et le guichet :** protégés selon les besoins en intégrité ou en confidentialité.
- **B) Protéger les données confidentielles :** toutes données qui doivent être étanches en lecture (par exemple données à la base de l'authentification).
- **C) Protéger en écriture :** certaines données comme des capacités ou aussi des codes exécutables partagés.
- **D) Authentifier réciproquement client et guichet.**
- **E) Garantir que l'exécution de l'opération** ne peut être faite que par le guichetier et selon sa spécification => notion de guichet **incontournable.**
- **F) Pouvoir enregistrer de façon non falsifiable toutes les opérations :** journalisation => non répudiation.
- **G) Pouvoir noter toutes les tentatives de fraude:** auditabilité.

Protection/Contrôle d'accès : Administration des guichets

- **1) Gestion des entités sujets, objets et des données d'authentification de ces entités :** désignation, création, destruction.
- **2) Gestion des guichets et des données d'authentification de ces guichets :** désignation, création, destruction.
- **3) Gestion des capacités (droits) selon la politique de sécurité :** création, destruction, propagation.

Protection/Contrôle d'accès : Notion de moniteur de référence

- **1) Les composants système dédiés à la sécurité reposent sur un mécanisme de sécurité central:** sur lequel repose toute la sécurité (de préférence basé sur du matériel).
- **2) Il doit être: Inviolable,**
  Incontournable,
  Correct

  Par rapport à un ensemble de propriétés permettant d'implanter une politique de sécurité.
- **3) Le moniteur de référence est le "méta guichet" :** permet de construire des guichets et de gérer les droits d'accès.

Protection/Contrôle d'accès : Construction du moniteur de référence

- **1) Un moniteur de référence est construit selon une hiérarchie de mécanismes.**
- **2) Dans un système centralisé : on utilise des mécanismes matériels au niveau le plus bas = >** pour contourner ces mécanismes il faudrait pouvoir modifier le matériel.
  - Gestion de la mémoire.
  - Mode d'exécution des processus et contrôle d'accès aux instructions privilégiées.
  - Programmes en mémoire morte
  - Système matériel d'authentification (exemple: lecteur de carte à puces).
- **3) Dans un système réparti** le moniteur de sécurité est :
  - Composé des moniteurs de sécurité locaux
  - Des protocoles de sécurité construits à partir des fonctions cryptographiques.

Protection

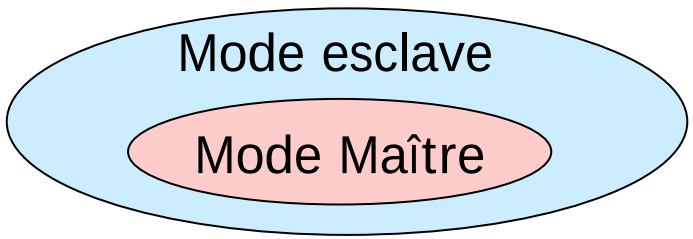
Politiques discrétionnaires dans les systèmes d'exploitation

Machines à anneaux

Machines à domaines (machines à capacités)

Protection
Machines à anneaux : Généralités

- **1) Mécanisme de protection matériel/logiciel** : appliqué par un processeur et son système d'exploitation.
- **2) Notion d'anneau** :
  - Un niveau de protection matérialisé par un entier : Les niveaux sont hiérarchisés: notion d'anneaux concentriques.
  - De 2 niveaux (mode maître esclave) à assez souvent 4 (Pentium) ou 8 niveaux (rarement plus)
  - Exemple: Mode maître/esclave



- **3) Sujets :** un programme (un processus) s'exécute à un instant donné dans un anneau donné (à un niveau de protection donné).
- **4) Objets :** un anneau (un niveau de protection) est associé à chaque ressource (instruction processeur , segment mémoire ...)

Protection Machines à anneaux : Règles de fonctionnement

- **1) Objectif visé** : certaines opérations (accès ressources) ne sont **exécutables** que dans un **anneau** qui le permet.
- **2) Moyen :** contrôle permanent des droits.
  - **Toute opération** (instruction, référence mémoire) est **soumise à un contrôle** du droit d'accès matériel.
  - **Le registre d'anneau** du sujet demandeur (processus) est comparé au registre d'anneau de la ressource (processeur, segment mémoire).
  - Pour qu'une opération soit exécutée **le niveau de l'appelant** doit être **supérieur ou égal** à celui de l'objet référencé sinon il y a déroutement.
- **3) Cas d'un déroutement ('trap')**
  - Exécution d'une instruction ou d'un accès **avec augmentation des droits** (délibéré ou accidentel).
  - **Déroutement = Demande de changement d'anneau**
  - Le déroutement provoque le passage à un guichet => **l'exécution d'un code de contrôle du droit de passage dans l'anneau appelé.**

Protection Machines à anneaux : Exemple

- **Exemple : Système organisé en quatre anneaux :**
  - 0 : superviseur d'entrées sorties (pilotes de périphs).
  - 1 : exécutif de requêtes.
  - 2 : système de fichiers.
  - 3 : processus usager.
- **Opération étudiée : Lecture d'article sur un fichier**
- **1)** Usager : anneau 3 -> Gestion des fichiers : anneau 2
  - Contrôle du droit d'accès sur le fichier,
  - Fonctionnement système de fichier: calcul d'adresse bloc.
- **2)** Gestion des fichiers : anneau 2 -> Appel à l'exécutif anneau 1
  - Contrôle du droit d'accès au volume logique ('file system'),
  - Requête au pilote pour accéder au volume physique.
- **3)** Exécutif : anneau 1 -> Superviseur d'entrées sorties anneau 0
  - Contrôle du droit d'accès à l'unité de disque.

Protection Machines à domaines : Introduction

- **1) Notion de domaine de protection** : une généralisation des anneaux à des ensembles de droits organisés de façon quelconque (sans hiérarchie).
  - Défini pour un sujet par un ensemble de droits sur des objets.
  - A tout instant un objet accessible doit appartenir au domaine de protection courant (au domaine du sujet en exécution).
- **2) Tout appel d'une opération est réalisé en fournissant une capacité**:
  - Référence de l'appelé (désignation)
  - Droit possédé par l'appelant sur l'appelé (protection).
- **3) Rôle du système d'exploitation (moniteur de référence de sécurité matériel / logiciel)**
  - La capacité est contrôlée avant d'exécuter chaque opération (instruction processeur, accès mémoire, méthode d'un objet ....).
  - La modification des capacités (la manipulation, la propagation des droits) ne peut être réalisée que par appel du moniteur (fonctionnant en mode protégé).

Protection Machines à domaines : Approche des listes de capacités

- **1) Machines à capacités ('Capability based') :** une approche de **capacité** stocke **la matrice** (sujet, objet) **par lignes.**
  - **Une ligne de la matrice : une liste de capacités ou c-liste.**
- **2) Stockage des capacités associées aux sujets :** Pour chaque sujet on gère au niveau système (dans une mémoire protégée) **l'ensemble des capacités de ce sujet**.
- **3) Machines à capacités au niveau du matériel :** le matériel implante toute la protection par capacité (segments spéciaux, vérification matérielle des droits par consultation des segments de capacité).
- **4) Machines à capacités en logiciel :** les listes de capacités sont traitées par programmes (le moniteur de référence matériel utilise des anneaux).

Protection Machines à domaines : Exemples d'approche à capacités (1)

Exemple 1 Capacités pour les primitives d'un système d'exploitation.

- **Système VAX-VMS**
- En fait tous les systèmes d'exploitation **bien protégés.**

- **1) Les primitives du noyau** sont protégées par des droits d'accès : droits processus, droits d'opérations réseaux, droit de modifier ses droits.
- **2) Une liste de capacités est associée au descriptif de chaque usager.**
- **3) A chaque fois qu'un usager** (sujet) accède à une primitive système (objet) on vérifie le droit du sujet.

Protection Machines à domaines : Exemples d'approche à capacités (2)

Exemple 2 Machines à Capacités (niveau matériel)

- **IBM AS 400**
- En fait **très peu d'autres exemples** car coûteux.

- **Capacités contrôlées en permanence** pour les droits d'exécution des accès mémoire et des opérations processeur.
- **Mécanismes matériels de gestion des segments de capacités :** les segments de capacités sont exploités et protégés au niveau matériel.

Protection Machines à domaines : Exemples d'approche à capacités (3)

Exemple 3 Murs Pare-Feux ('Firewalls')

- **Implantation d'une politique de capacités pour les réseaux** :
  - **Si l'on définit une liste de droits par émetteurs** : machine d'adresse IP donnée, numéro de port TCP donné.
  - **Sujets :** les sites émetteurs.
  - **Objets :** les sites destinataires.

Protection Machines à domaines : Approche des listes de contrôle d'accès

- **1)** Une approche de **liste de contrôle d'accès** stocke la **matrice des droits par colonnes**.
  - **Pour chaque objet : la liste des sujets** et **leurs droits** sur l'objet.
  - **Colonne : notion de liste de contrôle d'accès** ('ACL Access Control List based') **.**
- **2) Exemple :** Système de fichier UNIX
  - On associe à chaque fichier **une liste de droits** (lire, écrire, exécuter)
    a) Pour un usager particulier: **le propriétaire** (politique nominative)
    b) Les **membres du groupe** du propriétaire (politique de rôle)
    c) **Tous les autres usagers** du système (politique de rôle)
  - Dans de nombreux systèmes de fichiers les droits sont plus fins
    - **On peut définir des listes de contrôle d'accès fichiers** pour n'importe quel utilisateur ou n'importe quel groupe (exemple VAX-VMS).

Protection Machines à domaines : Comparaison des approches

- **1) Choix entre les deux approches** : Repose sur le coté plus ou moins pratique de la manipulation des listes de droits dans les sujets ou les objets:
- **2) Les listes de contrôle d'accès sont rangées dans les descriptifs des objets**
  - Pratique pour la révocation des droits puisqu'ils sont associés à l'objet.
  - Pratique pour des objets à longue durée de vie, de types peu variables.
  - Pratique s'il existe déjà des descriptifs d'objets significatifs (fichiers)
- **3) Les listes de capacités (c-listes) sont rangées dans les descriptifs des sujets.**
  - Pratique pour ce qui concerne les droits des objets manipulés en mémoire centrale par le noyau (peu de sujets actifs, descriptifs noyau).
- **4) Pour la distribution: une solution mixte**
  - Capacités stockées par émetteur (sujet).
  - Présentées à l'objet distant pour vérification par un guichet destinataire.

Protocoles de sécurité

Authentification

Introduction.

Authentification à mot de passe.

Protocoles d'authentification.

Authentification : Introduction

- **1) Authentification** : vérification d'identité de l'auteur d'une action (sujet).
  - Présentation d'un **identifiant :** existence d'un nommage
  - Présentation d'une **créance ('credential')** : 'lettre de créance' instrument permettant d'authentifier.
  - **Vérification de la validité** de la créance.
- **2) Impossibilité** : d'assurer une propriété de sécurité d'une action sans s'appuyer sur la garantie indiscutable des identités (client, serveur, affirmateur).
  - **Propriété d'intégrité** : authentification de l'écrivain.
  - **Propriété de non répudiation** : authentification du client et du serveur pour régler les différents juridiques.
  - **Propriété de protection** : vérification de l'identité du demandeur et de celle de l'affirmateur du droit.

Authentification : Pour une session ou en continu

- **Authentification en début de session** : une fois pour toute la session, toute une suite d'actions jusqu'à fermeture.
- **Authentification en continu**: pour chaque action individuelle

L'authentification devrait être assurée en continu

- **Quand l'entité est un usager :** il peut quitter son poste en le laissant ouvert
  - => Procédure de déconnexion automatique, procédure de ré authentification périodique.
- **Quand l'entité est informatique :** une substitution peut avoir lieu surtout en réseau
  - => Application des protocoles d'authentification en continu.
  - => problèmes de lourdeur et de performances.

Authentification : Différentes créances 1 - Connaissance

- **1) Information tenue secrète**: clé, mot de passe (« ce que l'on sait »)
  - Technique **la plus simple** et la **plus répandue.**
  - Applicable aux **entités informatiques.**
- **2) Problèmes bien connus** :
  - Le secret peut-être découvert.
  - Le secret peut-être confié à un tiers.
- **3) Quelques parades :**
  - **a) Obliger l'usager à changer** régulièrement de secret : mot de passe.
  - **b) Surveiller les tentatives d'accès** illicite : les enregistrer et les éditer.
  - **c) Prévenir l'usager des connexions** sur son compte en affichant la date et l'heure (ex du dernier accès).

Authentification : Différentes créances
2 – Moyen matériel

- **1) Secret matérialisé physiquement** (« ce que l'on possède »)
  - Une clé (traditionnelle) => une carte lue : magnétique,
  - Miniaturisation et radio : RFID Radio Frequency IDentifiers,
  - Complexification : carte à puce.
  - Technique **simple et répandue**.
- **2) Les problèmes :**
  - **La perte.**
  - **Le vol** du support.
  - **La duplication** plus ou moins facile mais toujours possible.
- **3) Les solutions :**
  - **La révocation.**

Authentification : Différentes créances
3 – Techniques biométriques

- **1) Classe de solutions** adaptées à l'identification des personnes (« ce que l'on est »).
- **2) Créance basée sur un caractère biologique ou morphologique ou comportemental.**
  - A priori caractérisant de manière unique l'utilisateur.
  - Liste de caractères dans les transparents suivants.
- **3) La vérification repose sur la classification** du caractère retenu dans un ensemble de personnes (analyse statistique, réseaux de neurones).
  - Nécessité d'études approfondies du caractère utilisé à **l'intérieur du groupe** autorisé et **dans une population quelconque.**
- **4) Incertitudes des techniques biométriques**
  - La variabilité intra-individuelle : le prélèvement pour vérification peut ne pas suivre exactement un profil préenregistré (en raison de son type).
  - Variabilité inter-individuelle : incertitudes.
  - Deux types d'erreurs possibles: le rejet à tort d'un individu autorisé ou l'acceptation à tort d'une personne non autorisée.

Authentification : Différentes créances
Techniques biométriques : Solutions (1)

- **1) L'empreinte digitale (morphologique)**
  - Le sujet applique son doigt sur un prisme.
  - La pression déclenche une analyse par balayage d'un faisceau infrarouge.
  - Le signal reçu dépendant des sillons de l'empreinte (creux et bosses successives) est analysé et classifié.
- **2) La reconnaissance faciale (morphologique)**
  - Identification par comparaison du visage à ceux mémorisés dans une base.
- **3) La vascularisation de la rétine (morphologique)**
  - Empreinte biométrique analogue à l'empreinte digitale qui est très stable.
  - L'image de fond de l'oeil est obtenue par un dispositif monoculaire utilisé dans les test de vision médicaux.
  - La numérisation est effectuée par une caméra infrarouge.
  - L'image est classifiée.
  - Voir aussi des études avec l’iris de l’oeil.

Authentification : Différentes créances Techniques biométriques : Solutions (2)

- **4) La voix (morphologique et comportemental)**
  - Le sujet prononce quelques mots.
  - Le système numérise et classifie le signal.
- **5) La géométrie de la main (morphologique)**
  - Une caméra enregistre l'image de la main.
  - La caractérisation est obtenue par mesure de la longueur et de la largeur de chaque doigts.
- **6) Dynamique de la signature (comportemental)**
  - Elle est obtenue par une tablette graphique reliée à un calculateur
  - La signature et surtout le mouvement réalisé par la main pour la fabriquer sont analysés en comparaison à plusieurs signatures de référence.
- **7) Dynamique de la frappe clavier (comportemental)**
  - Un clavier spécial permettant la mesure précise des intervalles dans les séquence de frappe ou la pression des doigts sur le clavier est utilisé.
- **8) Empreinte génétique (biologique)**
  - Analyse du code génétique de l'individu.
  - Demande encore actuellement trop longtemps.

Authentification : Différentes créances Techniques biométriques : Conclusion

- **1)** Solution en rapide **développement.**
- **2) Solution qui peut-être efficace** mais qui présente aussi **des taux d'échecs** plus ou moins élevés.
- **3) Solution le plus souvent onéreuse** : dans l'état actuel des procédés.
- **4) Solution qui peut-être difficile à accepter** dans certains cas par l'utilisateur.

Protocoles d'authentification

L'authentification à mots de passe

Introduction
Authentification à mot de passe en UNIX
Authentification à mot de passe en réseaux

Authentification à mot de passe : Introduction

- **1) Utilisateur :** nom d'usager U,
  mot de passe secret P
- **2) Vérification:** fichier de correspondance U , P.
- **3) Stockage:** fichier de mots de passe U, P **en clair.**
  - Possible si le vérificateur est très bien protégé.
  - En général : protection trop faible.
- **4) Stockage** : fichier de mots de passe U, P **chiffré**
  - **Fonction de hachage a sens unique (oneway hash function) H = >** On stocke U, H(P).
  - **Authentification usager U, P :** P est immédiatement **haché H(P).**
  - **On compare la valeur obtenue à celle enregistrée.**
  - Le fichier peut être **accessible en lecture par tous** puisqu'on ne peut déduire P de H(P).

Authentification à mot de passe : Exemple d'authentification en UNIX

- **1) Version de base** : mot de passe 8 caractères.
- **2) Hachage à sens unique** : CBC-MAC-DES
  - **Clé utilisée** : Le mot de passe **8 caractères** -> une **clé de 56 bits** pour chiffrer avec l'algorithme du DES.
  - **Message utilisé** : une suite de 25 blocs de 64 bits à 0.
- **3) Fonctionnement concret du hachage:**
  - On **chiffre en DES avec** le mot de passe comme clé
  - Un texte qui est composé de **64 bits à 0**.
  - On **réitère 25 fois** sur les valeurs successives obtenues.
- **4) Le résultat est traduit en 11 caractères ASCII**
  - Placé dans un fichier (au départ /etc/passwd).
  - Accessible en lecture par tous les usagers.

Authentification à mot de passe : Authentification en UNIX : attaques

- **1) Décryptage "force brute"**
  - **Essayer tous les mots de passe**: pas impossible si l'intrus a pu recopier le fichier (calculateur puissant).
  - **Essayer des mots de passe tirés d'un dictionnaire** de mots de passe (prénoms, noms de chiens .. etc) => on peut casser jusqu'à 30% des mots de passe générés à la légère.
- **2) Solutions**
  - Interdiction de l'utilisation des mots de passe **courts** ou cassables facilement.
  - Interdiction de **plus de n tentatives d'accès en échec**.
  - **Temporisation longue** pour chaque essai.
  - **Mise en protection du fichier des mots de passe** : accès contrôlé par primitive système.

Authentification à mot de passe : Authentification en UNIX : salage

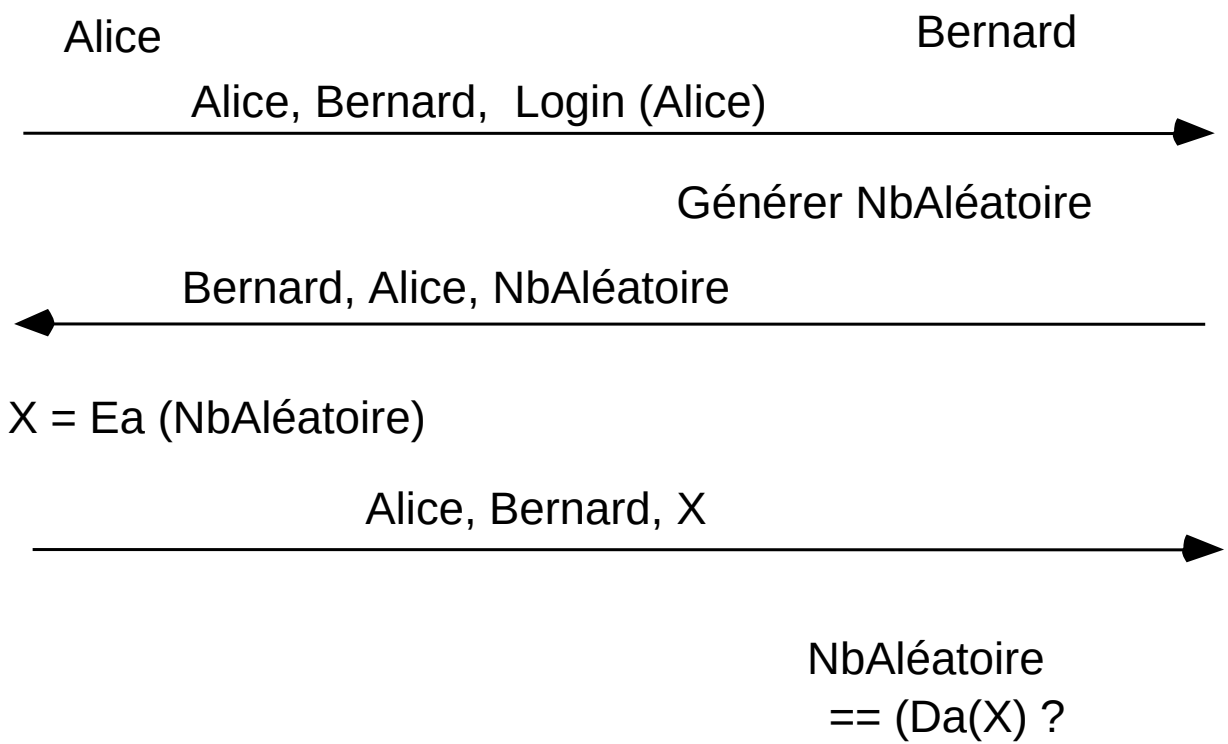
- **1) Clé complémentaire aléatoire : salage (‘Salt’)**
  - Adjonction au mot de passe P d'une clé complémentaire N (l'heure de création du mot de passe, une clé aléatoire)
  - On stocke dans le fichier des mots de passe protégé U , N , H(P , N). A chaque vérification on ajoute au mot de passe P de l'usager la clé N avant de crypter.
- **2) Salage UNIX:**
  - Clé complémentaire de 12 bits **("salt")** construite à partir de l'UID (code interne utilisateur) et de l'heure de génération du mot de passe.
  - Clé ajoutée au mot de passe de l'usager.
  - On multiplie par 4096 le nombre d'essais à réaliser pour chaque mot de passe par un pirate.

Authentification à mot de passe : Fonctionnement en réseau

- **Problème 1** : **assurer la confidentialité des mots de passe.**
  - **Solution 1** : Utilisation d'un chiffre à clé secrète pour protéger les mots de passe (accès LDAP protégé SSL).
  - **Solution 2** : Transmission d'un hachage du mot de passe avec une clé secrète (une forme de MAC : CHAP, Radius).
- **Problème 2** : un pirate pourrait écouter une authentification pour la réutiliser ultérieurement en forme cryptée (rejeu, "replay")
  - **Solution:** Utilisation d'un **nonce** (nombre aléatoire)
  - A chaque authentification on échange une valeur différente imprévisible.

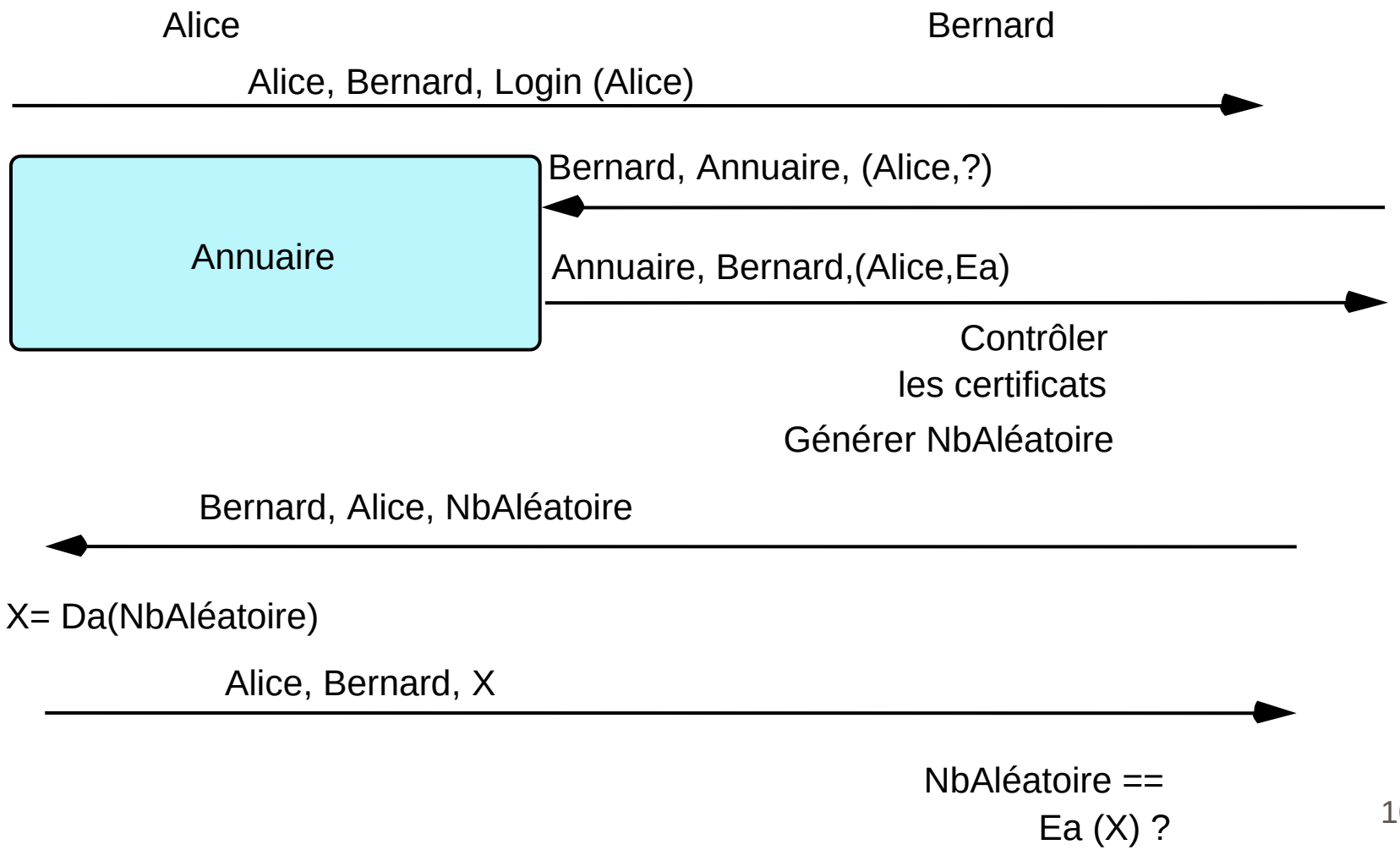
Authentification à clé secrète : Principes généraux d'une solution

- **Les deux entités** partagent une clé secrète Ea , Da
- **Bernard doit connaître toutes les clés** des usagers qui s'authentifient auprès de lui.
- **Inutilisable** si Alice et Bernard ne se connaissent pas.



Authentification à clé publique : Principes généraux d'une solution

- Les deux entités peuvent s'authentifier sans se connaître au préalable



Authentification à clés publiques Explication de la solution

- Bernard propose à Alice **un défi** qui est de crypter un nonce.
- **Il n'y a pas de mot de passe** au sens habituel: c'est la connaissance par l'utilisateur de sa clé privée qui sert de secret.
- Si **Alice retourne le nonce** crypté c'est elle puisque seule Alice connaît Da.
- Bernard **vérifie la cohérence de la valeur du nonce** ce qui empêche la répétition.
- Il faut que **l'annuaire des clefs publiques** (association de Alice et de sa clef publique) **soit accessible.**
- On utilise la commutativité du chiffre asymétrique retenu:

$$\mathbf{E\,(D(M))\ =\ D(E(M))\ =\ M}$$

Protocoles de sécurité :

Conclusion

Protocoles de sécurité :

- **1) Protocoles de sécurité et fonctions cryptographiques** :
  - un tout pour atteindre des propriétés de sécurité.
  - des implantations très nombreuses qui présentent des variantes souvent mineures des mêmes principes.
- **2) Protocoles assez stabilisés :** évolutions plus lentes qu'en cryptographie.
- **3) La preuve de la satisfaction des propriétés reste toujours délicate :**
  - Fonctions cryptographiques et complexité.
  - Attaques difficiles à prévoir.

Bibliographie

- **Bruce Schneier**, Cryptographie appliquée , Thomson Publishing, Paris 1995
- **Bruno Martin**, Codage, cryptologie et applications, Presses polytechniques et universitaires romandes 2004
- **Stéphane Natkin**, Les protocoles de sécurité de l'Internet, Dunod, 2002
- **Niels Ferguson , Bruce Schneier**, Cryptographie en pratique , Wiley 2003, Vuibert 2004, Paris 1995
- **A Menezes, P Van Oorschot, S Vanstone**, Handbook of applied cryptography, CRC Press Inc, 1997
- **Cours disponibles en ligne** et pages spécialisées sur des concepts, normes ou produits