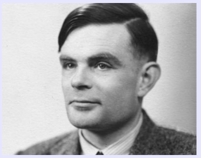
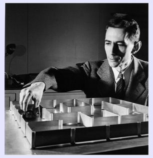
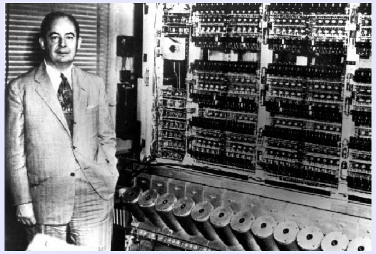




Information, Calcul et Communication

Module 3 : Systèmes

Information, Calcul et Communication
Sécurité des systèmes informatiques

Ph. Janson & J.-C. Chappelier

Motivation

L'univers numérique doit être sécurisé au même titre que le monde physique

Les affaires se traitent de plus en plus en ligne …

… donc de plus en d'argent et de pouvoir passent par Internet

… donc criminalité et conflits politiques se déroulent de plus en plus en ligne

… car ils suivent toujours argent et pouvoir

Principes de base

- **La sécurité totale n'existe pas** plus dans le monde informatique que dans le monde physique
- Dans les deux cas elle est
  - Une course aux armements entre mécanismes d'attaque et de défense
  - Un **compromis** entre le ***risque*** d'une attaque et le ***prix*** de la défense
- Comme dans toute situation de défense, les attaques visent les **maillons faibles**
  - Généralement entre le siège et l'écran (utilisateurs ou opérateurs des systèmes informatiques)
- L'**éducation** des utilisateurs et des opérateurs est donc essentielle
  ☞ C'est le but de ces deux leçons !

Objectifs de ces deux leçons

Comment sécuriser le monde numérique ?

- En quoi et comment les systèmes informatiques et leur contenu sont **menacés** et **menacent** indirectement les individus dans leur sphère privée ?
- Quels sont les principes de base à respecter et les **mécanismes** fondamentaux à déployer pour **protéger** l'information, les systèmes qui la traitent, et les réseaux qui la transportent ?
- Quels sont les moyens techniques utilisés (**cryptographie**) pour garantir *confidentialité*, *intégrité* des données et *responsabilité* des utilisateurs ?
- Quelles sont les principales **règles de bonne conduite** des utilisateurs et administrateurs de systèmes informatiques pour se protéger contre les hackers et leurs maliciels ?

Les menaces : 1. leurs objectifs

Objectifs des menaces sur les systèmes d'information :

- les **informations**
- les **applications** qui les gèrent
- les **logiciels** qui les hébergent
- les **ordinateurs** qui les exécutent
- les **réseaux** qui les relient
- les **bâtiments** qui les renferment

☞ et au travers de tout cela : les **personnes** concernées (utilisateurs)

Les menaces : 2. leur nature

Nature des menaces :

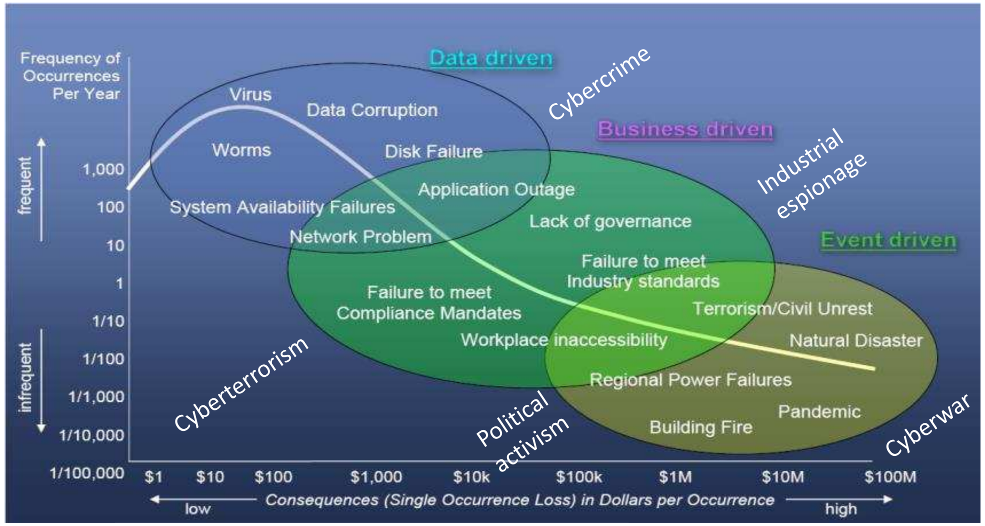
- le **vol** d'informations
- la **manipulation** d'informations
- la **destruction** d'informations
- le **démenti**
- l'**usurpation d'identité**
- le **contournement** des défenses

Les menaces : 3. leurs sources

Sources des menaces :

- **Environnementales** : catastrophes naturelles
  Note : des accidents de nature *environnementale* sont souvent source d'abus de nature *humaine* et d'attaques de nature *technique*.
- **Humaines**
  - internes :
    - erreurs
    - abus de privilèges
  - externes :
    - manipulation sociales (abus de confiance, mensonge, tromperie, corruption, etc.)
    - attaques physiques (espionnage, vol, sabotage, destruction, etc.)
- **Techniques**
  - attaques informatiques (par des humains) : exploitations de vulnérabilités logicielles
  - maliciels (logiciels malveillants : virus, vers, chevaux de Troie, etc.)

Les menaces : 4. ampleur...



Source: IBM Security Technology Outlook 2008

Les menaces : 4. ampleur... ...et relativité

- Coût annuel de la cybercriminalité : 0.5 à $1.5 \cdot 10^{12}$\$ (2016, suivant les sources)
- Nombre de vulnérabilités logicielles > 60 K (IBM)
- Nombre de maliciels identifiés > 150 M (Webroot.com, 2016)
- Nombre de sites web infectés > 1.3 M (Dasient.com, 2010)
- Taux de spam 50-60% (statista.com)
- Plus gros vol de données : $3 \cdot 10^9$ utilisateurs (Yahoo, 2013)
- 15'833 téléphones portables perdus dans le métro londonien en 2013 (source : McAfee à partir de données de Transport for London)
- statistiques en temps réel sur les attaques en cours :
  `https://cybermap.kaspersky.com/`
  `https://threatmap.checkpoint.com/ThreatPortal/livemap.html`
  `http://www.digitalattackmap.com/`

☞ aussi impressionnants que soient ces chiffres absolus, ils indiquent un **équilibre relatif entre coût des risques et prix des défenses**

Les défenses

Les menaces étaient :	les combattre exige :
le *vol* d'informations	**confidentialité** des informations
la *manipulation* d'informations	vérification de l'**intégrité** des informations
la *destruction* d'informations	**disponibilité** des informations
le *démenti*	**responsabilisation** des utilisateurs
l'*usurpation d'identité*	**authentification** des utilisateurs/des processus
le *contournement* des défenses	hiérarchisation des **autorisations** des utilisateurs/des processus

☞ L'ultime objectif : contrôler qui a quel droit

Destruction : équilibre menace/défense

Destruction

- Menace : la perte ou l'indisponibilité des données
- Défense : la réplication des données

  ☞ maintenir plusieurs copies (cohérentes !) des données

Équilibre menace/défense : **taux de réplication** :

- Tenir une seule autre copie sur une autre machine
  - Bien si la machine originelle tombe en panne
  - Pas suffisant si la machine originelle et la réplique tombent en panne/sont détruites
- Tenir *deux* copies sur d'autres machines
  - Bien si...
  - mais insuffisant si...
- ...

Destruction : degré de défense

1. taux de réplication

mais aussi :

2. Localisation des répliques : à coté/à distance (quelle distance ?)
3. Mise à jour des répliques : à chaque fois/par heure/par jour/...

☞ Choix du niveau de défense approprié (coût : argent, temps, pénibilité) adapté au risque de menace

Exemples extrêmes :

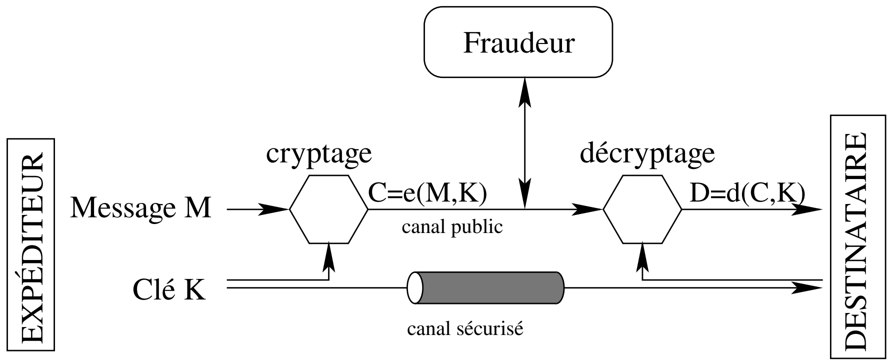
- *N* répliques lointaines mises à jour à chaque modification
- 1 réplique sur un disque à coté sauvegardé tous les soirs

Buts de la cryptographie (1)

Les menaces étaient :	les combattre exige :
le *vol* d'informations	**confidentialité** des informations
la *manipulation* d'informations	vérification de l'**intégrité** des informations
la *destruction* d'informations	*disponibilité* des informations
le *démenti*	**responsabilisation** des utilisateurs
l'*usurpation d'identité*	*authentification* des utilisateurs/des processus
le *contournement* des défenses	hiérarchisation des *autorisations* des utilisateurs/des processus

☞ Confidentialité, intégrité et responsabilité via la **cryptographie**

Cadre général (1)



$C = e(M, K)$ = message M chiffré avec la clé K
$e(M, K)$ = fonction de chiffrage
$d(C, K)$ = fonction de déchiffrage
$d(e(M, K), K) = M$

Buts de la cryptographie (2)

Confidentialité : protection des messages contre la lecture non autorisée :
impossible d'obtenir le message M à partir de $C = e(M, K)$ sans K

Intégrité :
impossible de substituer un autre message à la place de C

Responsabilité :
impossible d'attribuer un autre auteur à C (et donc à M)

☞ Le chiffrage devrait être universel, surtout à l'heure du « cloud » !

Cadre général (2)

Hypothèses :

- les algorithmes de chiffrage/déchiffrage sont connus de tous
- le fraudeur peut intercepter $C = e(M, K)$
- le fraudeur ne connaît pas la clé K

Niveaux d'attaque (« *cryptanalyse* » ; par difficulté décroissante) :

- trouver M ou K sachant seulement C $(= e(M, K))$;
- trouver K connaissant (M, C) pour certains M connus ;
- trouver K en pouvant connaître (M, C) pour des M choisis.

Chiffrage symétrique/asymétrique

Il y a deux grandes familles de crypto-systèmes :

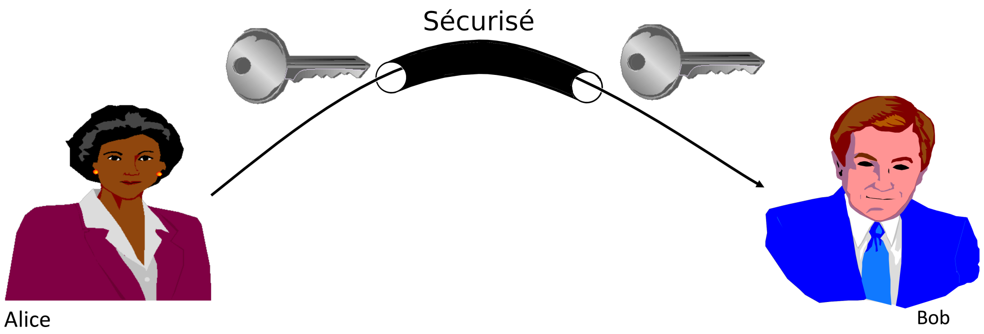
	Symétrique à clés secrètes	Asymétrique à clés publiques
Exemples :	**One-time pad** DES AES	**RSA** Diffie-Hellman courbes elliptiques
Confidentialité :	oui	oui
Intégrité :	oui	oui
Responsabilité :	non	oui

Note : on peut utiliser les deux en même temps (p.ex. envoi d'une clé privée d'un système symétrique par chiffrage asymétrique)

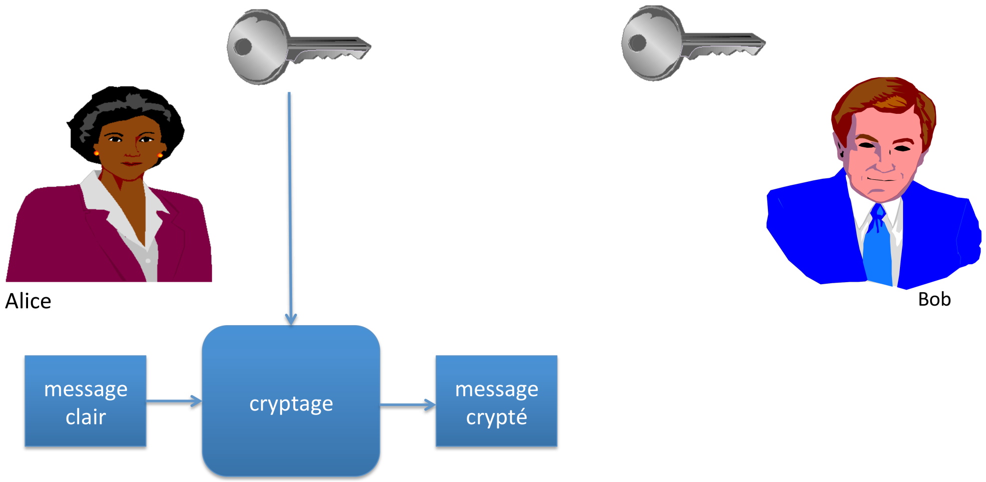
Chiffrage symétrique

- Il n'y a qu'une seule clé
- La clé est échangée entre les partenaires, à l'avance, via un canal sécurisé
- Le message est chiffré et déchiffré avec la même clé

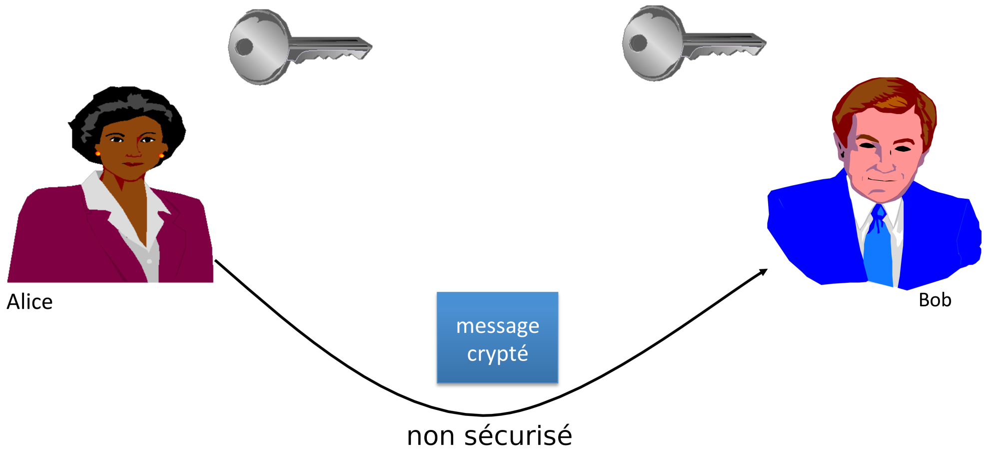
Chiffrage symétrique



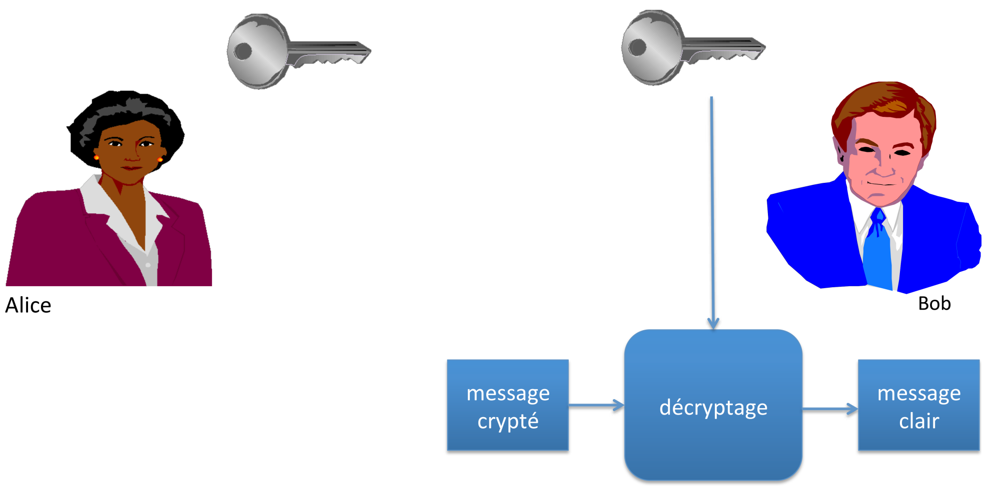
Chiffrage symétrique



Chiffrage symétrique



Chiffrage symétrique



Confidentialité parfaite

Un crypto-système est dit **parfait** si :
lorsque seulement le cryptogramme C est observé,
C ne donne aucune information sur M
c.-à-d. si M et C sont *indépendants* (au sens probabiliste).

Est-ce possible ?
Oui si :

1. il y a au moins autant de clés que de messages
2. l'entropie des clés est supérieure ou égale à celle des messages

Exemple : « One-time pad »

« One-time pad »

Messages, clés, cryptogrammes = séquences binaires de longueur n

clé : $K = k_1 k_2 ... k_n$ = séquence aléatoire de n bits :

$$p(k_i = 0) = 0.5$$

Chiffrage/déchiffrage :

$$c_i = m_i \oplus k_i$$

($\oplus$ = XOR ; c.-à-d. addition binaire bit à bit, sans retenue)

$$\begin{aligned} p(c_i = 0) &= p(m_i = 0 \text{ \underline{et} } k_i = 0) + p(m_i = 1 \text{ \underline{et} } k_i = 1) \\ &= 0.5 \times p(m_i = 0) + 0.5 \times p(m_i = 1) \\ &= 0.5 \end{aligned}$$

« One-time pad » : exemple

On veut envoyer $M = 01101001$

On choisit comme clé (hasard) : $K = 11001110$

On envoie :

$$\begin{array}{rll} & 01101001 & (M) \\ \oplus & 11001110 & (K) \\ \hline & 10100111 & (C) \end{array}$$

Déchiffrage :

$$\begin{array}{rll} & 10100111 & (C) \\ \oplus & 11001110 & (K) \\ \hline & 01101001 & (D = M) \end{array}$$

Confidentialité parfaite (2)

Pour un crypto-système parfait :

- il faut au moins autant de clés que de messages ;
- l'entropie des clés doit être supérieure ou égale à celle des messages.

Les **deux** aspects doivent être vérifiés.

☞ Conséquence : pour un système parfait, il faut des clés **suffisamment complexes** !

Système parfait $\neq$ système *pratique* (où la clé doit être assez simple et facilement réutilisable)

Sécurité algorithmique

sécurité parfaite :

les données observables (C) ne contiennent pas assez d'information pour craquer le code (M ou K)

sécurité algorithmique (par complexité) :

*les données observables permettent en théorie de craquer le code, **mais** calculer M ou K à partir de C est **difficile***

☞ utiliser des problèmes au moins aussi difficiles que tous ceux de NP (théorie de la complexité)

Fonctions à « sens unique »

Fonctions à sens unique :
f est facile à calculer, mais f^{-1} est difficile (au sens algorithmique).

Utilisation en cryptographie :

- fonction de chiffrage : $C = e(M, K) = e_K(M)$
- fonction de déchiffrage : $M = d(C, K) = e_K^{-1}(C)$

de sorte que e_K soit à « sens unique ».

☞ le calcul $C \mapsto M$ sera trop difficile à faire en pratique (sans K).

Exemple : DES

Principe de DES

Exemple de fonction à sens unique en cryptographie symétrique : DES

Problème NP-complet : résolution de systèmes d'équations non-linéaires dans GF(2) (= $\mathbb{Z}/2\mathbb{Z}$)

Exemple :

$$x_1\,x_4 \oplus x_2\,x_3\,x_5 = 1$$
$$x_2\,x_3 \oplus x_1\,x_3\,x_4 = 1$$
$$x_1\,x_3 \oplus x_1\,x_2\,x_5 = 1$$

a pour solution : $(x_1, x_2, x_3, x_4, x_5) = (1, 0, 1, 1, 0)$

Problème NP-complet (équivalent à SAT)

(plus de détails en annexe)

Sécurité de DES

Craquer DES = NP-complet :
résolution par substitution : système de n équations polynomiales

MAIS

Le système DES est sûr tant que (P $\neq$ NP, et surtout)
n est suffisamment grand

« suffisamment grand » ? 56 bits (en 1978) ?... ...aujourd'hui : ?...
remplacé par d'autres algorithmes (AES)

Autre **problème de fond** :

> *pour un problème NP-complet, quelqu'un pourrait trouver la solution par hasard (cf définition de NP)*

☞ la sécurité ne sera **jamais** garantie pour un cas précis !

Fonctions à « sens unique » : autre exemple (hors cryptographie)

Exemple du stockage des mots de passe (**authentification**) :

- Comment garantir l'identification SANS stocker les mots de passe en clair (!) ?

Solution :

- on stocke les versions chiffrées par une fonction à sens unique.
- on chiffre le mot de passe entré à chaque login et on compare les chiffrages

Il est alors difficile en pratique de trouver les mots de passe à partir du fichier des mots chiffrés.

Exemple simple :
mots de passe = $(n1, n2)$ (deux entiers)
mot de passe chiffré : $e = n1 \times n2$: facile à calculer
déchiffrage : factorisation de e : difficile !

Cryptographie asymétrique (ou « à clés publiques »)

Problème : comment transmettre (rapidement) les clés de façon sûre ?

Une solution : ne pas transmettre de clé du tout ! !
☞ **systèmes à clés publiques**

Mais alors, chaque paire d'utilisateurs doit avoir une clé distincte :
n utilisateurs $\Rightarrow n(n-1)$ clés *différentes*
☞ difficulté de les générer/mémoriser

Il faut un schéma systématique pour la *distribution des clés* !

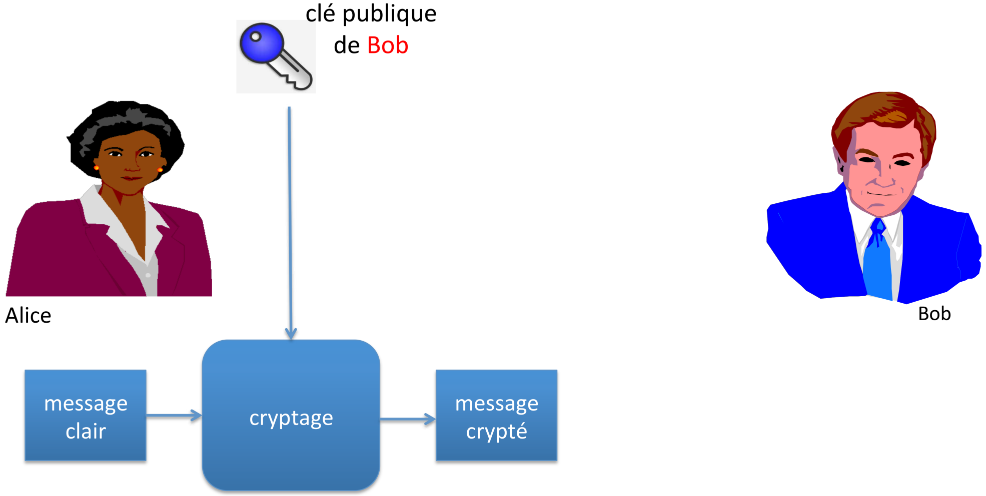
Exemple dans ce cours :

- RSA (Rivest-Shamir-Adleman, 1978)

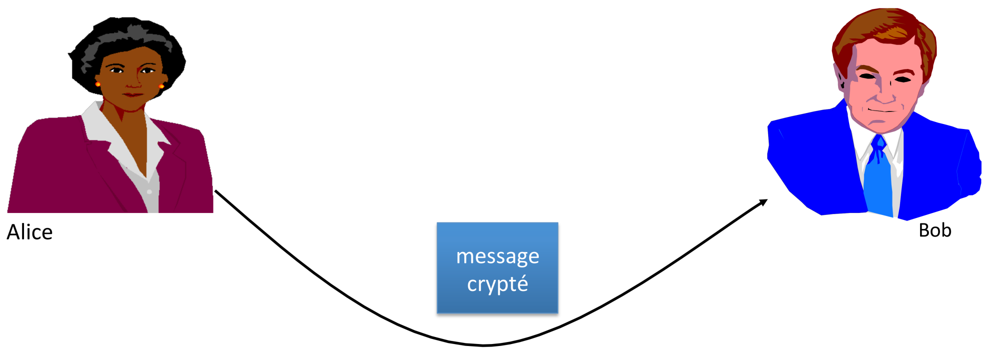
Cryptographie asymétrique

- Chaque utilisateur a deux clés
  - Une clé privée
  - Une clé publique
- Le **chiffrage** se fait avec la clé **publique** du *destinataire*
- Le **déchiffrage** se fait avec la clé **privée** du *destinataire*

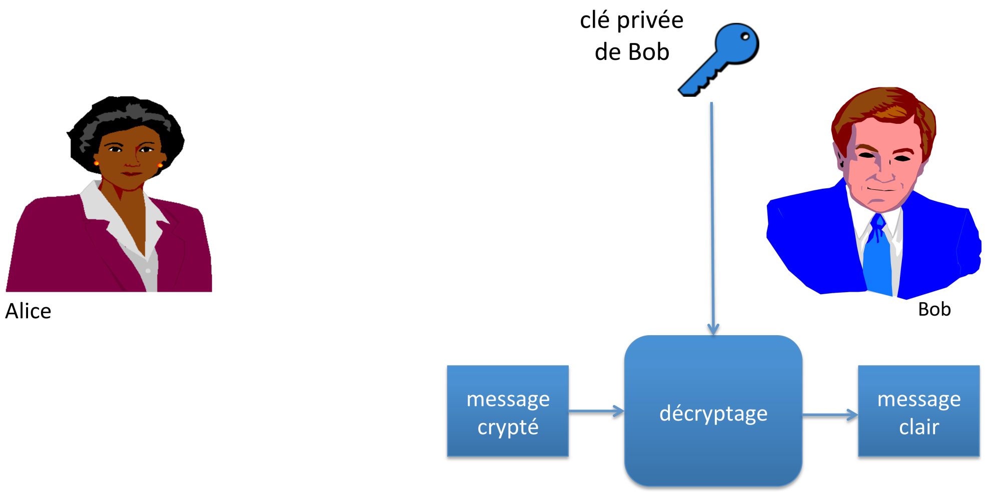
Chiffrage asymétrique



Chiffrage asymétrique



Chiffrage asymétrique



RSA (Rivest-Shamir-Adleman, 1978)

M, C = nombres entiers (modulo n)

- Choisir 2 nombres premiers p, q tels que tout message $M < p \cdot q$
- Calculer $n = pq$ et $m = (p-1)(q-1)$
- Choisir un $d < m$ qui n'a aucun facteur en commun avec m
- Calculer e tel que $ed = 1 \mod m$.
- Publier e et n
- Garder secret d, m, p et q.

Chiffrage : $C = e(M) = M^e \mod n$

Déchiffrage : $M = e^{-1}(C) = C^d \mod n$

Pour communiquer, j'utilise **la clé publique de l'autre**.

Exemple (simpliste) de RSA

Prenons $p = 5$ et $q = 11$ ☞ $n = 55$ et $m = (5-1)\cdot(11-1) = 40$

Note : bien sûr, ici p et q sont triviaux à deviner connaissant n car c'est un exemple simpliste de cours. En réalité c'est *pratiquement* infaisable (= infaisable en un temps raisonnable sur les machines actuelles [sauf preuve du contraire !]) pour des nombres à, disons, 600 chiffres (en décimal) car il n'y a pas d'algorithme efficace connu et que la recherche exhaustive prendrait littéralement des siècles.

Choisissons p.ex. $d = 27$ ($= 3^3$, qui n'aucun facteur commun avec $m = 2^3 \times 5$)
☞ $e = 3$

On veut envoyer le message 000010 (c.-à-d. 2 ; pourquoi sur 6 bits ici ?)

On envoie alors : $C = 2^3 = 8 \mod 55$ (on envoie donc : « 001000 »)

Déchiffrage :

$$D = 8^{27} = 2 \mod 55$$

Calcul de l'exponentielle discrète

L'exponentielle discrète est facilement calculable,
(nécessitant au plus $\Theta(\log_2 n)$ multiplications)
en utilisant l'algorithme « mettre au carré et multiplier » :

Exp
entrée : x, $n \geq 1$, p sortie : $x^n \mod p$

Si $n = 1$
    sortir : $x \mod p$
Sinon, si n est pair
    sortir : Exp$(x^2 \mod p,\ n/2,\ p) \mod p$
Sinon
    sortir : $x \times$ **Exp**$(x^2 \mod p,\ (n-1)/2,\ p) \mod p$

Exemple de calcul de l'exponentielle discrète

Exemple :
$x = 8$, $n = 27$, $p = 55$

$$\begin{aligned} 8^{27} \mod 55 &= 8 \times 64^{13} = 8 \times 9^{13} \mod 55 \\ &= 8 \times 9 \times 81^{6} = 17 \times 26^{6} \mod 55 \\ &= 17 \times 676^{3} = 17 \times 16^{3} \mod 55 \\ &= 17 \times 16 \times 256^{1} = 52 \times 36 \mod 55 \\ &= 2 \mod 55 \end{aligned}$$

Pourquoi RSA fonctionne ?

RSA fonctionne si $D = M$, c.-à-d. $M^{ed} = M \mod n$.

Puisque $ed = 1 \mod m$, il existe $k > 0$ tel que $M^{ed} = M \cdot M^{k\,m}$.
En outre, comme p et q sont des nombres premiers (th. d'Euler-Fermat) :

$$M^{p-1} = 1 \mod p$$
$$M^{q-1} = 1 \mod q$$

ainsi :

$$M^{ed} = M \cdot M^{k\,m} = M \cdot M^{k\,(p-1)(q-1)} = M \cdot \left(M^{p-1}\right)^{k\,(q-1)} = M \cdot 1 \mod p$$

De même : $M^{ed} = M \mod q$.

Or, si p et q sont deux nombres premiers et
si $x = y \mod p$ et $x = y \mod q$, alors $x = y \mod (pq)$.

Donc, nous avons bien $M^{ed} = M \mod n$.

Et comment résoudre $e \cdot d = 1 \mod m$?

Trouver e et k tels que $e \cdot d - k \cdot m = 1$ peut être fait en utilisant l'**algorithme d'Euclide de plus grand diviseur commun**
(puisque le plus grand diviseur commun de d et de m est précisément 1).

Soient $\mathbf{u}$, $\mathbf{v}$ et $\mathbf{t}$ des vecteurs de $\mathbb{Z}^2$ (c.-à-d. des paires de nombres entiers).

L'étape d'initialisation de l'algorithme consiste en $\mathbf{u} = (0, m)$, $\mathbf{v} = (1, d)$.

La condition d'arrêt est que la deuxième composante v_2 de $\mathbf{v}$ égale 1.
Dans ce cas, la première composante est $v_1 = e$,
c.-à-d. qu'à la fin de l'algorithme $\mathbf{v} = (e, 1)$.

Après l'étape d'initialisation, l'algorithme continue en boucle jusqu'à ce que la condition d'arrêt soit remplie :
$\mathbf{t} \leftarrow \mathbf{u} - r\,\mathbf{v}$,
$\mathbf{u} \leftarrow \mathbf{v}$,
$\mathbf{v} \leftarrow \mathbf{t}$
avec $r = \frac{u_2}{v_2}$

Exemple résoudre $27e = 1 \mod 40$?

Trouvons e tel que $27\,e = 1 \mod 40$ (c.-à-d. $d = 27$ et $m = 40$) :

u	**v**	r	**t**
$(0,40)$	$(1,27)$	$\frac{40}{27} = 1$	$(-1,13)$
$(1,27)$	$(-1,13)$	$\frac{27}{13} = 2$	$(3,1)$
$(-1,13)$	$(3,1)$	(stop)	

ainsi $e = 3 \mod 40$.

Sécurité de RSA

Une voie pour craquer RSA : trouver les facteurs de n : p, q
(permet de tout reconstruire)

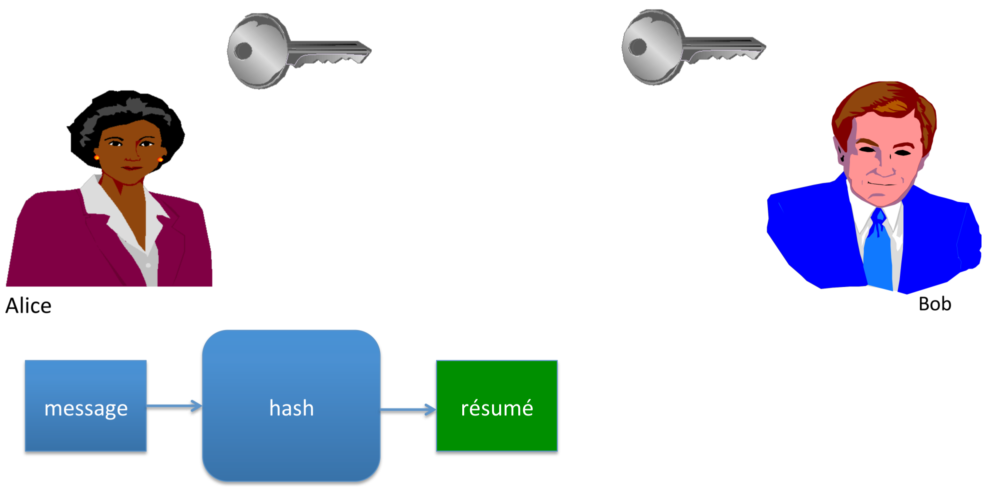
Trouver les facteurs premiers d'un nombre est un problème difficile, mais faisable pour des $n < 1024$ bits.

MAIS il n'est pas sûr qu'il soit *nécessaire* de trouver les facteurs de n pour craquer RSA !

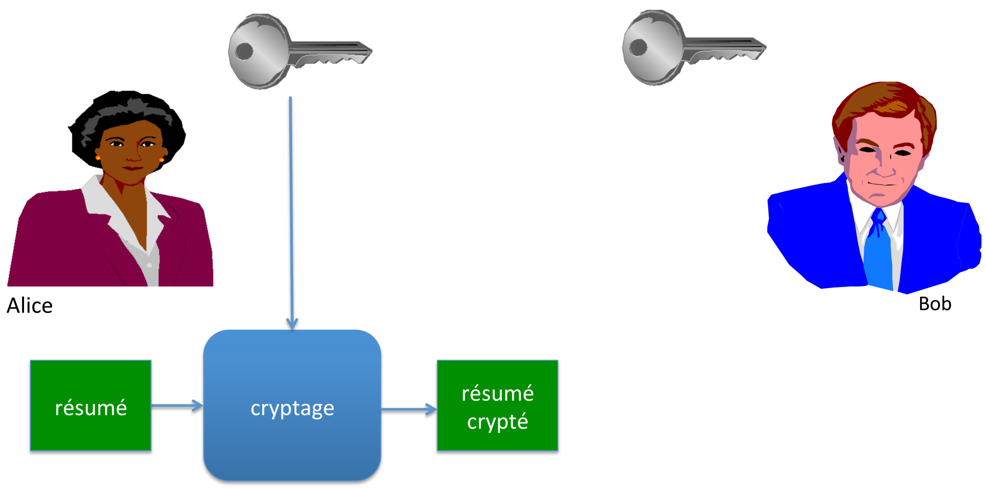
Intégrité

- **Menace :** modification des données
- **Défense :** résumé confidentiel

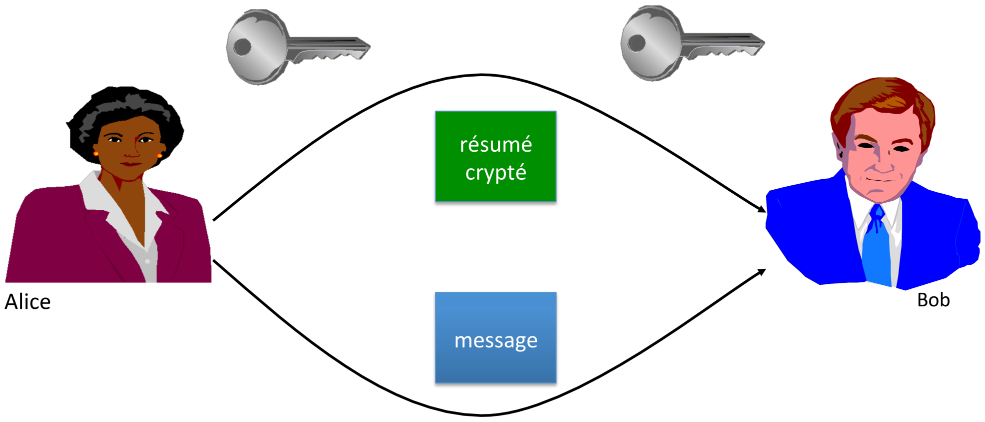
Intégrité à base de cryptographie symétrique



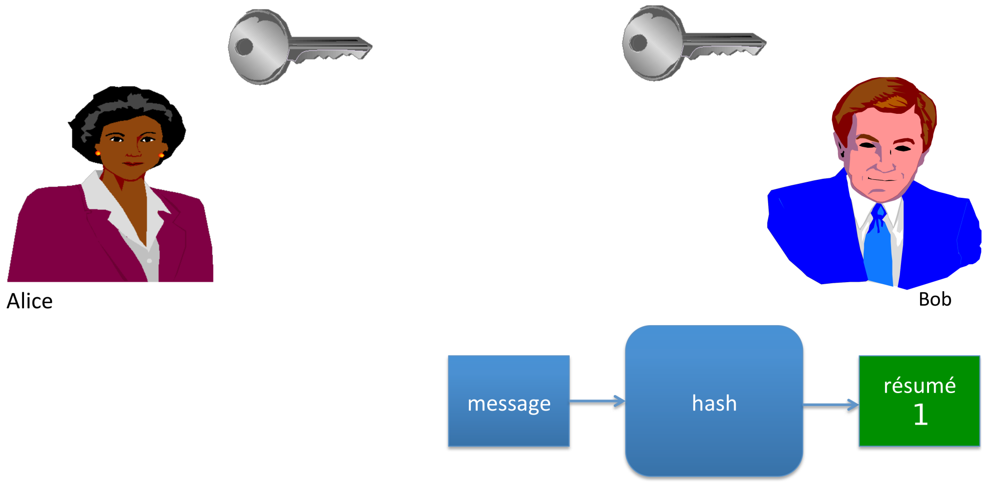
Intégrité à base de cryptographie symétrique



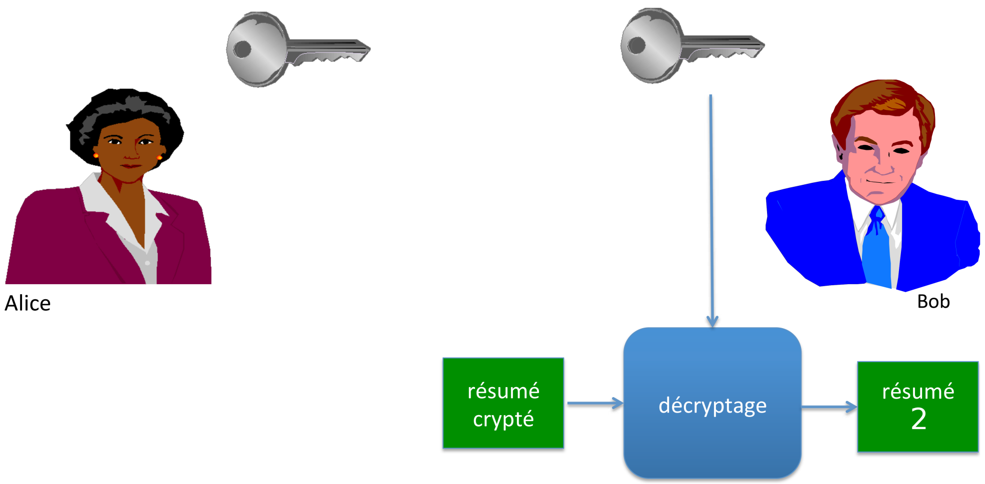
Intégrité à base de cryptographie symétrique



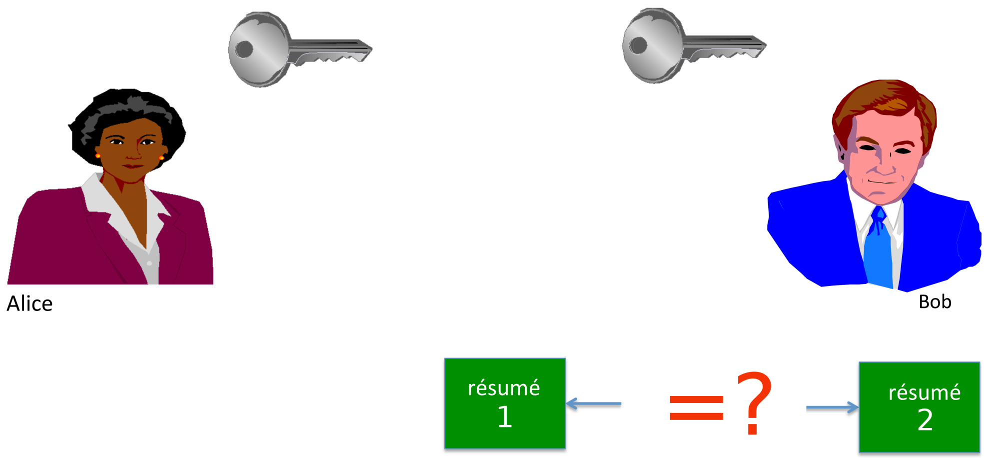
Intégrité à base de cryptographie symétrique



Intégrité à base de cryptographie symétrique



Intégrité à base de cryptographie symétrique



Responsabilité

- **Menace :** démenti
  (« *ce n'est pas moi qui ai envoyé ça* »
  ou « *c'est lui qui a envoyé ça* »)
- **Défense :** signature digitale

Rappel :

	Symétrique à clés secrètes	Asymétrique à clés publiques
Confidentialité :	oui	oui
Intégrité :	oui	oui
Responsabilité :	non	oui

Responsabilité contre Confidentialité

Responsabilité : s'assurer ou prouver que M vient bien de son auteur.

Probabilité d'infraction = probabilité de créer un message correct (« acceptable ») alors qu'on n'est pas autorisé.

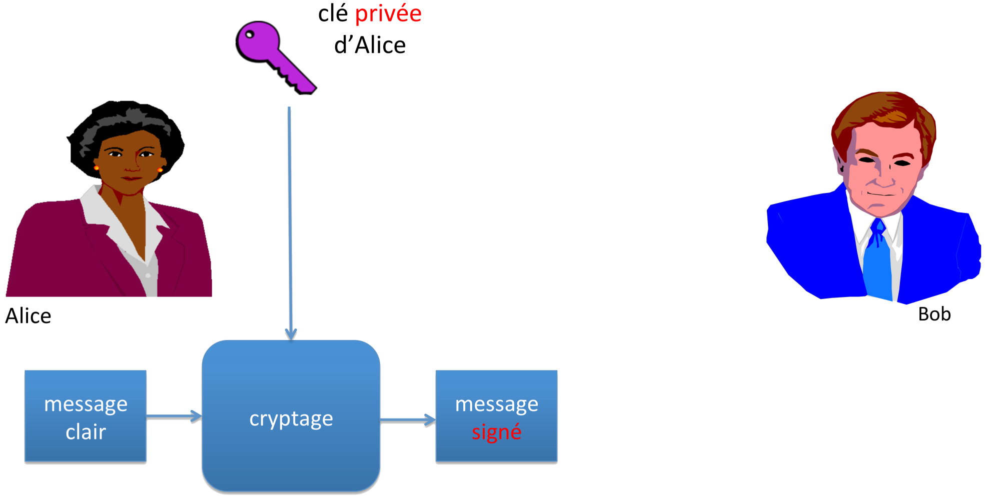
On peut prouver que pour avoir une petite probabilité d'infraction, il faut que le cryptogramme donne **beaucoup d'information** sur la clé.

Or, pour assurer la confidentialité, on aimerait le contraire !

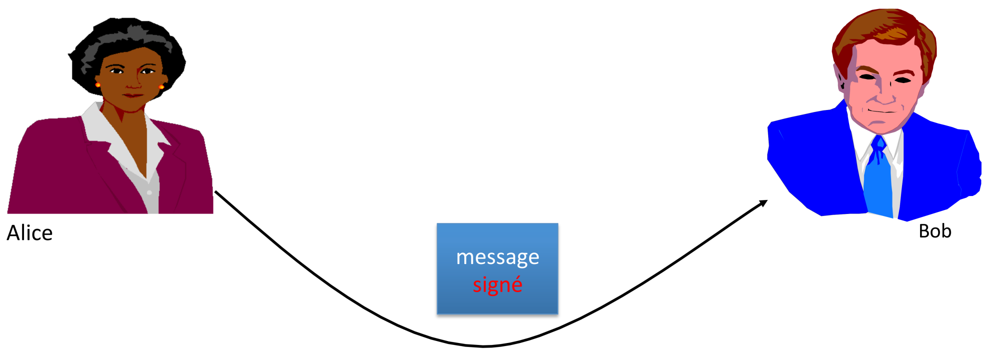
Du strict point de vue de la théorie de l'information, **responsabilité et confidentialité sont incompatibles**...

...donc :
Responsabilité $\Longrightarrow$ sécurité imparfaite :
utilisation de la sécurité *algorithmique*

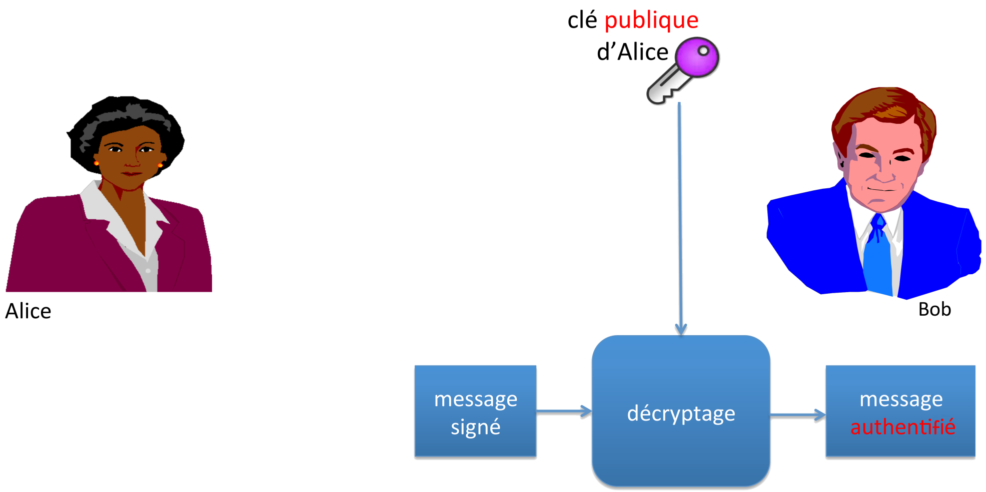
Responsabilité avec cryptographie asymétrique



Responsabilité avec cryptographie asymétrique



Responsabilité avec cryptographie asymétrique



Cryptographie asymétrique : responsabilité et intégrité

Le gros avantage du système de *responsabilité* de la cryptographie asymétrique est que nous assurons « pour le même prix » également l'**intégrité** du message signé :

la signature sert aussi de résumé :

le déchiffrage (correct) du message signé garantit en effet à la fois

- le message d'origine (intégrité)
- et son auteur (responsabilité)

car

- seul M a pu produire $d_A(M)$ (sinon le crypto-système serait ambigu)
- seul-e A a pu produire $d_A(M)$ (sinon le crypto-système n'est pas sûr)

Note : et si l'on chiffre pas le message signé, tout le monde peut faire cette double vérification (intégrité et responsabilité).

Signature dans un système RSA

Utilisateur A envoie M à B (message codé $C = e_B(M)$)

Signature :

$$S(M) = e_B(d_A(M))$$

Vérification :

$$e_A(d_B(S(M))) \stackrel{?}{=} M$$

Cela présuppose que $d_A(M)$ est dans le domaine de e_B, c'est-à-dire pour RSA que $d_A(M) < n_B$.

En pratique, on ne signe pas M lui-même, mais un résumé unique (hash) de M

Ce que j'ai appris aujourd'hui

- Savoir identifier les menaces et connaître les niveaux de défense appropriés
  la sécurité totale n'existe pas !
- Principes de base de la cryptographie
- Protection parfaite .vs. algorithmique
- Confidentialité : One-time pad et RSA
- Intégrité et Responsabilité dans un crypto-système

La suite (et fin !) d'ICC

Semaine prochaine :

jeudi à 11h15 (en CO1) : fin du cours :

- Authentification
- Attaque de mots de passe
- Autorisation
- Règles de bonne conduite

vendredi de 13h15 à 16h00 : examen

ANNEXE

(base de) DES (2)

Soit M un message binaire de $2n$ bits.

- M est divisé en deux parties de longueur égale (n) : $M = (M_0, M_1)$
- on utilise $d-1$ clés : $k_1, ..., k_{d-1}$ chacune de m bits
- f est une fonction de $GF(2)^m \times GF(2)^n$ dans $GF(2)^n$:

$$f(x_1, \cdots, x_m, y_1, \cdots, y_n) = (z_1, \cdots, z_n)$$

- f est non-linéaire en x (multiplications entre des x_i et des x_j)

(base de) DES (3)

Chiffrage par transformations itératives, $i = 2...d$:

$$M_i = M_{i-2} \oplus f(k_{i-1}, M_{i-1})$$

On envoie :

$$C = (M_{d-1}, M_d)$$

Déchiffrage par transformation, $i = d...2$:

$$M_{i-2} = M_i \oplus f(k_{i-1}, M_{i-1})$$

(On refait simplement les mêmes calculs dans l'autre sens.
Notez que, dans GF(2), $\ominus$=$\oplus$)

(base de) DES : exemple, chiffrage

Considérons la fonction non linéaire suivante ($n = 3$ et $m = 3$) :

$$f(x_1, x_2, x_3, y_1, y_2, y_3) = (x_1\, x_2\, y_1\, y_2, \quad x_2\, x_3\, y_1\, y_3, \quad (x_1 \oplus x_2)\, y_1\, y_3)$$

et choisissons la clé $K = $ 101011 ($d = 3$) :

$$K_1 = 101, \;\; K_2 = 011$$

On veut envoyer M = 101111 :

$$M = 101111 \longrightarrow M_0 = 101, \quad M_1 = 111$$

Itérations :

$$\begin{aligned} M_2 = M_0 \oplus f(K_1, M_1) &= (1,0,1) \oplus f((1,0,1),(1,1,1)) \\ &= (1,0,1) \oplus (0,0,1) && = (1,0,0) \end{aligned}$$

$$\begin{aligned} M_3 = M_1 \oplus f(K_2, M_2) &= (1,1,1) \oplus f((0,1,1),(1,0,0)) \\ &= (1,1,1) \oplus (0,0,0) && = (1,1,1) \end{aligned}$$

Donc finalement, $C = (M_2, M_3) =$100111 est envoyé.

(base de) DES : exemple, déchiffrage

On reçoit C = 100111 :

$$C = 100111 \longrightarrow M_2 = 100, \quad M_3 = 111$$

Itérations :

$$\begin{aligned} M_1 = M_3 \oplus f(K_2, M_2) &= (1,1,1) \oplus f((0,1,1),(1,0,0)) \\ &= (1,1,1) \oplus (0,0,0) && = (1,1,1) \\ M_0 = M_2 \oplus f(K_1, M_1) &= (1,0,0) \oplus f((1,0,1),(1,1,1)) \\ &= (1,0,0) \oplus (0,0,1) && = (1,0,1) \end{aligned}$$

Donc finalement, $D = (M_0, M_1) =$101111 a été envoyé.