Chap 1. Introduction* (Durée 1h).

§1. Contenu, description et objectifs du cours.

§2. Notions d'arithmétique et Eléments d'informatique théorique. (compléments algorithmiques)

* : optionnel. Durée totale ≈ 30 h.

Références

1) • Paar §.

2) • B. Martin. Codes, cryptographie

3) • D. R. Stinson. Cryptography : Theory and Practice. 1995.

4) Buchmann. Intro to cryptography.

---

§1 • L'objectif du cours est d'avoir une bonne compréhension des mécanismes cryptographiques, sans trop exiger de l'auditoire, en vue d'implémenter des applications par le futur ingénieur. L'un des choix est d'adapter l'aspect mathématique en introduisant les notions nécessaires par besoin.

• Le chap 2. décrit la cryptographie symétrique, à bloc variable et fixe. L'étude des DES (et AES)

illustre bien la cryptographie symétrique (mécanismes de diffusion, confusion). En pratique, les modes opératoires permettant l'utilisation des primitives de cryptage et décryptage. On peut très bien, par la suite introduire le protocole MAC (Passage authentification) avec le hachage (mais c'est mieux de le faire au chap. signature et hachage.)

- le chap 3. qui est fondamental étudie en détail les primitives à clé publique, illustrés par RSA, Protocole Diffie-Hellman (PDH), et une introduction aux courbes elliptiques. C'est dans ce chapitre que les éléments mathématiques sont importants: On donne juste le nécessaire (notions arithmétiques, notions de groupes, ...).
- Suite logique au chapitre 3, le chap 4. étudie la signature numérique, le hachage et le MAC. reprenant les services de sécurité cryptographique.
- le chap. 5 traite la gestion de clé (établissement, transport) et introduit la notion de certificat.

§ 2. Rappels

→ arithmétique : (thm de Fermat, Euler, ϕ, ...). 1.5

→ Groupes, Groupes cycliques.

→ classes P, NP, classes probabilistes et intérêt pour la cryptographie.

2

Nous avons délibérément suivi le cours de la réf 1). D'autres bons réf. existent on peut consulter la réf 2) ou 3) un peu plus pointues. Une fois absorbées ces bases pour l'étudiant, il peut se diriger vers plusieurs directions, notamment la cryptographie à base d'Identité (IBE : identity-Based cryptography)

→ voir les sites de la réf 1) et 2).

→ Propositions de projets.

chap 2. Cryptographie Symétrique.

✓ §1. Cryptage à longueur variable. (stream ciphers).

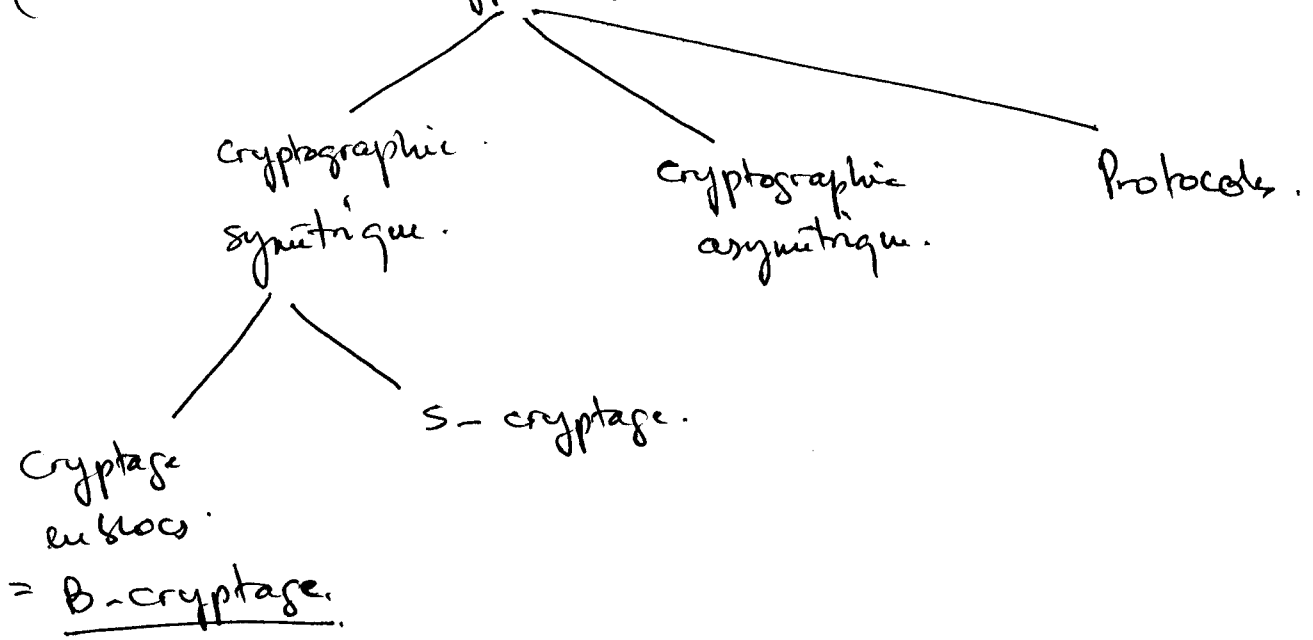
✓ §2. DES.

§3*. AES. (Projet).

✓ §4. Modes opératoires.

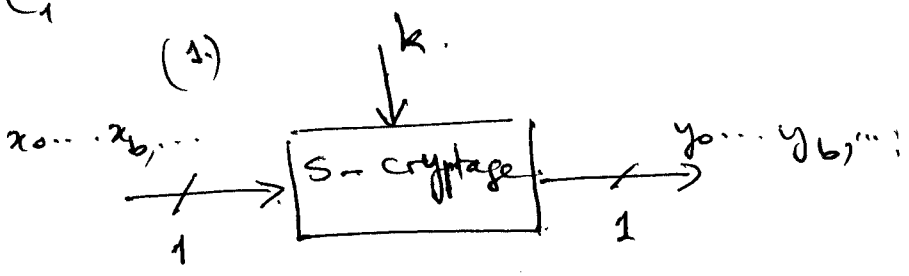
§1. Cryptage à longueur variable (Stream ciphers).
= S-cryptage.

(1) • Classification : Cryptographie.

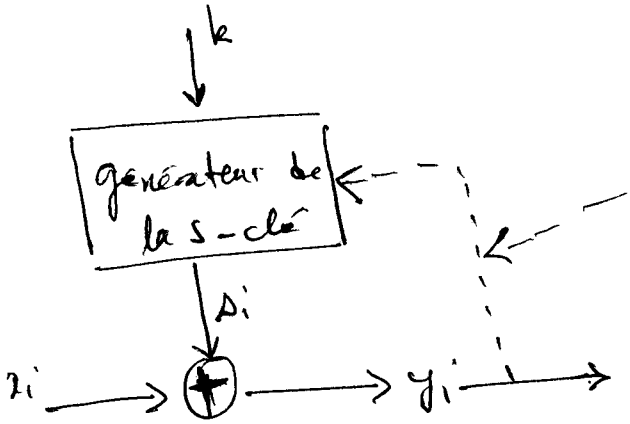


(2) • S-cryptage Vs. B-cryptage :

(1)



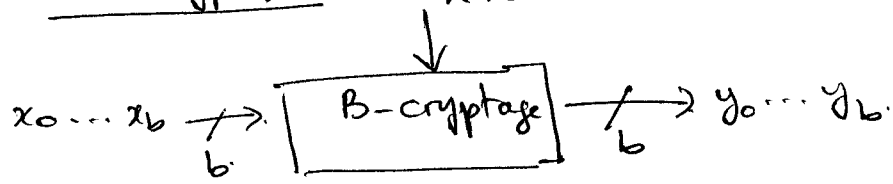
→ cryptage de bits individuellement.
avec une S-clé k.
appliqué au clair $x_0, \dots, x_b, \dots$



→ S-cryptage asynchrone : dépend du chiffré
Sinon synchrone

⇒ voir aussi suite générateur plus loin.

(2) $_2$ **B-cryptage** : k : clé



on chiffre un bloc de bits, un à la fois avec la même clé k

taille de bloc : 128 bits (16 bytes) pour AES
64 b. (8 bytes) " DES, 3DES

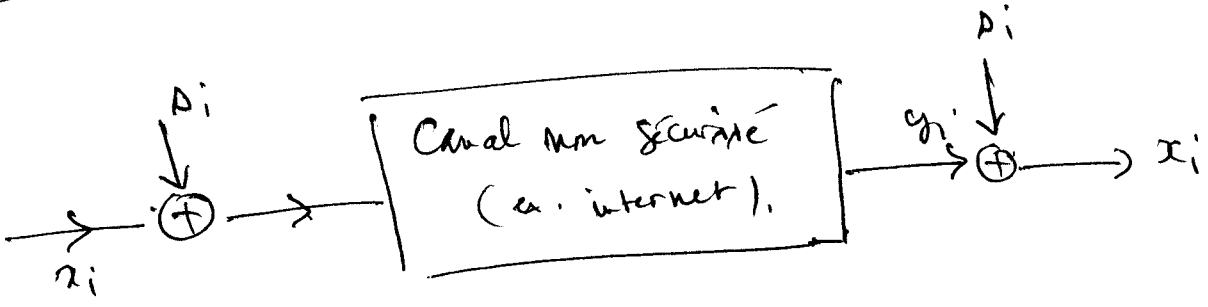
En pratique : ✓ B-cryptage plus utilisé que les-cryptage
✓ Mais le s-cryptage parfois rapide (ex. tél. cellulaire demandant moins de ressource

(3) • **s-cryptage et décryptage** :

entrée : $x_i, y_i, s_i \in \{0,1\}$.

cryptage : $y_i = e_{s_i}(x_i) = x_i + s_i \mod 2$

décryptage : $x_i = d_{s_i}(y_i) = y_i + s_i \mod 2$.



→ rapide.

→ Comment obtenir la suite (s_i) ? C'est la base de la sécurité de ce système

(4) Génération de nombres aléatoires.

(RNG : Random Number Generator).

(4.1). 3 types de RNG :

- **True RNG :** TRNG. Caractérisé par le fait que la suite ne peut être reproduite. Par exemple : flipper une pièce 100 fois : 2^{100} possibilités. TRNG utilisé pour engendrer des clés de session

- **Pseudo-Random :** PRNG. génère une suite (s_i) à partir d'une "base", comme :

$$\begin{cases} s_0 = \text{base} \\ s_{i+1} = f(s_i) \quad i = 0, 1, 2, \dots \end{cases}$$

ou. $s_0 = \text{base}$. et $s_{i+1} = f(s_i, s_{i-1}, \dots, s_{i-t})$, t entier fixé.

Exemple. $\begin{cases} s_0 = \text{"base"} \\ s_{i+1} = a \cdot s_i + b \mod m \end{cases}$; $a, b \in \mathbb{N}$. $i = 0, 1, 2, \dots$

Sortie de PRNG. approxime statistiquement. T-RNG.

- **Cryptography Secure PRNG :** CSPRNG.

C'est un PRNG avec la propriété :

Etant donné n bits de (s_i). consécutifs, $s_{i+1}, \dots, s_{i+n}$, il est difficile de trouver un algorithme polynomial calculant s_{i+n+1} avec une probabilité de succès $> \frac{1}{2}$.

(4.2) one-time Pad (OTP)

Un cryptosystème est sûr inconditionnellement s'il ne peut être cassé même avec une infinité de Ressources calculatoires.

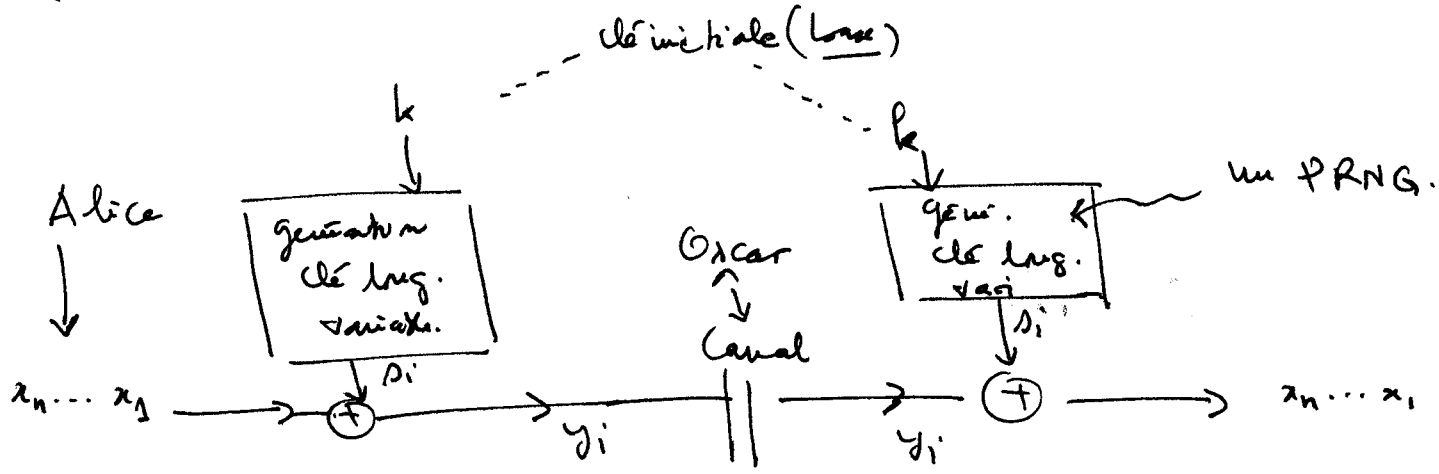
L'exemple est

OTP:

1. k_i est engendré par un TRNG.
2. La clé est connue que de A et B.
3. chaque k_i est utilisé une seule fois

Sûr mais impratiquable : $\log(\text{clé}) = \log(\text{texte})$.

(4.3). Vers le n-cryptage pratique.



- **Def.** Un cryptosystème est calculatoirement sécurisé si le meilleur algorithme connu pour le casser exige t opérations.
- Utilisant une clé k connue par A et B ($\sim$ 100 bits), et on utilise un PRNG pour engendrer (k_i) avec k base.

Exemple. **Insuffisance de PRNG.** — Un Exemple d'Attaque.

Supposons que $\begin{cases} S_0 = \text{base} \\ S_{i+1} = A S_i + B \pmod m, \quad i \geq 0. \end{cases}$

où $|m|_2 \sim 100$ bits, $S_i, A, B \in \{0, 1, \dots, m-1\}$.

module m public ; (A, B) secret (avec éventuellement S_0),

$\Rightarrow$ taille de la clé $\sim$ 200 bits.

suffisant contre une attaque exhaustive

Alice peut chiffrer selon la figure précédente.

$y_i = x_i + \Delta_i \pmod 2$ où Δ_i fait partie des bits représentant un certain S_j.

- Hypothèse : O sait que x contient 300 bits de clair et bien sûr le chiffré. O peut calculer les 300 bits de la S-clé :

$$\Delta_i = y_i + x_i \pmod m \quad i = 1, \dots, 300.$$

D'où O connaît : $S_1 = (\Delta_1 \cdots \Delta_{100})$, $S_2 = (\Delta_{101} \cdots \Delta_{200})$,

et $S_3 = (\Delta_{200} \cdots \Delta_{300})$. Je peut écrire :

$$\left\{ \begin{array}{l} S_2 \equiv A S_1 + B \pmod m \\ S_3 = A S_2 + B \pmod m. \end{array} \right\} \text{ O inconnus } A, B \text{ sur } \mathbb{Z}_m.$$

D'où $\begin{cases} A \equiv (S_2 - S_3)/(S_1 - S_2) \pmod m. \\ B \equiv S_2 - S_1 (S_2 - S_3)/(S_1 - S_2) \pmod m. \end{cases}$

Si $\text{pgcd}(S_1 - S_2, m) \neq 1$, plusieurs solutions, et

O peut trouver A, B en utilisant quelques bits du ~~chiffré~~ clair ou par essais de cryptage $\square$

Conclusion : On vient de voir que l'utilisation de PRNG seul est faible pour la sécurité ; d'où on fait appel à CS-PRNG (i.e. Cryptographiquement secure PRNG) : si on a $b_1 \dots b_n$, il est difficile calculatoirement de calculer les bits $b_{n+1}, b_{n+2} \dots$

La question est donc :

Comment construire des cryptosystèmes à longueur variable pratique ? On a 3 propositions :

(1) S-cryptosystème optimaux par le software :

peu d'instructions CPU par calculer un bit d'une S-clé

(2) S-cryptosystème facile par le hardware :

Un exemple connu suffisamment est l'utilisation de LSFR (Linear-Shift Feed-Back Register)

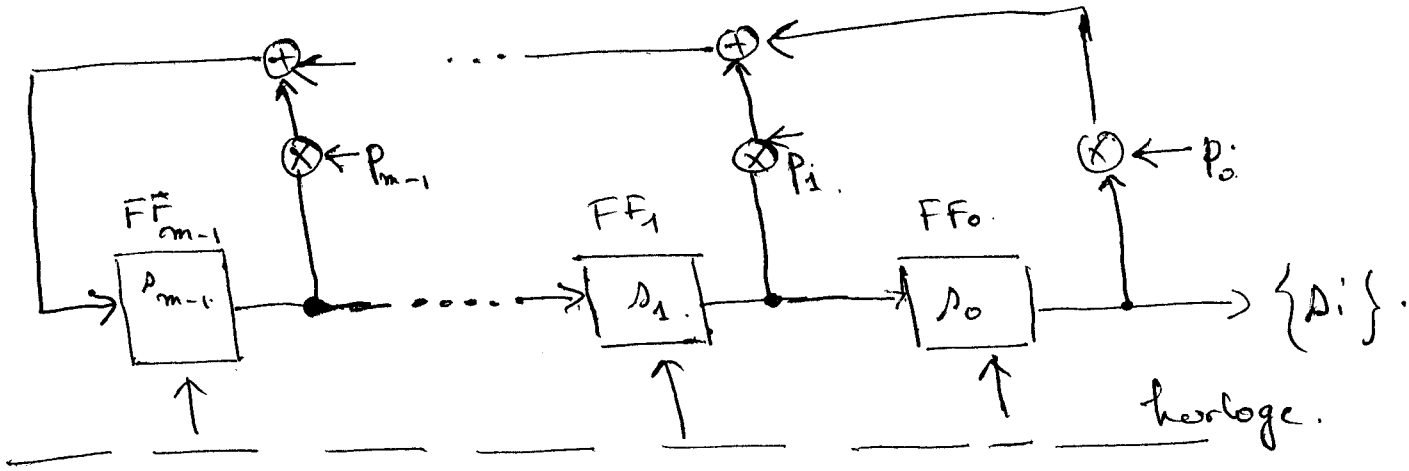
(3) S-cryptosystème basé sur le B-cryptage.

(par ex : CFB, OFM, ... voir 5 modes opératoires, après avoir étudié le DES ou le AES).

(5). S-cryptage basé sur les Registres à décalage.

- La s-clé $s_1, s_2, \ldots$ est engendré par un LSFR; facile à implémenter hardware, servent en GSM parce. On peut les rendre résistants aux attaques. On verra les bases ici.

- (5.1) LSFR. La forme d'un LSFR de degré m est:



avec les $p_i \in \{0, 1\}$; $p_i = 1$. (resp 0) si feed-back actif (resp. non actif).

FF_i : flip-flop. contenant la valeur initial s_i $(i = 0, \ldots, m-1)$

ainsi $s_m = s_{m-1} p_{m-1} + \cdots + s_0 p_0 \mod 2$.

$s_{m+1} = s_m p_{m-1} + \cdots + s_1 p_0$, etc.

En général la partie s_{i+m} est:

$$s_{i+m} = \sum_{j=0}^{m-1} p_j s_{i+j}$$

Théorème. La période maximale d'un LSFR est $2^m - 1$.

Preuve. Compter le nombre d'états du LSFR. ⊕

• (5.2). Attaque avec clair / chiffré.

supposons donc que la clé $k = (p_{m-1}, \dots, p_0)$ servant de base pour le calcul de la s-clé $(\Delta_i)_i$.

Hypothèse : { clair $x_0, \dots, x_{2m-1}$ connu par Oscar. ainsi que son chiffré $y_1, \dots, y_{2m-1}$.

Oscar calcule $\Delta_i = x_i + y_i \pmod 2$ $i = 0, 1, \dots, 2m-1$.

on a : $(*)$ $\Delta_{i+m} = \sum_{j=0}^{m-1} p_j \Delta_{i+j} \pmod 2$ $i \geq 0$.

L'eq. $(*)$ peut servir pour déduire les (p_i) :

$$(S)\begin{cases} i=0, & \Delta_m = p_{m-1}\Delta_{m-1} + \dots + p_0 p_0 \pmod 2. \\ i=1, & \Delta_{m+1} = p_{m-1}\Delta_m + \dots + p_0 \Delta_1 \\ \dots & \\ i = m-1, & \Delta_{2m-1} = p_{m-1}\Delta_{2m-2} + \dots + p_0 \Delta_{m-1} \end{cases}$$

Donc Oscar résout (S) pour trouver (p_i).

Remarque. Malgré l'attaque ci-dessus, on peut renforcer le système (ex. Trivium vir ...).

§2. D.E.S (Data Encryptin standard).

Le DES est l'un des cryptosystèmes les plus utilisés pour le B-cryptage (par blocs), malgré son remplacement par l'AES. Son étude permet de comprendre la conceptin des B-cryptosystèmes et les principes sous-jacents (ex. confusin et diffusin de Shannon). Il existe des améliorations tel le 3.DES.

C • Confusin : opératin cryptographique où la relatin entre la clé et le chiffré est obscure.

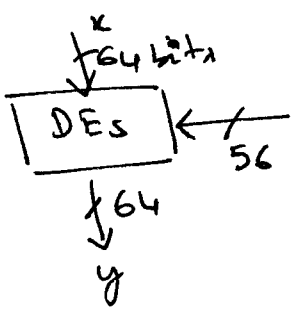
D • Diffusin : opératin cryptographique où l'influence d'un symbole du clair est étalé sur plusieurs symboles du chiffré (dans le but de cacher les propriétés statistiques du clair).

Pour renforcer cette idée, on concatène des opérations primitives en utilisant un chiffrement produit :

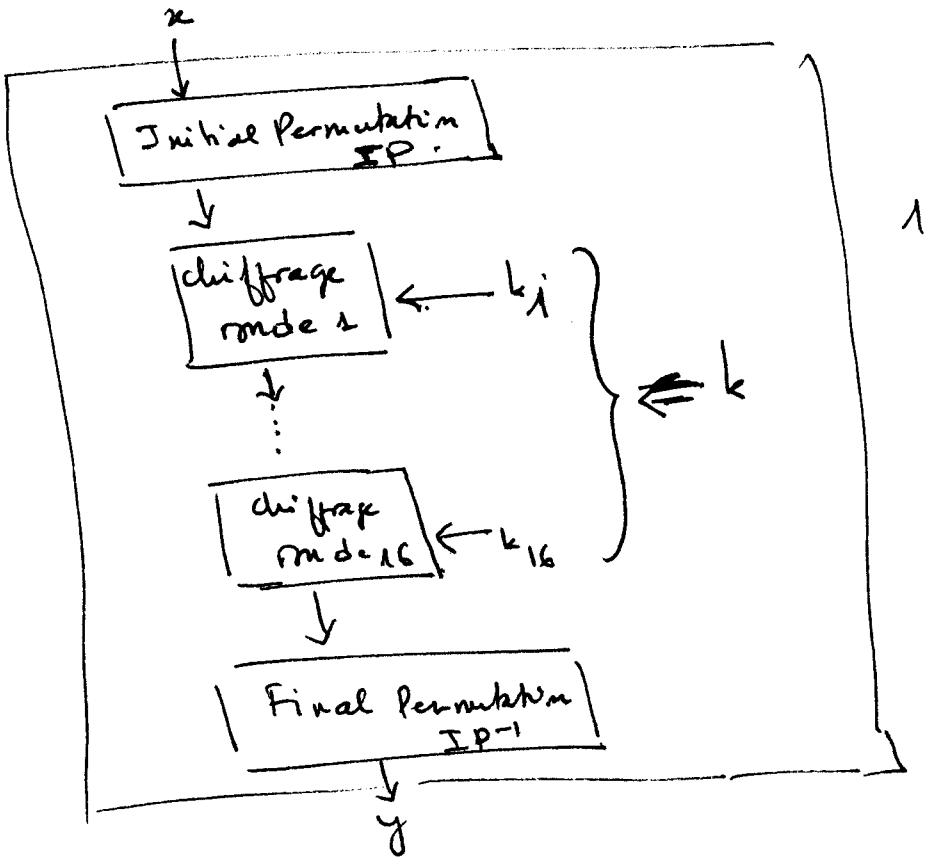
$$x \text{ clair} \longrightarrow \boxed{(P_1, C_1) \xrightarrow[y_1]{} (P_2, C_2) \longrightarrow \cdots \longrightarrow (P_N, C_N)} \rightarrow y_N \text{ chiffré}$$

qui est la structure du DES (et aussi presque de l'AES).

(1) L'Algorithme DES : chiffrement par bloc de taille 64 bits :



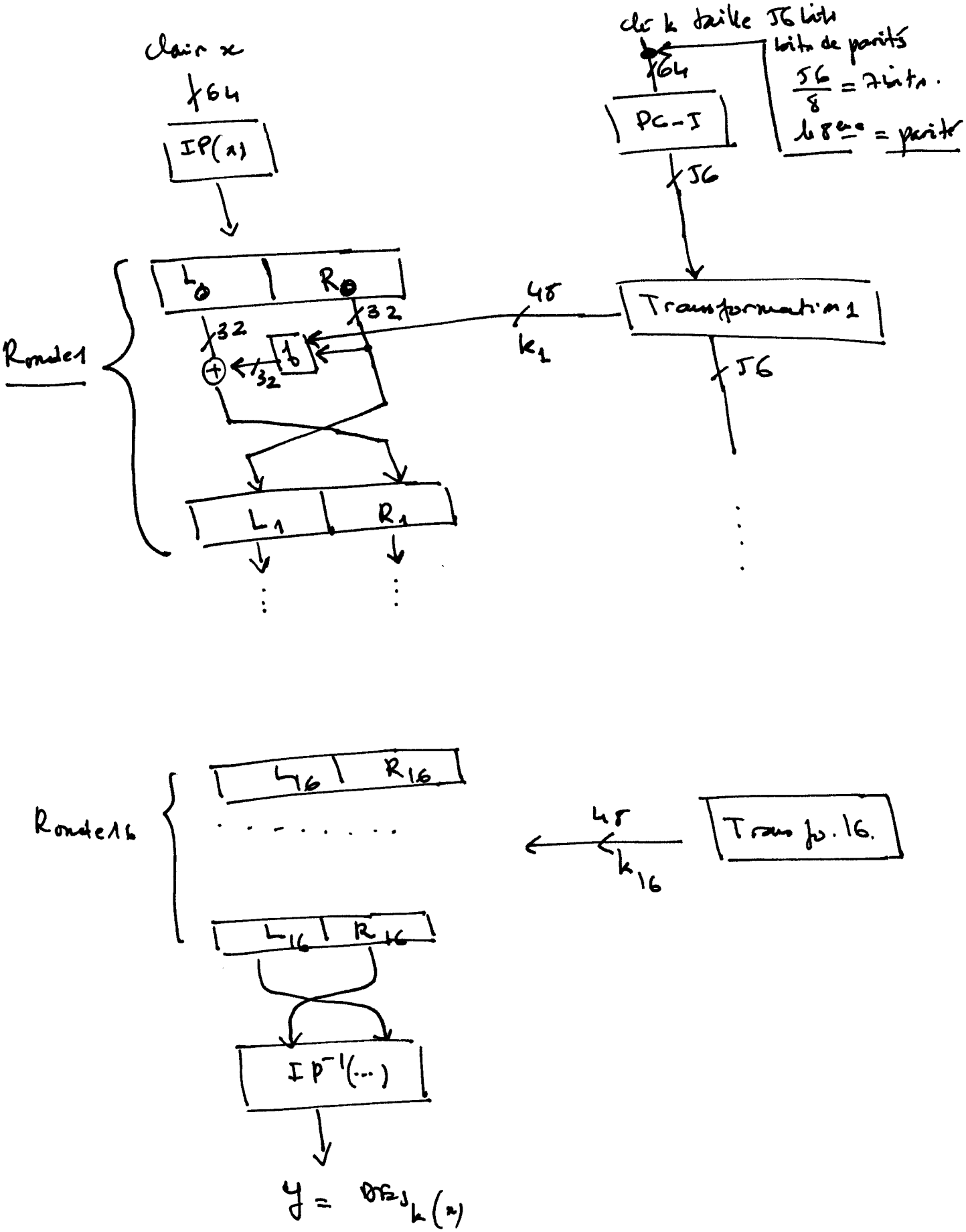
Le B. chiffrage est fait itérativement : chaque étape (qu'on va appeler ronde) exécute les mêmes opérations ; il y a 16 rondes. chaque ronde a une clé k_i extraite de k.



16 rondes du DES.

La structure des rondes (identiques) est basée sur le réseau de Fiestel facile à implémenter et aussi la méthode de décryptage consiste à ordonnancer la clé k_i en ordre inverse

Le réseau de Fiestel consiste en 16 rondes avec l'entrée (L_0, R_0)


clair x
64
IP(x)
clé k taille 56 bits
bits de parité
64
56/8 = 7 bits.
1 bit = parité
PC-1
56
L_0
R_0
32
32
48
k_1
Transformation 1
Ronde 1
56
L_1
R_1
L_{16}
R_{16}
Ronde 16
48
k_{16}
Transfo. 16.
$IP^{-1}(\cdots)$
$y = DES_k(x)$

→ Au début, on applique $IP(x)$ une permutation ;
le résultat obtenu est partagé en L_0, R_0.

on applique ensuite

ronde i $\begin{cases} L_i = R_{i-1} & \text{(copie)} \\ R_i = L_{i-1} \oplus f(R_{i-1}, k_i) \end{cases}$

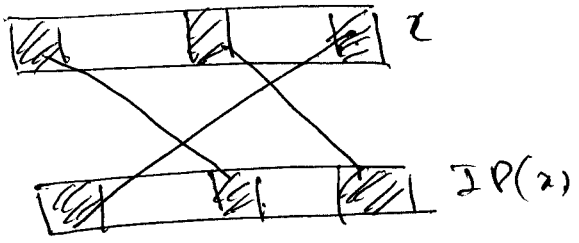
noter que la partie gauche seulement est cryptée ici par f, au rond prochain, l'autre partie est cryptée (swapping).

⇒ la substance du chiffrement est donc dans la fonction f, qui réalise la Confusion et la diffusion. C'est une fonction non-linéaire, utilisant la clé k_i, qui est surjective.

(2) Structure interne du DES :

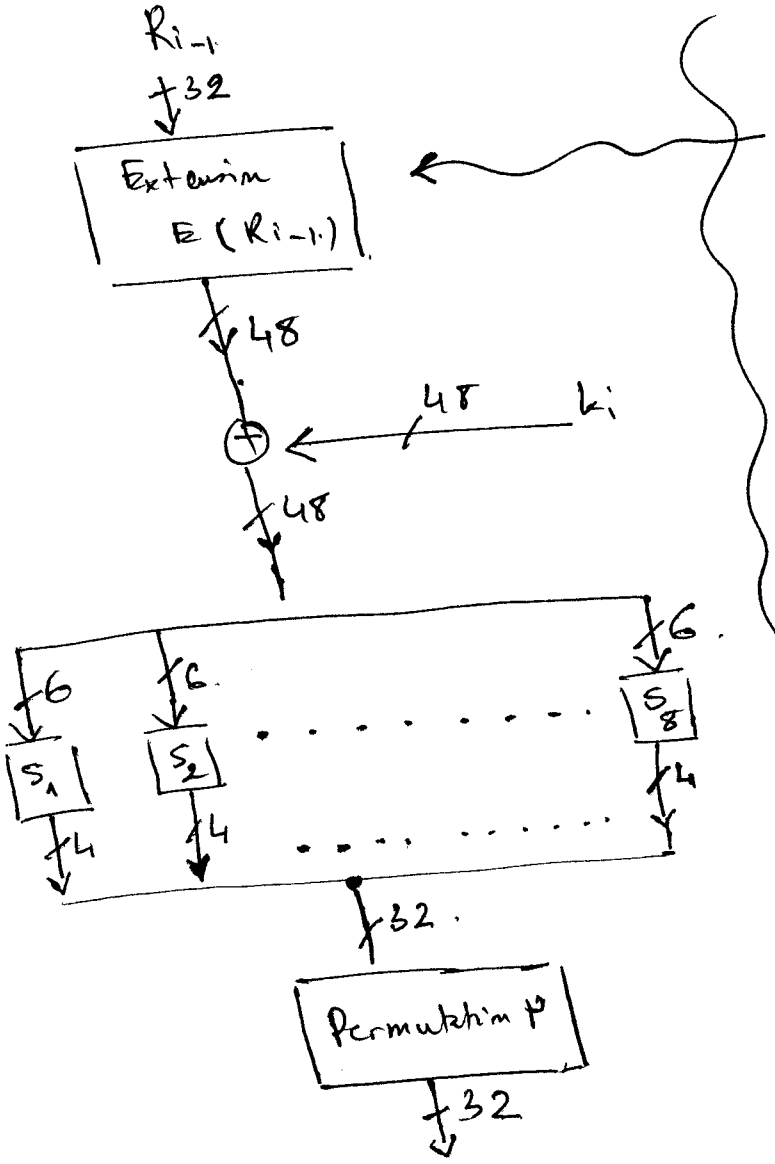
Les blocs constituants le DES sont :
PI et PI^{-1}, les rondes du DES, la fonction f et l'ordonnancement des clés.

(2.1) PI et PI^{-1} : permutation (fixe tables) permettant d'arranger les choses. Facile à réaliser hardware.



(2.2) La fonction f.

Rappelons que $R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$ au rond i.



Désorganise en bloc de 4 bits en répétition de certains bits : renforce l'idée de la diffusion. On obtient $6 \times 8 = 48$ bits

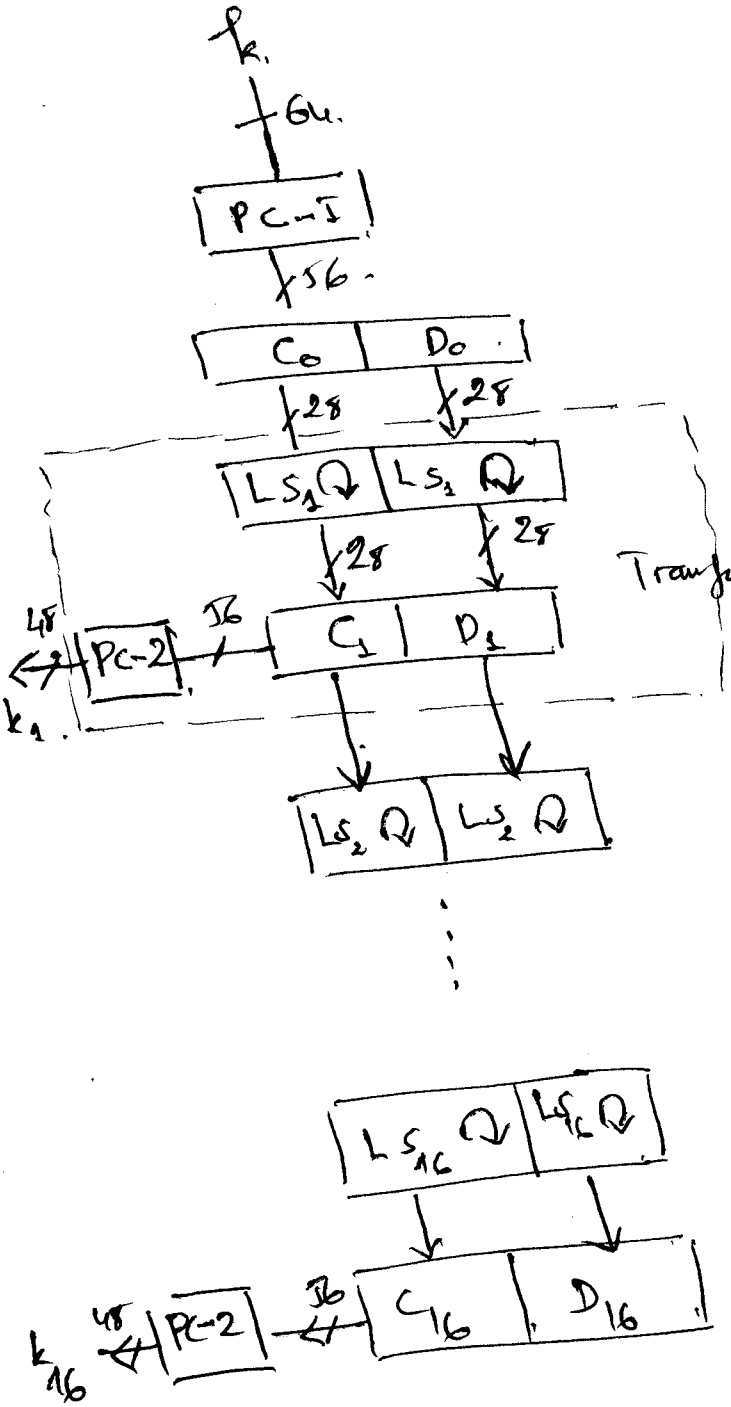
Les S_i : S-box reçoit 6 bits et renvoit 4 bits : les S_i sont distincts et sont codés dans des tables. Augmente la confusion (par leur non-linéarité) i.e. $S_i(a \oplus b) \neq S_i(a) + S_i(b)$

- Les S-box augmentant la sécurité du DES, permettant aussi de parer à l'attaque par cryptanalyse différentielle introduite en 1990 par Shamir et Biham (connu par les chercheurs d'IBM 16 ans par avance !!!).

P renforce la diffusion.

(2.3) ordonnancement clé

- Les 56 bits (actuels) sans bits de parité sont permutés par PC-1, ensuite partagé en deux blocs de 28 bits chacun C_0 et D_0.



- LS_i : left shift by i positions.

rnde i : LS_1 pour $i = 1, 2, 9, 16$.

i : LS_2 $i \neq 1, 9, 2, 16$

$\Rightarrow C_0 = C_{16}$ et $D_0 = D_{16}$.

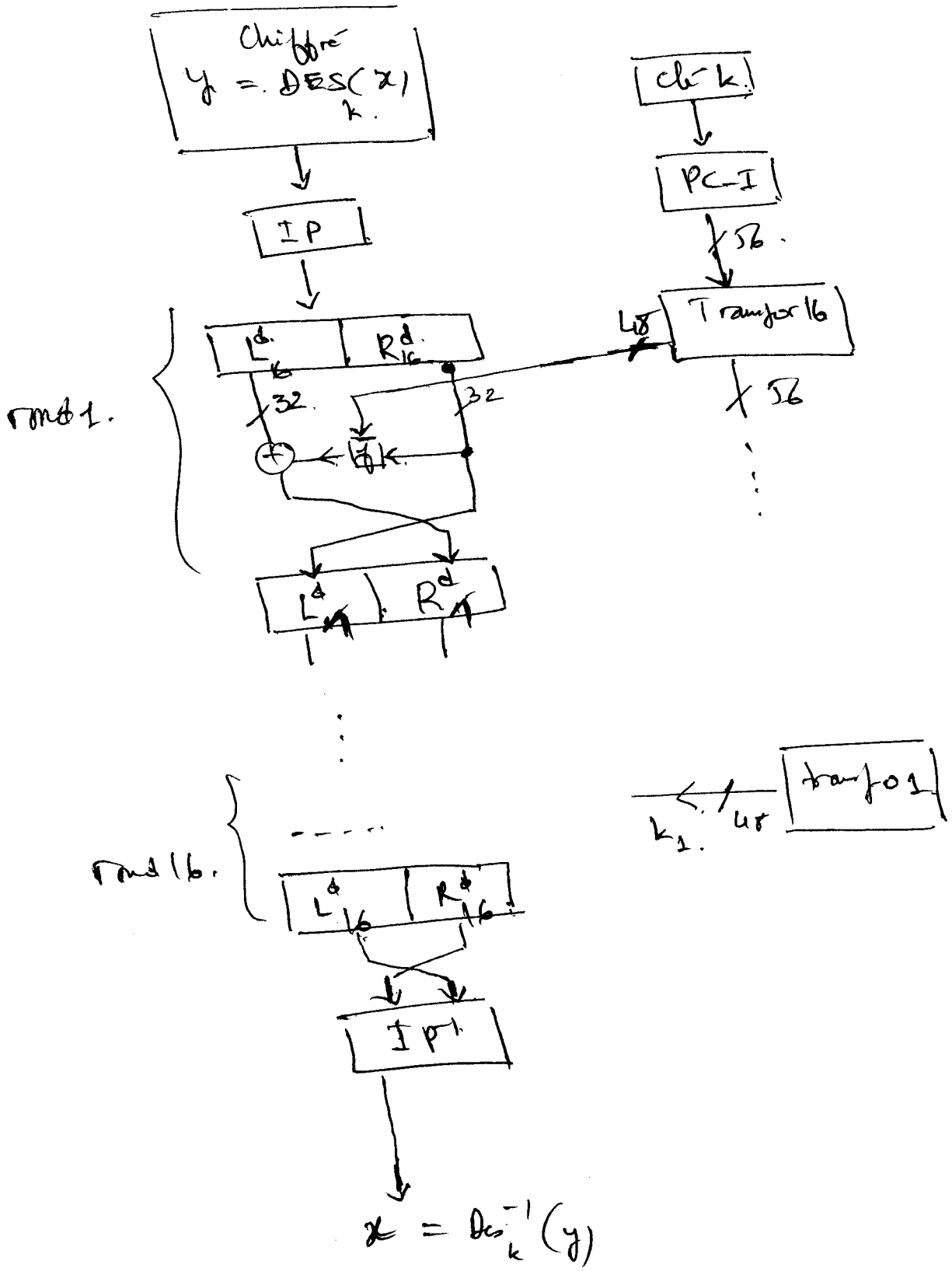
- PC_2 permute les 56 bits et ignore 8 bits $\Rightarrow$ 48 bits.

- Ces opérations sont faciles à réaliser en hardware.

(2.4) Décryptage :

Le décryptage se fait essentiellement comme le cryptage, basé sur le réseau de Fiestel, sauf que l'ordonnancement de clés est inversé : round 1, clé k_{16}.

2, clé k_{15}, etc...



Justification :

- Etant donné k, trouver k_{16} :

on a $C_0 = C_{16}$ et $D_0 = D_{16}$. D'où k_{16} après PC-1 :

$$\begin{aligned} k_{16} &= PC - 2(C_{16}, D_{16}) \\ &= PC - 2(C_0, D_0) \\ &= PC - 2(PC-1(k)). \end{aligned}$$

- $$\begin{aligned} k_{15} &= PC - 2(C_{15}, D_{15}) \\ &= PC - 2(RS_2(C_{16}), RS_2(D_{16})) \end{aligned}$$

(RS$_2$: Right shift.)

$$= PC - 2(RS_2(C_0), RS_2(D_0)).$$

$k_{14}, k_{13}, \ldots$ s'obtiennent par RS

- on a : Ronde$_i$ décryptage inverse Round$_{16-i+1}$ du crypt.

- $(L_0^d, R_0^d) = IP(Y) = IP(IP^{-1}(R_{16}, L_{16}))$
$= (R_{16}, L_{16})$

$\Rightarrow \quad L_0^d = R_{16}$ et $R_0^d = L_{16} = R_{15}$.

- Ronde$_1^d$ inverse Ronde$_{16}$:

$L_1^d = R_0^d = L_{16} = R_{15}$.

$R_1^d = L_0^d \oplus f(R_0^d, k_{16}) = R_{16} \oplus f(L_{16}, k_{16})$

$= L_{15} \oplus f(R_{15}, k_{16}) \oplus f(R_{15}, k_{16}) = L_{15}$

etc.

$$\boxed{L_i^d = R_{16-i} \text{ et } R_i^d = L_{16-i}}$$

(2.5) Conclusion : sécurité du DES.

- Attaque Exhaustive :

La faiblesse du DES est son petit espace de clés réduite à 56 b

Principe Attaque Exhaustive

Entrée : Au moins deux paires (clair, chiffré) = (x, y).

Sortie : k, telle que $y = DES_k(x)$

Attaque : tester tous les 2^{56} clés possibles jusqu'à :

$$DES^{-1}_{k_i}(y) = x, \quad i = 0, 1, \ldots, 2^{56}-1.$$

1998, Deep-Crack en 56 heures.
($ 250,000). Société Electronic Frontier Foundation.

- Attaque Analytique :

{ cryptanalyse analytique // linéaire } : les S-box résistent.

Départ # paires (clair, chiffré) $\simeq 2^{47}$.

- Amélioration 3-DES.

§ AES : **Projets d'Echanges**

§4. Modes opératoires : Mise en œuvre d'un cryptosystème.

Le B-chiffrement, tel le DES, AES, 3-DES est un algorithme de cryptage de blocs de petite taille 8 byte ou 16 byte; alors qu'on cherche à chiffrer des e-mails, des fichiers de grande taille. On verra ici comment utiliser le B-cryptage pour cette tâche, et aussi des modes d'authentification (voir par ex. MAC.). On peut utiliser aussi le B-chiffrement pour le hachage (voir chap. privé), même si la cryptographie asymétrique est plus adaptée. On peut aussi exploiter le B-chiffrement pour construire des S-chiffrem

(4.1) Chiffrement avec le B-cryptage & modes opératoires.

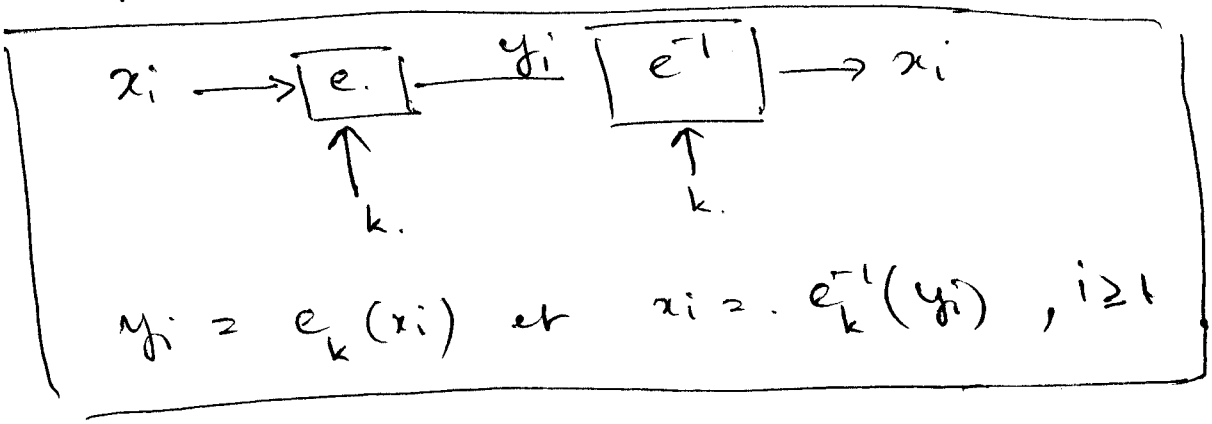
(4.1.1) Electronic Codebook Mode (ECB.)

Notons $e_k(x_i)$ le B-chiffrement du clair x_i par la clé k

$e_k^{-1}(y_i)$ l'opération de décryptage

des blocs de taille b. (Si le texte dépasse b, on le décompe en blocs de taille b, et on 'padde' si la taille $< b$).

ECB



$y_i = e_k(x_i)$ et $x_i = e_k^{-1}(y_i)$, $i \geq 1$

- Avantages : facile à implémenter et parallélisable
- Inconvénients : clé fixée ; faible devant attaques par substitution ⇒ Intégrité détruite

(4.1.2) Mode CBC (Chiffer Block chaining Mode).

- Chainage des blocs tq y_i dépend aussi des clairs précédents ($x_{i-1}, x_{i-2}, \cdots$).
- Rendu aléatoire en utilisant un vecteur d'initialisation (IV).

Définition CBC

$e(\cdot)$ cryptage bloc de taille b ; x_i, y_i de taille b ;

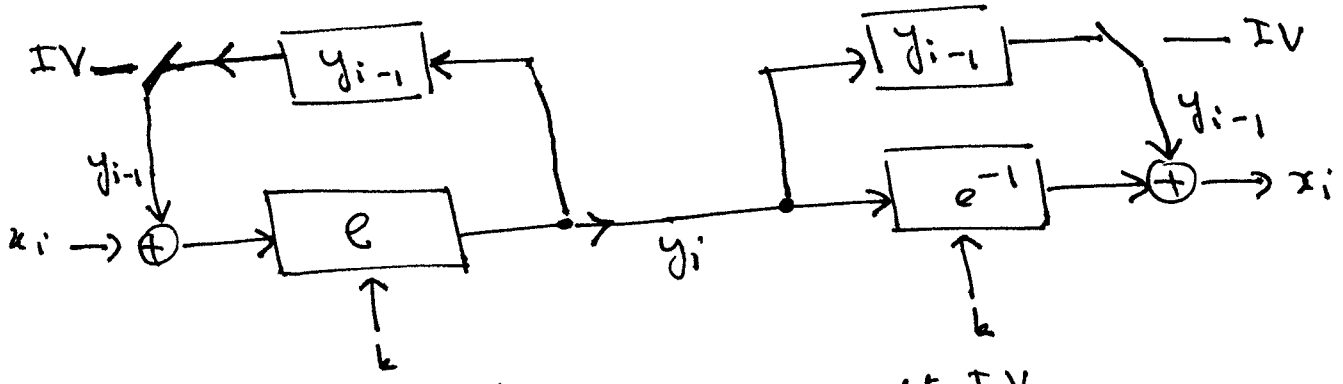
IV nonce (number used once) de long. b.

Cryp, 1er bloc : $y_1 = e_k(x_1 \oplus IV)$

Cry, En général : $y_i = e_k(x_i \oplus y_{i-1})$ $i \geq 2$.

Dec, 1er bloc : $x_1 = e_k^{-1}(y_1) \oplus IV$

Dec, En général : $x_i = e_k^{-1}(y_i) \oplus y_{i-1}$, $i \geq 2$.



⇒ y_i dépend de $x_1, \cdots, x_{i-1}$ et IV

(4.1.3) Mode OFB (Output Feedback Mode)

permet de construire des S-chiffrement.

Def OFB

e( ) B-cryptage de taille b ; x_i, y_i, s_i de long. b ;

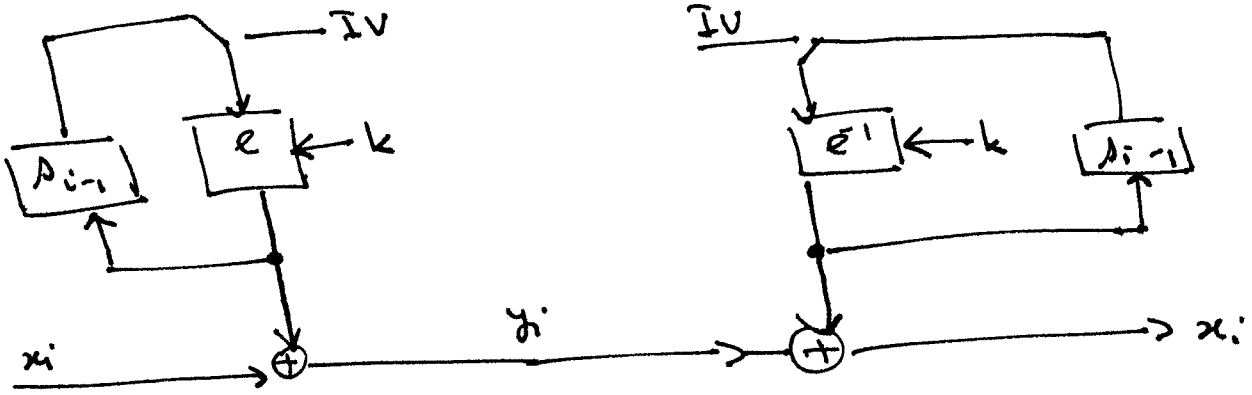
IV nonce de taille b.

Cryptage (1er Bloc) : $s_1 = e_k(IV)$, $y_1 = s_1 \oplus x_1$

" (général) : $s_i = e_k(s_{i-1})$, $y_i = s_i \oplus x_i$, $i \geq 2$

Decryptage (1er bloc) : $s_1 = e_k(IV)$ $x_1 = s_1 \oplus y_1$

Decryptage (général) : $s_i = e_k(s_{i-1})$, $x_i = s_i \oplus y_i$, $i \geq 2$.



(4.1.4) Mode CFB (Cipher Feedback Mode).

ressemble OFB.

Def · C 1er bloc : $y_1 = e_k(IV) \oplus x_1$

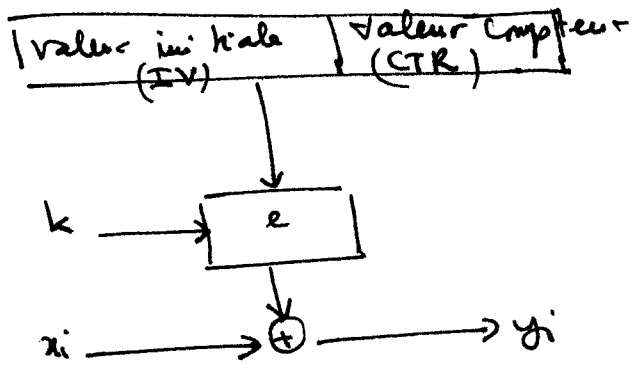
C En général : $y_i = e_k(y_{i-1}) \oplus x_i$, $i \geq 2$

D 1er bloc : $x_1 = e_k(IV) \oplus y_1$

D Général : $x_i = e_k(y_{i-1}) \oplus y_i$, $i \geq 2$

IV est un nonce qui peut être modifié.

(4.1.5) CTR mode (Counter Mode)



Def. CTR mode.

$(IV \| CTR_i)$ Concaténation de IV et CTR_i de longueur b.

Cryptage : $y_i = e_k(IV \| CTR_i) \oplus x_i \quad i \geq 1$

Décryptage $x_i = e_k(IV \| CTR_i) \oplus y_i \quad , i \geq 1.$

- Alice et Bob peuvent s'échanger la valeur initiale $(IV \| CTR_1)$ publique et à chaque chiffrement le Compteur est incrémenté (qui peut être un LSFR de taille maximum).

(4.1.6) GCM (Galois Counter Mode)

→ Il permet de calculer aussi un MAC (Message Authentification Code) en plus de la fonction de cryptage avec le mode CTR.

→ Alice calcule un MAC rajouté à la fin de son message crypté. Bob calcule son 'MAC' et le compare à celui d'Alice : si égalité authentique.

En plus, le message est intègre (personne ne l'a manipulé en cours de la transmission).

Définition. GCM (Galois Counter Mode).

Soient $e(.)$ un B-cryptage de taille 128 bits;

x le clair $= (x_1, \dots, x_n)$

AAD = additional authenticated data.
(ex. nom, fonction, mail, etc)

1. **Cryptage**
   a. Déduire une valeur CTR_0 de IV et calculer
   $$CTR_1 = CTR_0 + 1.$$
   b. Calculer les chiffrés $y_i = e_k(CTR_i) \oplus x_i$, $i \geq 1$.

2. **Authentification**
   a. Générer une sous-clé d'authentification $H = e_k(0)$.
   b. $g_0 = AAD \times H$
   c. $g_i = (g_{i-1} \oplus y_i) \times H \quad (1 \leq i \leq n)$
   d. Calculer le ~~tag~~ $T = (g_n \times H) \oplus e_k(CTR_0)$.

- Bob reçoit $[(y_1, \dots, y_n), T, ADD]$ et calcule $(x_1, \dots, x_n)$ avec la même procédure en mode CTR. Ensuite, avec l'algorithme GCM calcule T' en utilisant (y_i) et ADD. Si $T = T'$, alors Bob est sûr que x et ADD n'ont pas été modifié.
- Tous les calculs sont faits dans $GF(2^{128})$ avec le polynôme irréductible $p(x) = x^{128} + x^7 + x^2 + x + 1$.

(4.2) Question de sécurité encore

- (1) $Dec_{k_i}(x_1) \stackrel{?}{=} y_1$ où (x_1, y_1) paire clair/chiffré.

  $i = 1, \ldots, 2^{56} - 1$.

Trouver k_i.

il se peut que égalité dans (1), mais k_i est une fausse clé.

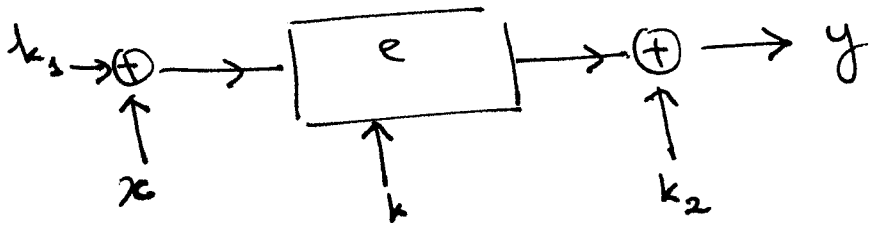
Théorème Soit un B-cryptage dont la taille de clé est x, de taille de cryptage n et soit $(x_1, y_1), \ldots, (x_t, y_t)$ paire clair/chiffré. La moyenne de fausse clés réalisant (1) est 2^{x-tn}.

- On augmente la sécurité soit en cryptant + sieurs fois! ou la technique 'key whitening' (KW):

- Déf (KW):

  cryptage : $y = e_{k,k_1,k_2}(x) = e_k(x \oplus k_1) \oplus k_2$

  décryptage : $x = e^{-1}_{k,k_1,k_2}(y) = e^{-1}_k(y + k_2) \oplus k_1$.



Chap. 3. Cryptographie Asymétrique.

§1. cryptographie à clé publique

✓ §2. RSA.

✓ §3. Protocole Diffie-Hellman (DF) et log discret (LD)

✓ §4. ELGAMAL

✓ §5. Courbes Elliptiques (Ecc)

§2. Cryptographie à clé publique

(1) La cryptographie symétrique comme DES, AES, s-cryptographie, utilise la même clé pour crypter et décrypter, et l'algorithme de décryptage ressemble à celui de cryptage.

Quelques inconvénients :

- La clé le doit être partagée entre Alice et Bob en utilisant un canal sûr
- nombre de clés (et leur stockage) entre n partenaires est $\frac{n(n-1)}{2}$ assez grand.
- Impossible de traiter la non-répudiation

(2) La cryptographie à clé publique apporte des réponses à ces questions et permet aussi de traiter les points suivants :

(A) cryptage : on peut chiffrer des messages en utilisant des cryptosystèmes tels RSA, ELGAMAL, ECC, ...

(B) Etablissement de clés : sur canal non sûr (insécurisé) par ex : protocole DH, transport clé RSA.

(C) Identification : en utilisant la signature et Protocole challenge - Réponse. (typ. smart card, mobile, ...)

(D) Non-Répudiation : à l'aide de la signature

Rq : En pratique les protocoles de sécurité sont hybrides

(3) Principe Cryptographie à clé publique.

- Supposons que A veut envoyer un message chiffré à B ; il n'est pas nécessaire que la clé de cryptage soit secrète, et il suffit que B possède une clé privé de decryptage.

⇒ B publiera une clé publique k_{pub} et gardera une clé privé k_{pr}

⇒ **la clé de B est $k = (k_{pub}, k_{pr})$.**

A		B
	$\xleftarrow{k_{pub}}$	$k = (k_{pub}, k_{pr})$
$e_{k_{pub}}(x) = y$	$\xrightarrow{y}$	$x = d_{k_{pr}}(y)$.

- Exemple de transport d'une clé de chiffrement symétrique par l'utilisation d'un cryptosystème à clé publique.

A		B
	$\xleftarrow{k_{pub}}$	k_{pub}, k_{pr}
Choisir une clé k_{DES}.		
$y = e_{k_{pub}}(k_{DES})$	$\xrightarrow{y}$	$k_{DES} = d_{k_{pr}}(y)$
Crypter message x par DES:		
$z = DES_{k_{DES}}(x)$	$\xrightarrow{z}$	$x = DES^{-1}_{k_{DES}}(z)$.

La sécurité est basée sur le concept :

Déf Une fonction f est à sens unique (one-way function) si

1. $y = f(x)$ est facile à calculer
2. $x = f^{-1}(y)$ est difficile à calculer.

Ex : RSA est basée sur le fait que si p, q sont entiers premiers, il est facile de calculer $y = p \cdot q$, mais il est difficile de retrouver la factorisation de y : problème de factorisation

Rq. 1) La notion de certificat relie la clé publique à B :

authentification

2) Un cryptosystème a un niveau de sécurité de n bits si le meilleur algorithme d'attaque exige 2^n étapes de calcul. (voir table livre Understanding cryptography de Paar et Pelzl, page 156)

§2. RSA.

(2.1) Chiffrement et déchiffrement

Un message clair x et son chiffré y appartiennent à $\mathbb{Z}_n$.

Algorithme RSA1.

Cryptage RSA. clé publique $k_{pub} = (n, e)$

$$y = e_{k_{pub}}(x) = x^e \bmod n.$$

où $x, y \in \mathbb{Z}_n$.

Décryptage RSA. clé privée $k_{pr} = d$

$$x = d_{k_{pr}}(y) = y^d \bmod n.$$

Génération de clés pour RSA. **Algorithme RSA2**

sortie : clé publique $k_{pub} = (n, e)$ et une clé privée $k_{pr} = (d)$

1. Choisir deux entiers premiers assez grand p et q
2. $n = p \cdot q$
3. $\phi(n) = (p-1)(q-1)$
4. Choisir l'exposant public $e \in \{1, 2, \dots, \phi(n)-1\}$ tq $\operatorname{pgcd}(e, \phi(n)) = 1$
5. calculer la clé privée d telle que $d \cdot e \equiv 1 \bmod \phi(n)$.

Ainsi B publie $k_{pub} = (n, e)$ et privatise $d = k_{pr}$.

Proposition (Correction de RSA).

$$d_{k_{pr}}\left(e_{k_{pub}}(x)\right) = x.$$

Preuve. On a

$$d_{k_{pr}}\left(e_{k_{pub}}(x)\right) = \left(x^e\right)^d \equiv x^{ed} \mod n \qquad (1)$$

et $d \cdot e \equiv 1 \mod \phi(n) \iff d \cdot e = t \cdot \phi(n) + 1$ pour un certain $t \in \mathbb{N}$.

D'où $d_{k_{pr}}\left(e_{k_{pub}}(x)\right) = x^{de} \equiv x^{1+t\cdot\phi(n)} = x \cdot \left(x^{\phi(n)}\right)^t \mod n$ (2)

- **Théorème d'Euler** : si $\text{pgcd}(n, x) = 1$, alors $x^{\phi(n)} \equiv 1 \mod n$

(et donc $\forall$ entier l, $\left(x^{\phi(n)}\right)^l \equiv 1 \mod n$).

Donc on distingue deux cas :

→ $\text{pgcd}(x, n) = 1$, $d_{k_{pr}}(y) \overset{(2)}{\equiv} \left(x^{\phi(n)}\right)^t \cdot x$

$\equiv 1 \cdot x \equiv x \mod n.$

→ $\text{pgcd}(x = pq, x) \neq 1$, donc x est de la forme

$x = r \cdot p$ ou $x = s \cdot q$ avec $r < q$ et $s < p$.

supposons que $x = rp \Rightarrow \text{pgcd}(x, q) = 1$.

(2) : $\left(x^{\phi(n)}\right)^t = \left(x^{(q-1)(p-1)}\right)^t = \left(\left(x^{\phi(q)}\right)^t\right)^{p-1}$

$\equiv 1^{(p-1)}$ (Euler : $x^{\phi(q)} \equiv 1 \mod q$)

$\equiv 1 \mod q$

Donc $\left(x^{\phi(n)}\right)^t = 1 + u \cdot q$

D'où $$\left(x^{\varphi(n)}\right)^t \cdot x = x + x u q$$
$$= x + r \cdot p \cdot u \cdot q$$
$$= x + r \cdot n \cdot q$$
$$\equiv x \quad (\mathrm{mod}\ n). \quad \square$$

Remarques importantes. Jusqu'à maintenant tout est beau ! Mais pour des raisons d'efficacité et de sécurité, on note

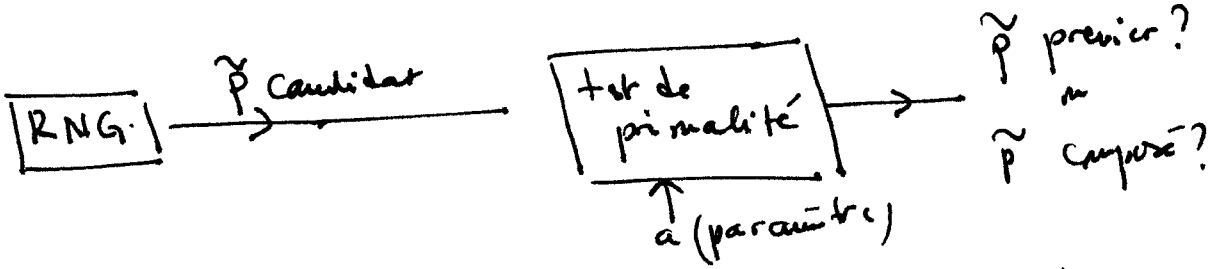
A) Dans l'algorithme RSA1, il faut utiliser un algorithme d'exponentiation modulaire assez rapide (basé sur la décomposition binaire d'un nombre : square and multiply.

Dans RSA2, étape 4, utiliser l'algorithme d'Euclide étendu pour trouver la clé d (voir Rappels).

B) Engendrer des nombres premiers p et q suffisamment grand et équilibré. Et donc n est ramené à des tests de primalité

(2.2) Recherche de nombres premiers assez grands

si le module $n = p \cdot q$ est de taille $|n|_2 = \lceil \log_2 n \rceil \simeq 1024$,
$|p| \simeq |q| = 512$ bits. Comment engendrer les nombres premiers ?



Questions :
- Q1) Combien d'entiers aléatoires qu'on doit engendrer pour dire que $\tilde{p}$ est premier ?
- Q2) Rapidité du test de primalité ?

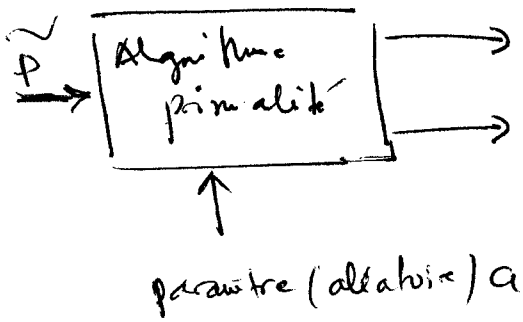
(Q1) $\mathbb{P}[\tilde{p} \text{ est premier}] \simeq \frac{2}{\ln(\tilde{p})}$.

Ex: RSA, module $n = 1024$ bits. $|p| = |q| \simeq \frac{1024}{2} = 512$.

$$\mathbb{P}(\tilde{p} \text{ premier}) \simeq \frac{2}{\ln(2^{512})} = \frac{2}{512 \cdot \ln 2} \cong \frac{1}{177}.$$

$\Rightarrow$ on teste 177 fois $\tilde{a}$ pour $\tilde{p}$ ☒

(Q2) Test de primalité est plus facile que la factorisation.



- Test de Fermat (basé sur le théorème de Fermat).

Entrée : $\tilde{p}$ candidat et λ (paramètre sécurité).

Sortie : $\tilde{p}$ composé ou $\tilde{p}$ est probablement premier.

1. Pour $i = 1$ à λ faire.
   - 1.1. choisir $a \in \{2, 3, \dots, \tilde{p} - 2\}$.
   - 1.2. Si $a^{\tilde{p}-1} \neq 1$.
   - 1.3. alors retourner ($\tilde{p}$ est composé).
2. Retourner ($\tilde{p}$ est premier avec une grande prob.).

(Théo. Fermat : si p premier, $a^{p-1} \equiv 1 \bmod p$).

- **Test de Miller-Rabin :**

Théorème. Soit la décomposition $\tilde{p} - 1 = 2^u \cdot r$ où r est impair. S'il $\exists\, a \in \mathbb{N}$,

$$a^r \not\equiv 1 \bmod \tilde{p} \quad \text{et} \quad a^{r \cdot 2^j} \not\equiv \tilde{p} - 1 \bmod \tilde{p}$$

$\forall j = \{0, \dots, u-1\}$, alors $\tilde{p}$ est composé. Sinon $\tilde{p}$ premier avec grde probabilité.

Exercice. Écrire un algorithme de test de primalité basé sur le théorème Miller-Rabin.

(2.3). Attaques de RSA.

3 types d'attaques exploitant comment RSA est implémenté ou utilisé, au lieu de l'attaque de l'algorithme lui même.

→ • Attaques par protocoles : Exploite la malléabilité de RSA.

Déf. Un cryptosystème est malléable si l'attaquant O. est capable de transformer le chiffré en un autre chiffré qui même a retrouve le clair.

O.
A ⟶ B
y.

•• Par Ex, RSA : soit y le chiffré de RSA où

O remplace y par $s^e \cdot y$ où $s \in \mathbb{N}$.

B déchiffre $s^e \cdot y$ par.

$$(s^e \cdot y)^d \equiv s^{ed} \cdot x^{ed} \equiv s \cdot x \mod n.$$

qui est un clair valide : donc O a détruit le message et B ne le sait pas.

Une telle attaque peut être évitée en utilisant le padding (voir Paul. page 192 – 193.) qui consiste à rajouter une information aléatoire au clair dans le message avant la transmission.

→ • Attaques mathématiques

O connaît le module n, $k_{pub} = e$. et y chiffre.

Son but est de trouver $k_{pr} = d$ tq $e \cdot d \equiv 1 \bmod \phi(n)$,

O peut appliquer l'algo d'Euclide Etendu pour trouver d,

mais il ne connaît pas $\phi(n)$.

Si par contre il factorise $\phi(n) = (p-1)(q-1)$, il peut

écrire $\begin{cases} d^{-1} \equiv e \bmod \phi(n) \\ x \equiv y^d \bmod n. \end{cases}$

⇒ revient à factoriser n. qui doit être assez grand.

($n \geq 1024$.)

Vu la rapidité des attaques., il est recommandé d'avoir

$n \geq 2048$.

→ • Attaques canneaux cachés.

Exploite la consommation d'énergie et les calculs faits

par l'algo RSA.

§3. Cryptographie à clé publique basée sur le logarithme discret (LD)

RSA est basé sur le problème de la factorisation d'entiers qui est une fonction à sens unique. Une autre fonction à sens unique est le logarithme discret. (PLD) :

> (PLD) en général :
> soit G un groupe cyclique fini de cardinal n.
> soit $\alpha \in G$ un élément primitif de G, et soit $\beta \in G$.
> le PLD est de trouver x, $1 \leq x \leq n$ tq :
> $\beta = \alpha^x$, noté $x = \log_\alpha \beta$.

Le Protocole Diffie-Hellman d'Echanges de clés, le cryptosystème ELGAMAL (ainsi que la signature numérique) sont basés sur le PLD, et on peut étendre à plusieurs structures algébrique-Géométriques. telle les courbes elliptiques.

(3.1) Échange de clés DH. (Construction de clés DH)

- **SET UP**

  1. Soit p un entier premier assez grand
  2. Choisir $\alpha \in \{2, 3, \dots, p-2\}$.
  3. Publier p et α.

- Alice (A) et Bob (B) peuvent construire une clé commune :

A.	(canal non sécurisé)	B.
✓ Choisir $a \in \{2, \dots, p-2\}$, $a = k_{prA}$ (clé privée de A)		✓ Choisir $b = k_{prB} \in \{2, \dots, p-2\}$.
✓ Calculer $A = k_{pubA} \equiv \alpha^a \bmod p$.		✓ Calculer $B = k_{pubB} \equiv \alpha^b \bmod p$
	$A \longrightarrow$	
	$\longleftarrow B$	
✓ $k = k_{AB} = k_{pubB}^{k_{prA}} = B^a \bmod p$.		$k = A^b \bmod p$.

Noter que les calculs se font dans $\mathbb{Z}_p^*$ et que

$$B^a \equiv (\alpha^b)^a = \alpha^{ab} \bmod p, \quad A^b \equiv \alpha^{ab} \bmod p.$$

Ainsi $k_{AB} = \alpha^{ab}$ est la clé commune.

Remarques. Pour l'infrastructure noter que le Protocole DH exige de trouver p premier (voir test de primalité, § RSA) et aussi on fait l'exponentiation modulaire rapidement. Aussi le groupe cyclique fini G est ici $\mathbb{Z}_p^*$ de cardinal $p-1$. ☒.

(3.2) Attaques du P.L.D.

(3.2.1) Algorithmes génériques.

Utilise seulement la loi de groupes et non pas la structure particulière du groupe.

A1 • Recherche exhaustive !

Dans G, on teste :

$$\alpha^1 \overset{?}{=} \beta,\ \alpha^2 \overset{?}{=} \beta,\ \dots,\ \alpha^x = \beta.$$

$\Rightarrow$ temps $O(|G|)$.

par exemple si $G = \mathbb{Z}_p^*$, $|G| = p-1$.

on prend $p \simeq 2^{80}$ pour éviter l'attaque.

A2 • Baby-step Giant step de Shanks..!

C'est un algorithme meilleur que A1 utilisant un compromis espace / temps. Supposons que $x = \log_\alpha \beta$.

On écrit $x = x_g m + x_b$. $0 \le x_g, x_b < m$.

où $m = \lceil \sqrt{|G|} \rceil$

$$\Rightarrow \quad \beta = \alpha^x = \alpha^{x_g m + x_b}$$

$$\Rightarrow \quad \beta \cdot (\alpha^{-m})^{x_g} = \alpha^{x_b} \qquad (1)$$

L'idée est de chercher une solution (x_g, x_b) de (1)

Déroulement : On calcule et on stocke α^{x_b}, $(0 \leq x_b < m)$

avec un temps $O(m) = O(\sqrt{|G|})$ temps

$O(\sqrt{|G|})$ valeurs stockées en mémoire.

(Etape bébé).

Etape. Ensuite, on calcule si (1) est vérifiée pour $0 \leq x_g < m$ pour chacune valeur x_g stockée.

$\Rightarrow$ { total $O(\sqrt{|G|})$ étapes calcul.
$O(\sqrt{|G|})$ espace.

Donc une attaque d'ordre 2^{80}, exige $|G| \geq 2^{160}$.

A3. <u>Méthode de Pollard</u> : $O(\sqrt{|G|})$ calcul,

basé sur le paradoxe des anniversaires :

On génère aléatoirement $\alpha^i \cdot \beta^j$ dans G. On continue jusqu'à collision :

$$\alpha^{i_1} \cdot \beta^{j_1} = \alpha^{i_2} \cdot \beta^{j_2} \qquad (2)$$

Si on prend $\beta = \alpha^x$, on remplace dans (2), on a

$$i_1 + j_1 x \equiv i_2 + x j_2 \Rightarrow x = \log_\alpha \beta \equiv \frac{i_2 - i_1}{j_1 - j_2} \pmod{|G|}$$

Dans ECC, le groupe est de taille $|G| > 2^{160}$.

A4. Algorithme de Pohlig-Hellman

Basé sur le théorème chinois, il exploite une factorisation possible du groupe G en conjonction avec l'un des algorithmes A_i $(i \leq 3)$.

Soit $|G| = p_1^{e_1} \cdots p_l^{e_l}$.

Au lieu de calculer $x = \log_\alpha \beta$, on calcule un L.D. plus petit $x_i \equiv x \bmod p_i^{e_i}$ dans un sous-groupe d'ordre $p_i^{e_i}$; ensuite on utilise le théorème chinois pour déduire une solution. Chaque P.L.D dans les sous-groupes est calculé par A_2 ou A_3.

(3.2.2.) Algorithmes non génériques : méthode d'index.

La méthode d'index calcule le L.D. dans certains groupes spécifiques comme $\mathbb{Z}_p^*$ et $GF(2^m)^*$ et donne lieu à des algorithmes sous-exponentiel.

(3.2.3) Sécurité du protocole d'Echange DH.

O. veut calculer k_{AB}, connaissant α, p et il peut obtenir A et B ($= k_{pub,A}$ et $k_{pub,B}$). Est-il capable de calculer $k = \alpha^{AB}$ (problème PDH).

PDH. Soit G groupe cyclique fini d'ordre n, $\alpha \in G$ primitive, et $A = \alpha^a$, $b = \alpha^b$ dans G.

Trouver α^{ab} ?

Supposons $G = \mathbb{Z}_p^*$, alors si $\mathcal{O}$ peut résoudre PLD alors il peut résoudre PDH :

car $\mathcal{O}$ calcule $a = \log_\alpha A = k_{pr.A} \pmod p$

et calcule $k_{AB} = B^a \bmod p$. ☒

D'où pour assurer la sécurité de PDH, on doit choisir p assez grand.

§4. EL GAMAL.

§§4.1. Cryptage et Décryptage

basé sur le PLD et le PDH. On verra ici la méthode sur $\mathbb{Z}_p^*$. Il provient du protocole DH d'échange de clé. Alice chiffre un message x en le multipliant par k_{AB}.

Cryptosystème de cryptage.

Alice.	Canal.	Bob.
		(a). choisir $d = k_{prB} \in \{2, \dots, p$
	$\leftarrow B$	(b) calculer $B = k_{pubB} = \alpha^d$ m
(c). choisir $i \in \{0, \dots, p-2\}$.		
(d) Calculer $k_E = k_{pub_A} = \alpha^i \bmod p$.	$\xrightarrow{k_E}$	(f) Calculer $k_M = k_{E_2}^d$.
(e) Calculer $k_M = B^i \bmod p$.		
(g) crypter $x \in \mathbb{Z}_p^*$: $y \equiv x \cdot k_M \bmod p$.	$\xrightarrow{y}$	
		(h) décrypter : $x = y \cdot k_M^{-1} \pmod p$

Protocole EL GAMAL (modifié)

Alice || Canal || **Bob**

Setup

Bob :
- choisir un premier p grand
- choisir $\alpha \in \mathbb{Z}_p^*$ primitif
- choisir $k_{pr} = d \in \{2, \dots, p-2\}$
- calculer $k_{pub} = \beta = \alpha^d \ (p)$

$k_{pub} = (p, \alpha, \beta)$

Cryptage

Alice :
- choisir $i \in \{2, \dots, p-2\}$ ← aléatoire
- calculer $k_E \equiv \alpha^i \ (p)$ clé provisoire
- calculer la clé masque : $k_M \equiv \beta^i \ \text{mod } p$
- crypter $x \in \mathbb{Z}_p^*$: $y \equiv x \cdot k_M \ (p)$

(k_E, y)

Décryptage

Bob :
- calculer le masque : $k_M = k_E^d \ (\text{mod } p)$
- décrypter : $x \equiv y \cdot k_M^{-1} \ \text{mod } p$

2ème version adaptée

On a : $d_{k_{pr}}(k_E, y) \equiv y \cdot k_M^{-1} \ (\text{mod } p)$.

$\equiv (x \cdot k_M) \cdot (k_E^d)^{-1} \ (p)$

$= (x \cdot \alpha^{di}) \cdot ((\alpha^i)^d)^{-1} \ (p)$

$= x \cdot \alpha^{di} \cdot \alpha^{-di} \ (p) \equiv x \ (\text{mod } p)$.

☒

Remarque. ELGAMAL est probabiliste par le cryptage. par le choix de $i \in \{2, \dots, p-2\}$. Ainsi même si

$$x_1 = x_2 \not\Rightarrow Cry(x_1) = Cry(x_2). \quad \square$$

§§4.2. Infrastructure et sécurité.

(A)

- **Génération de clés** : (voir page 3.17 §§.4.1). Un entier p doit être généré utilisant un vrai RNG. par ex. nombres (dans le § RSA) et qui doit être de taille au moins 1024 bits. En plus on a utilisé une exponentiation modulaire à calculer rapidement par square-and-multiply. (S and M). on a utilisé aussi un RNG.
- **Cryptage** : les opérandes ont une longueur $\lceil \log_2 p \rceil$: utiliser aussi S and M. algorithme.
- **Décryptage** ; exponentiation $k_M = k^d$ (S and M) et une inversion utilisant l'algo d'Euclide Etendu.

(B) Sécurité :

→ Attaques passives : retrouver x à partir de $(p, \alpha, \beta = \alpha^b, k_E = \alpha^i, y = x \cdot \beta^i)$ est protégée par le fait que le PDL est difficile.

Supposons que O. possède un moyen 'rapide' de calcul du LD :

⟹ retrouver x en trouvant $d_A = k_{prB}$:

$$d_A = \log_\alpha \beta \pmod p$$

ensuite,

$$x \equiv y_2 \cdot \left(k_E^{d_A}\right)^{-1} \bmod p.$$

⟹ Retrouver la clé aléatoire i d'Alice :

$$i = \log_\alpha k_E \pmod p$$

et $$x \equiv y_1 \cdot (\beta^i)^{-1} \bmod p$$

→ Attaques actives :

- L'authenticité de la clé de Bob doit être vérifiée ;

Sinon Oscar peut convaincre Alice que la nouvelle clé forgée par lui même est celle d'Alice : les certificats sont utilisés dans ce cas.

- La clé secrète (éphémère) d'Alice doit être renouvelée de préférence. En effet, supposons qu'Alice utilise i pour crypter x_1 et x_2. Alors $k_1 = \beta^i$ et le même et aussi k_E.

Alice $\xrightarrow[(y_1, k_E)]{\text{II}^\text{e}}$ Bob

Alice $\xrightarrow[(y_1, k_E)]{}$ Bob

Supp O connaît m_1, il peut crypter le 2ème message :

$$x_2 = y_2 \cdot k_1^{-1} \pmod p \text{ où } k_1 = y_1 \cdot x_1^{-1} \pmod p$$

Rq. On peut utiliser PRNG mais en changeant la base.

- Malléabilité :

O observe (k_E, y) et le remplace par :

$$(k_E, \lambda y) \quad \text{où} \quad \lambda \in \mathbb{N}.$$

B calcule :

$$d_{k_{prB}}(k_E, \lambda y) = \lambda \cdot y \cdot k_n^{-1}(p).$$

$$= \lambda \cdot (x \cdot k_n) \cdot k_n^{-1}(p)$$

$$= \lambda x \quad (\text{mod } p)$$

et le message de départ x est multiplié par λ.

($\Rightarrow$ padding).

Comme RSA.

§5. Cryptographie à base de Courbes elliptiques (ECC)

ECC (Elliptic Curve Cryptography) est basé sur le LD. dans le groupe-pour-jacent. Et donc les protocoles DH sont aussi valables. L'un des intérêt de ECC est qu'on a une meilleur sécurité avec des opérandes plus courts. (comparé à RSA). On verra ici le principe de fonctionnement (sans trop détailler les aspects mathématiques). Une large bibliographie existe. On verra aussi, une fois absorbé les Concepts de base, comment instaurer la cryptographie IBE (Identity Based Encryption) à la fin.

§§ 5.1. La loi de groupe sur une courbe elliptique.

• **Définition** C courbe elliptique sur $\mathbb{Z}_p$ $(p \geq 3)$ est l'ensemble des points $(x,y) \in \mathbb{Z}_p$ tq.

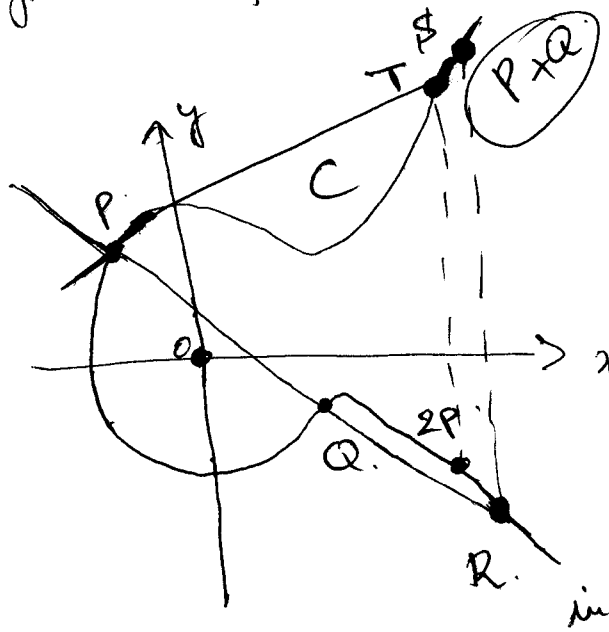
$$y^2 \equiv x^3 + ax + b \pmod p,$$

en plus d'un point O_∞ (à l'infini : jouant le rôle de l'élément neutre de la loi de groupe définie sur C).

où $a, b \in \mathbb{Z}_p$ avec le discriminant

$$\Delta = 4a^3 + 27 \cdot b^2 \not\equiv 0 \pmod p. \quad \square$$

Z **Remarque** ! (les dessins suivants sont dans $\mathbb{R}$ ou $\mathbb{C}$ pour faire manifester une courbe elliptique)



1) $P + Q = S$.
La droite (PQ) intersecte C en un pt. R dont on prend l'opposé S qui est P+Q.

2) La tangente au pt P intersecte C au pt T, dont on prend l'opposé qui est 2P.

Proposition. $(C, +)$ est un groupe commutatif avec l'élément neutre O_∞ et l'opposé de P est le pt $-P$ tq

$$P + (-P) = O_\infty.$$

Analytiquement. $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$

$P + Q = R$.

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \pmod p \\ y_3 = \lambda(x_1 - x_3) - y_1 \pmod p \end{cases}$$

où λ est la pente de la droite PQ ou la pente de la tangente au pt. P (si on calcule $P+P$).

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} \pmod p \text{ si } P \neq Q. \\ \dfrac{3x_1^2 + a}{2y_1} \pmod p \text{ si } P = Q. \end{cases}$$

§§5.2. PLD sur une courbe elliptique.

Théorème Sur une courbe elliptique C existe des sous-groupes cycliques.

Exemple Soit C: $y^2 \equiv x^3 + 2x + 2 \pmod{17}$

$(C, +)$ est un groupe cyclique ici, et $\#C = 19$ premier $\Rightarrow$ chaque pt de C est primitif. Soit $P = (5,1)$.

On calcule $\langle P \rangle = \{P, 2P, 3P, \dots, 19P\}$

$2P = (6,3)$ $\quad 3P = (10,6), \dots, 18P = (5,16)$,

$19P = \mathcal{O}_\infty$ $\quad 20P = P + 19P = P + \mathcal{O}_\infty = P$.

$18P + P = \mathcal{O}_\infty$. (l'inverse de P est $18P$). ☒

Théorème (Hasse) Une courbe elliptique C sur $\mathbb{Z}_p$ vérifie:

$$p + 1 - 2\sqrt{p} \leq \#C \leq p + 1 + 2\sqrt{p}$$

Déf PLD sur courbe elliptique.

Soit $\mathcal{E}$ une courbe elliptique sur $\mathbb{Z}_p$.
Soit P un élément primitif et $T \in C$.

Trouver $d \in \mathbb{N}$, $1 \leq d \leq \#C$ tq.

$$\underbrace{P + \dots + P}_{d \text{ fois}} \quad dP = T$$

Rq 1) d est la clé privée et T est publique.

2) On adapte l'algo (S and M.) ici :

Entrée C avec $P \in C$ et $d = \sum_{i=0}^{t} d_i 2^i$ binaire.

Sortie $T = dP$.

- $T = P$
- Pour $i =$ ~~1~~ jusqu'à d par -1 faire :
  - $T = T + T$ (mod p)
  - si $d_i = 1$, $T = T + P$ (mod p)
- Retourner (T).

§§ 5.3. Protocole DH d'échange de clé sur CE.

SETUP : 1. Soit p premier et $C : y^2 = x^3 + a.x + b$ (p).

2. Soit $\mathcal{P} = (x_p, y_p)$ primitif.

$(p, a, b, \mathcal{P})$ publique.

Protocole DH

Canal.

Alice # Bob.

- Choisir $k_{prA} = a \in \{2, 3, \dots, \#C - 1\}$
- Calculer $k_{pubA} = aP = A = (x_A, y_A)$

- Choisir $k_{prB} = b$.
- Calculer $k_{pubB} = B = bP = (x_B, y_B)$

A →

← B

- Calculer $aB = T_{AB} = (x_{AB}, y_{AB})$

- Calculer $bA = (x_B, y_B) = T_{AB}$.

$$T_{AB} = aB = a(bP) = (ba).P.$$

La clé commune T_{AB} peut être utilisée par exemple un chiffrement DES, AES, etc.

§§ 3.4. Sécurité.

Si O. veut attaquer le Protocole DH, il a l'information E (courbe elliptique), p, P, A, B. et veut calculer T_{AB}. Il doit résoudre Prob. LD sur CE. :

$$a = \log_P A \quad \text{ou} \quad b = \log_P B.$$

- Les attaques non génériques sont plus faibles.
- Reste les attaques génériques : Shank, Baby-Step, ...

temps $O(\sqrt{|G|})$, $|G| > 2^{160}$ 3.2.5

$\Rightarrow$ $|p|_2 \simeq 160$ bits.

D'où le choix de la courbe C est important

Chap 4. Signatures et hachages

§ Signature numérique.

§ Fonction de hachage.

§ MAC.

Chap. 4. Signature Numérique et Fonctions de Hachage.

En pratique, on signe un haché !

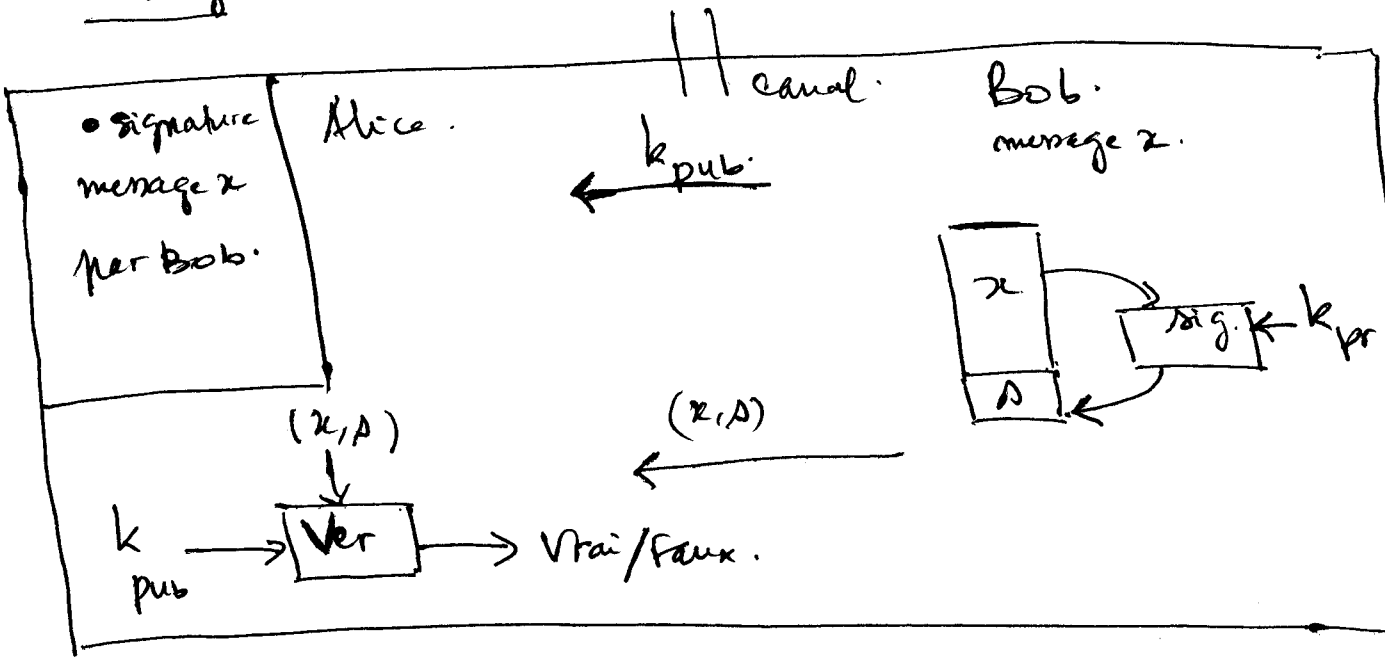
§ 4.1 Signature

C'est l'un des aspects de la cryptographie asymétrique les plus utilisés, par ex : certificats numériques pour le e-commerce, signature de contrats, mise à jour de logiciels, ...

On verra le principe de la signature numérique, signature par RSA, EL.GAMAL, DSA (digital signature Standard), Courbes elliptiques.

§§ 1.1. Principe et services sécurité de la signature.

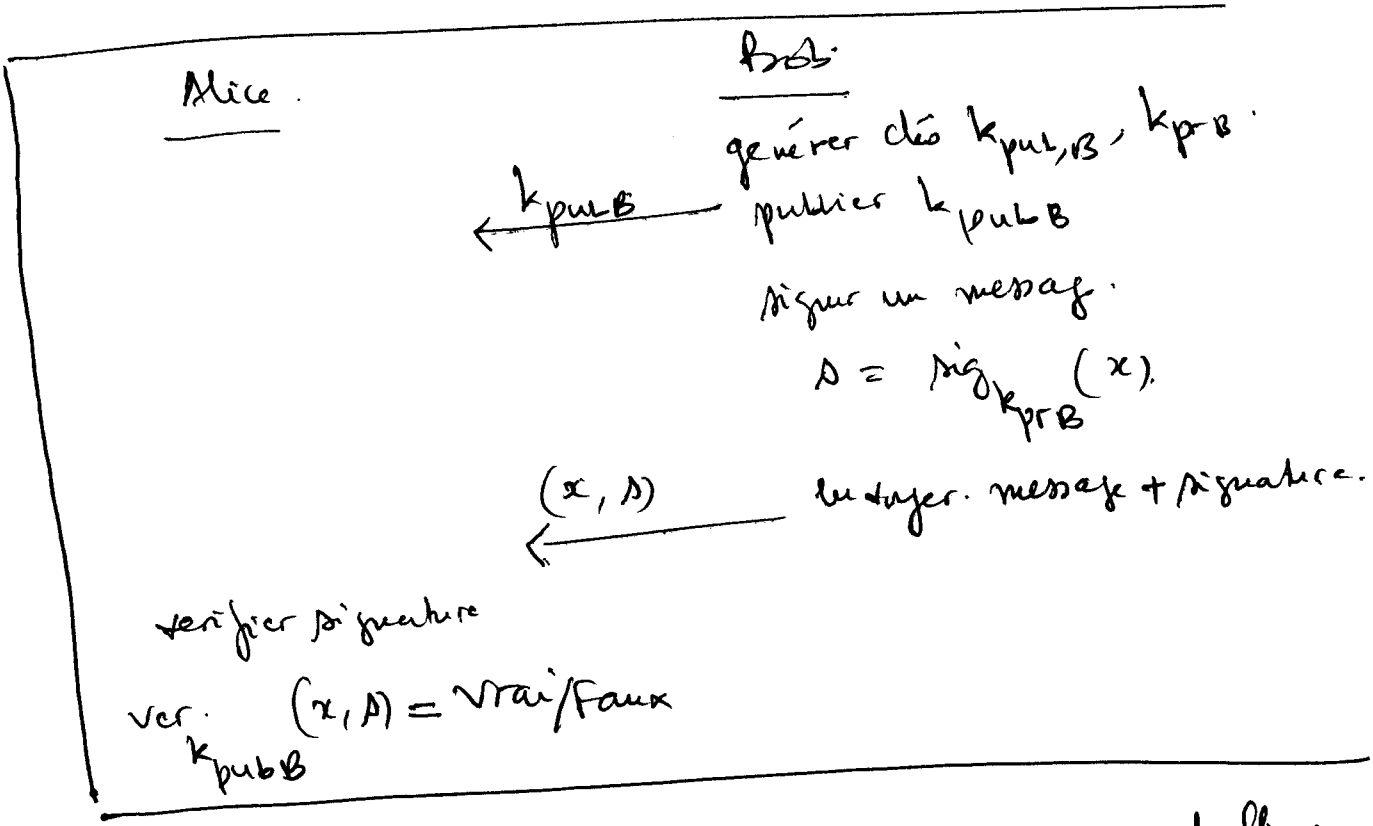
• La cryptographie asymétrique ne permet pas de traiter la distribution entre Alice et Bob ; le fait qu'1 des clés privées en crypto asymétrique permet de dire qu'un message émane de la personne ayant cette clé.



→ Bob envoie sa clé publique k_{pubB} à Alice ;

forme un message x ; le signe par un algorithme de signature sig et calcul $s = sig(x, k_{prB})$ en utilisant sa clé privé k_{prB}. Bob envoie la paire (x, s) à Alice qui la vérifie par un algorithme de vérification utilisant k_{pubB}. Noter que s est un nombre qui dépend du message x.

D'où le protocole :



→ Service de sécurité atteints avec nos connaissances actuelles :

1. Confidentialité : (par crypto symétrique, moins par Asym).
2. Intégrité. } signatures numériques et MAC.
3. Authentification }
4. Non-Répudiation. Signature.

§§1.2. Signature par RSA.

Bob veut signer un message x par RSA :

SET UP

- $k_{pr} = k_{pr_B} = (d)$.
- $k_{pub} = k_{pub_B} = (n, e)$.

$x \in \{0, \dots, n-1\}$.

Protocole de Base signature RSA.

Alice		Bob
		• $k_{pr} = d$, $k_{pub} = (n,e)$.
	$\xleftarrow{(n,e)}$	
		• $s = sig_{k_{pr}}(x) \equiv x^d \pmod n$.
	$\xleftarrow{(x,s)}$	
• $x' = s^e \pmod n$		
$x' \equiv x \pmod n \Rightarrow$ signature valide		
$x' \not\equiv x \pmod n \Rightarrow$ signature invalide		

ona : $s^e \equiv (x^d)^e = x^{de} \equiv x \pmod n$.

Car $de \equiv 1 \pmod{\phi(n)}$

Rq. Noter que les rôles de k_{pr} et k_{pub} sont échangés :

on signe avec la clé privée et on vérifie par k_{pub}.

Aspects calculatoire et sécurité signature RSA.

- la signature s est de long. $\lceil \log_2 n \rceil \simeq$ 1024. à 3072 bits.
  cette long peut être un pb pour les systèmes consommant peu d'energie tel téléphone cellulaire.
  Le Processus de génération de clés et les calculs ont les mê que pour l'infrastructure cryptage / decryptage.

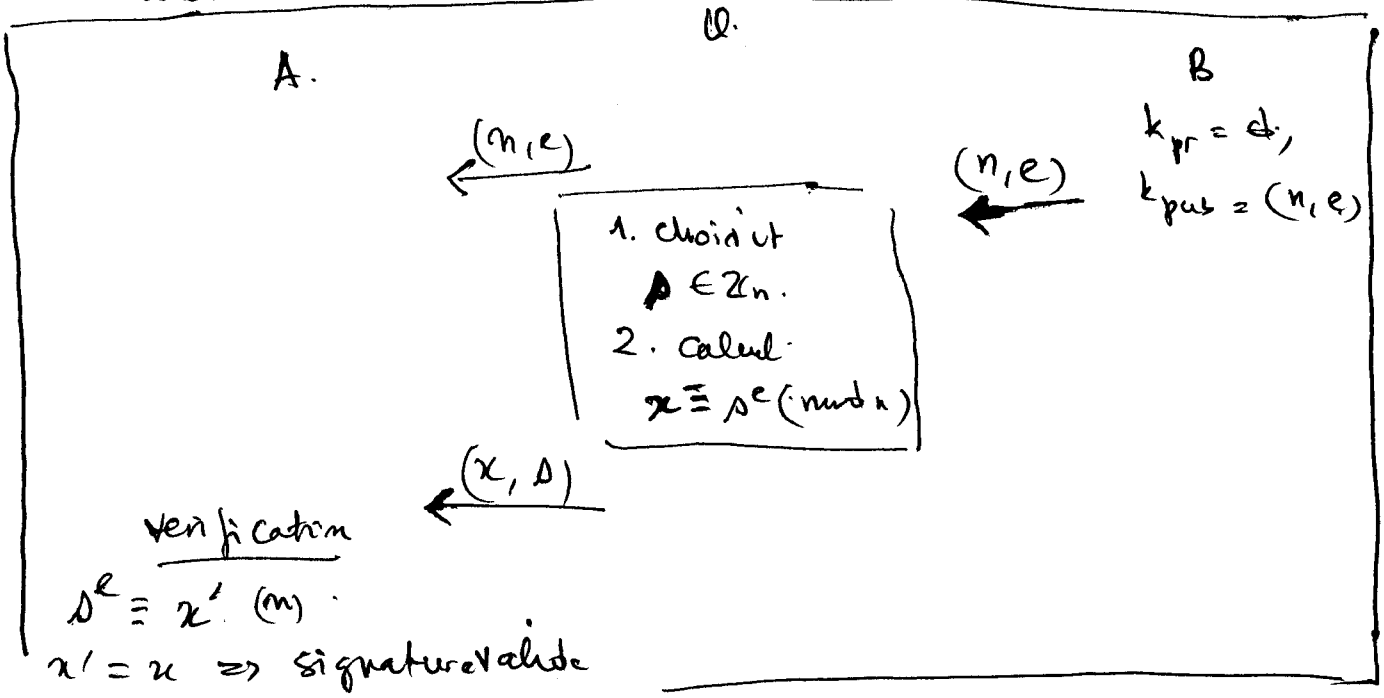
- Pour Prévenir les attaques, on utilise le concept de Certificat (Authentifié).

- Les attaques pour les mêmes que précédemment vu.

(chap3. §: 2. RSA) : factoriser n.

- Forger une fausse signature ; Man-in - the middle.

On peut générer une signature exacte pour un message aléatoire x:



L'attaque précédente est parée par le padding. (voir par. page 268). 14.5

§.4.2. Signature ElGAMAL :

Différent de RSA, ici, le cryptage et la signature sont assez distincts.

SETUP : génération de clés

1. Choisir un premier p assez grand
2. Choisir un élément primitif $\alpha \in \mathbb{Z}^*_p$ un sous-groupe de $\mathbb{Z}^*_p$.
3. Choisir aléatoirement $d \in \{2, 3, \dots, p-2\}$
4. Calculer $\beta = \alpha^d \pmod p$

La clé publique $k_{pub} = (p, \alpha, \beta)$, clé privée $k_{pr} = d$.

Génération de la signature.

1. Choisir aléatoi une clé éphémère $k_E \in \{0, \dots, p-2\}$ tq $pgcd(k_E, p-1) = 1$ (i.e. k_E est inversible mod $p-1$).
2. Calculer la signature :

$$(r, \Delta) = \left(\alpha^{k_E} \bmod p,\ (x - d \cdot r) k_E^{-1} \bmod p-1\right).$$

$Sig_{k_{pr}}(x, k_E) = (r, \Delta)$

Vérification : 1. Calculer $t = \beta^r \cdot r^\Delta \bmod p$.

2. Vérifier :

$$t \begin{cases} \equiv \alpha^x \bmod p \Rightarrow \text{signature valide} \\ \not\equiv \alpha^x \bmod p \Rightarrow \text{“ invalide} \end{cases}$$

$ver_{k_{pub}}(x, (r, \Delta))$

Prop. Le schéma ci-dessus est correct.

Preuve. on a :

$$\beta^r \cdot r^s \equiv (\alpha^d)^r \cdot (\alpha^{k_E})^s \pmod p.$$
$$\equiv \alpha^{dr + k_E \cdot s} \pmod p$$

La signature est valide si $\beta^r \cdot r^s \equiv \alpha^x \pmod p$! c.à.d si

$$\alpha^x \equiv \alpha^{dr + k_E \cdot s} \pmod p \quad (1).$$

D'après le théorème de Fermat, la relation (1) est vraie si

$$x \equiv dr + k_E \cdot s \pmod{p-1}.$$

càd si $s \equiv (x - dr) \cdot k_E^{-1} \pmod{p-1}$.

Ce qui est vrai par la construction 2. du protocole génération de la signature ci-dessus ☐.

- **Calcul et sécurité de ELGAMAL.**

→ La phase SETUP est identique à celle du cryptage (cf §. ELGAMAL, chap 3). La même difficulté revient à calculer le LD que précédemment.
On utilise des calculs rapides ↯ (M and S), exponentiation modulaire, Euclide Etendu. $p \geq 1024$.

→ **Attaque lorsque k_E est ré-utilisée :**

Supp que x_1 et x_2 ont même clé éphémère k_E. ⇒ $r_1 = r_2 = \alpha^{k_E} \cdot$ et

$$\begin{cases} s_1 \equiv (x_1 - d) k_E^{-1} \pmod{p-1} \\ s_2 \equiv (x_2 - d) \cdot k_E^{-1} \pmod{p-1}. \end{cases}$$

avec deux inconnues d (clé privée) et k_E.

$\Rightarrow$ $s_1 - s_2 \equiv (x_1 - x_2) \cdot k_E^{-1} \pmod{p-1}$

$\Rightarrow$ $k_E \equiv \dfrac{x_1 - x_2}{s_1 - s_2} \pmod{p-1}$,

si $\text{pgcd}(s_1 - s_2, p-1) \neq 1$, plusieurs solutions k_E.

et $d \equiv \dfrac{x_1 - s_1 k_E}{r} \cdot \pmod{p-1}$

Conclusion : changer les k_E assez souvent.

→ Exercice : Comme RSA, si Oscar peut forger une signature en utilisant un message aléatoire.

§4.3. Algorithme DSA (Digital signature Algorithm)

Utilisé en pratique, plus que ELGAMAL ; la taille de la signature est 320 bits.

§§4.3.1. Algorithme DSA.

Génération de clés DSA : SETUP.

1. Générer p premier $2^{1023} < p < 2^{1024}$.
2. Trouver un diviseur premier q de $p-1$, $2^{159} < q < 2^{160}$.
3. Trouver α avec $\text{ord}(\alpha) = q$
4. Choisir un entier aléatoire d, $0 < d < q$
5. Calculer $\beta \equiv \alpha^d \bmod p$.

Les clefs sont $k_{pub} = (p, q, \alpha, \beta)$

$k_{pr} = (d)$

Noter qu'on a deux structures de groupes sous-jacentes:

$\mathbb{Z}_p^*$ d'ordre $\sim 2^{1024}$. (1024 bits)

Sous-groupe $\mathbb{Z}_q^*$ d'ordre 2^{160} (160 bits).

→ Comme EL GAMAL, $sig(x) = (r, s)$ d'un message x, est un couple où $|r|_2 = |p|_2 = 160$, soit 360 bits.

Signature DSA (noté Sig_{DSA})

1. choisir k_E clé Ephémère, $0 < k_E < q$
2. Calculer $r \equiv (\alpha^{k_E} \bmod p) \pmod q$.
3. Calculer $s \equiv (SHA(x) + d \cdot r) \cdot k_E^{-1} \pmod q$

SHA est une fonction de hachage; DSA utilise SHA1 qui en regessant x message, ressort une empreinte de taille 16 bits. (voir § hachage)

Vérification Signature DSA noté Ver_{DSA}.

1. Calculer $w \equiv s^{-1} \pmod q$
2. Calculer $u_1 \equiv w \cdot SHA(x) \pmod q$
3. Calculer $u_2 \equiv w \cdot r \pmod q$
4. Calculer $v \equiv (\alpha^{u_1} \cdot \beta^{u_2} \bmod p) \pmod q$
5. Vérifier: $ver_{k_{pub}}(x, (r, s))$:

$$\begin{cases} v \equiv r \bmod q \Rightarrow \text{valide} \\ v \not\equiv r \bmod q \Rightarrow \text{invalide} \end{cases}$$

Proposition Le schéma DSA de signature ci-dessus est correct.

preuve • D'abord penser que faut-il prouver, i.e. DSA est correcte ?

Celà revient à montrer que $Sig_{Pr.}(x) = (r, s)$ vérifie la condition $v \equiv r \pmod q$.

$$s \equiv (SHA(x) + d \cdot r) \cdot k_E^{-1} \pmod q.$$

$$\Rightarrow \quad k_E = s^{-1} SHA(x) + d s^{-1} r \pmod q.$$

$$= u_1 + d u_2 \pmod q$$

$$\Rightarrow \quad \alpha^{k_E} \pmod p = \alpha^{u_1 + d u_2} \pmod p$$

$$= \alpha^{u_1} \cdot \beta^{u_2} \pmod p$$

on réduit $\pmod q$:

$$(\alpha^{k_E} \bmod p) \bmod q = ((\alpha^{u_1} \cdot \beta^{u_2}) \bmod p) \bmod q.$$

Comme $r = (\alpha^{k_E} \bmod p) \bmod q$ et $v = (\alpha^{u_1} \beta^{u_2} \bmod p) \bmod q$

$$\Rightarrow \quad r \equiv v \pmod q. \quad \square$$

Rq • la génération de clé est l'étape qui demande plus de calcul (engendrer p, q ; utiliser un générateur de nbrs premiers).

Attaques non génériques pour PLD. (2 fois).

§ 4.3. Signature avec CE. (Courbes elliptiques)

CE. : 160 – 256 bits Contre. 1024 – 3072 bits.
RSA et DL.

La signature avec CE est analogue à DSA, sur $\mathbb{Z}_p$.

Génération de clés

1. Soit E une courbe elliptique avec
   - p module
   - a, b coefficients
   - pt A engendrant le groupe cyclique d'ordre premier q.
2. Choisir aléatoirement $0 < d < q$
3. $B = dA$

Clés : $k_{pub} = (p, a, b, q, A, B)$

$k_{pr} = (d)$.

Signature CE.

1. Choisir alea. k_E, $0 < k_E < q$.
2. $R = k_E \cdot A$.
3. $r = x_R$ (coordonnée x de R).
4. Calculer $s = (h(x) + d \cdot r) \cdot k_E^{-1} \bmod q$

h : fonction de hachage

Vérification · CE.

1. $w = s^{-1} \pmod q$.
2. $u_1 = w \cdot h(x) \pmod q$
3. $u_2 = w \cdot r \pmod q$.
4. $P = u_1 A + u_2 B$.
5. $\text{Ver}_{\text{pub}}(x, (r,s))$!

$x_P \begin{cases} \equiv r \pmod q & \text{valide} \\ \not\equiv r \pmod q & \text{invalide} \end{cases}$

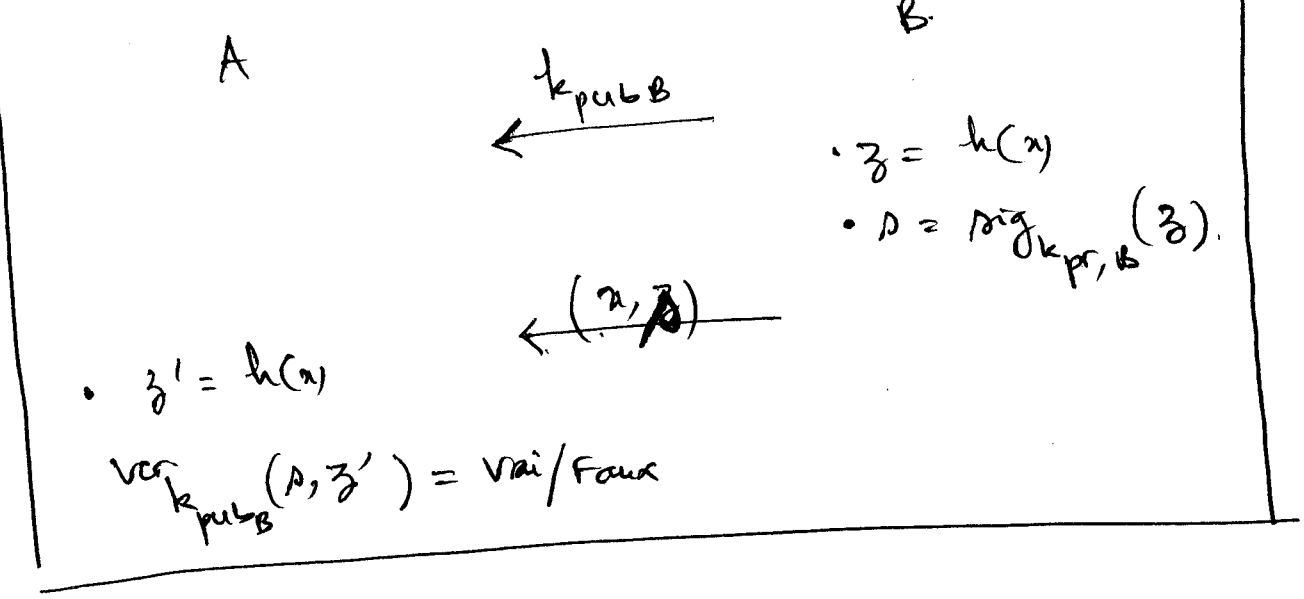
- **Exercice** Montrer que l'algo. de vérification est correcte.
  (i.e. que (r,s) vérifie $x_P \equiv r \pmod q$).

→ • Trouver une Courbe elliptique n'est pas facile.
  (NIST a standardisé 6 pt).
  • Les Calculs sont standards.
  • La sécurité repose sur le PLD des Courbes elliptiques.
  La meilleur attaque est en $O(\sqrt{q})$.

§4.2. Fonctions de hachage.

Les fonctions de hachage sont des primitives pratiques pour signer de long messages. Elles permettent de calculer une empreinte du message de longueur fixe sans utiliser de clé. Ainsi la signature de x est plus courte que x.

Protocole de base de signature avec fonction de hachage.



Propriétés de h

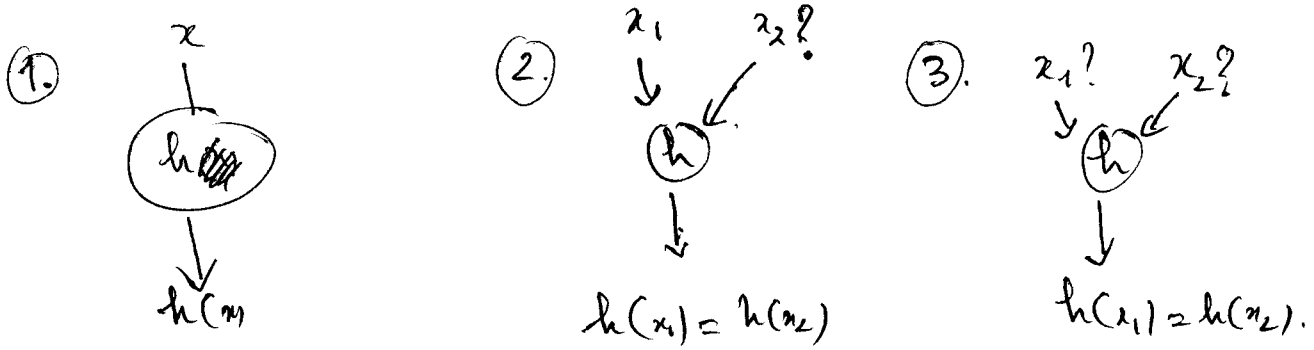
- doit être rapidement calculable,
- de longueur fixe (128 – 512 bits)
- sensible aux modifications : toute modification de x résulte en valeur différente de $h(x)$

§4.2.1. Sécurité et propriétés de h.

1. One-way : résistance à la pré-image.

2. Résistance à la seconde préimage.

3. Résistance aux collisions.



1. Etant donné $z = h(x)$. il est difficile de retrouver x.
En effet, dans le protocole suivant,
supposons que Bob crypte x le message, mais non pas la signature : Bob transmet $(e_k(\cdot) : DES, AES, \dots)$.

$$\left(e_k(x), sig_{k_{pub}}(z)\right).$$

supposons que Bob utilise RSA :

$$s = sig_{k_{pub}}(z) \equiv z^d \pmod n$$

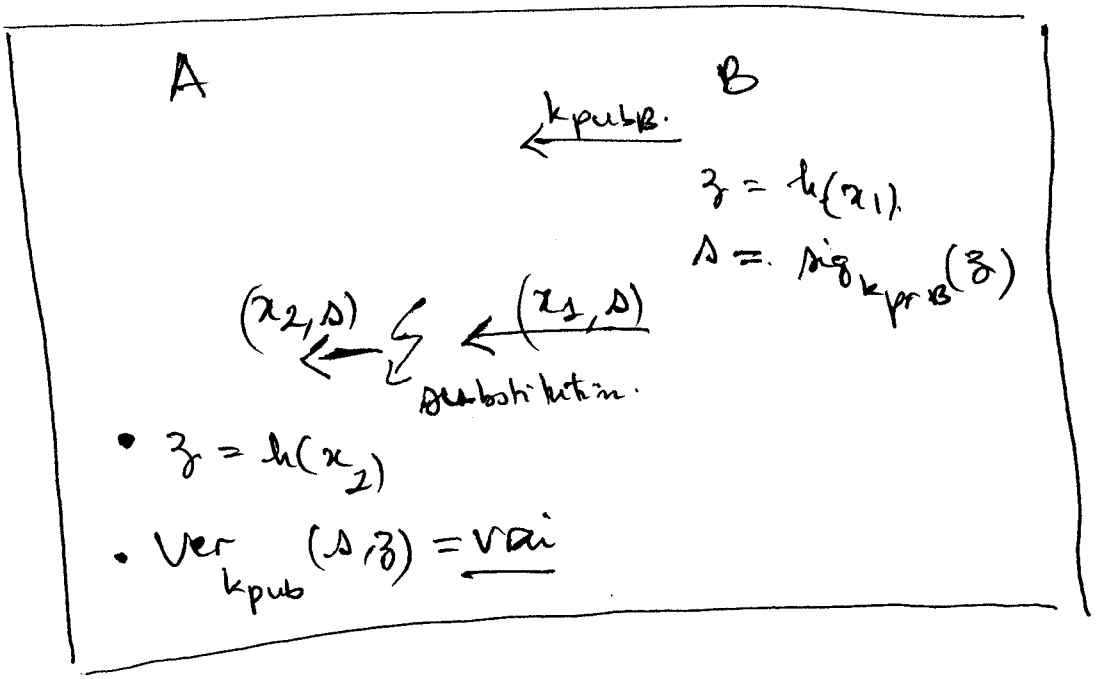
Oscar peut calculer $s^e \equiv z \pmod n$

Si h n'est pas à sens unique, O. peut calculer $x = h^{-1}(z)$. $\Rightarrow$ h doit être one-way.

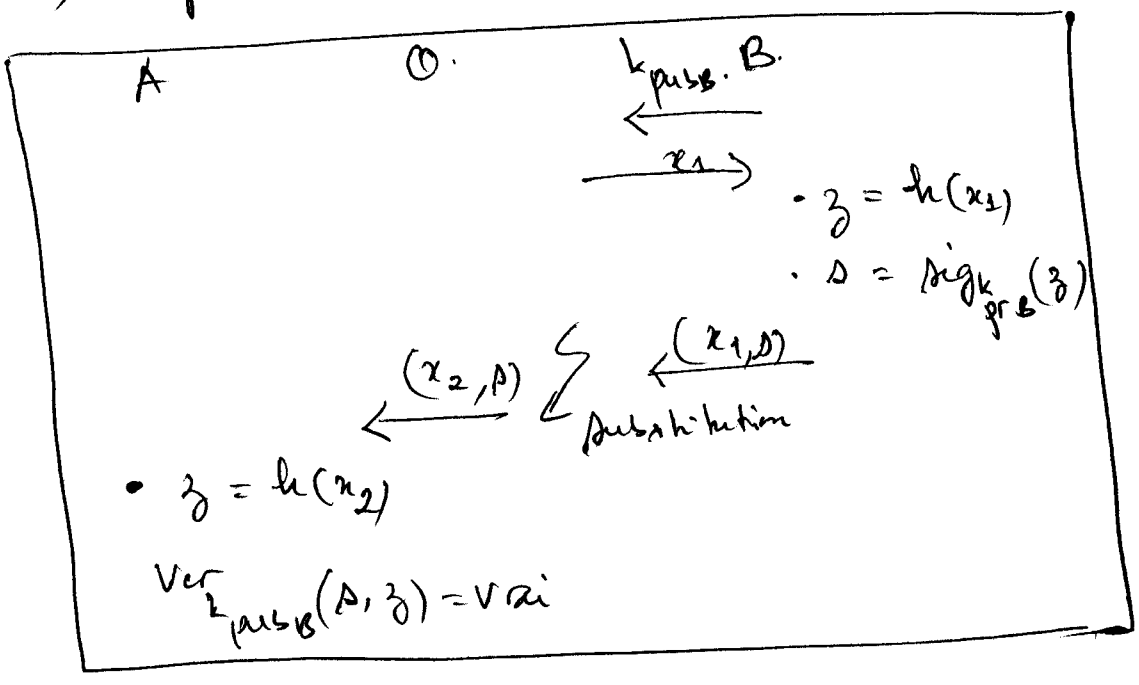
(2) Etant donné $h(x_1) = z$, il est difficile de trouver x_2 tq. $h(x_2) = z$. (Collision faible).

Supposons que Bob hache x_1 et signe $h(x_1)$.

Supposons que O peut trouver x_2 avec $h(x_1) = h(x_2)$:



(3) Collision forte : h est fortement résistante au collision, s'il est difficile de trouver x_1, x_2 tq $h(x_1) = h(x_2)$. Cette propriété est plus difficile que (2). A la base de cette propriété, O peut attaquer :



cette attaque marche théoriquement par le résultat du paradoxe des anniversaires, appliqué ici,

Soit t messages $(x_1, \dots, x_t)$ doit produire Oscar tels que $h(x_i) = h(x_j)$. $\bar{m}$ x_i message de taille n:

$$\mathbb{P}(\text{pas collision}) = \prod_{i=1}^{t-1} \left(1 - \frac{i}{2^n}\right)$$

Or $\lambda = 1 - \mathbb{P}(\text{pas collision})$

$$\simeq 1 - e^{t(t-1)/2^{n+1}}$$

$t \gg 1 \Rightarrow$ $$\boxed{t \simeq 2^{(n+1)/2} \cdot \left(\ln\left(\frac{1}{1-\lambda}\right)\right)^{1/2}} \quad (1).$$

Par ex: $n = 80$ bits.

$t \simeq 2^{80/2}$ bits. ☒

Exercice. Montrer en détail (1).

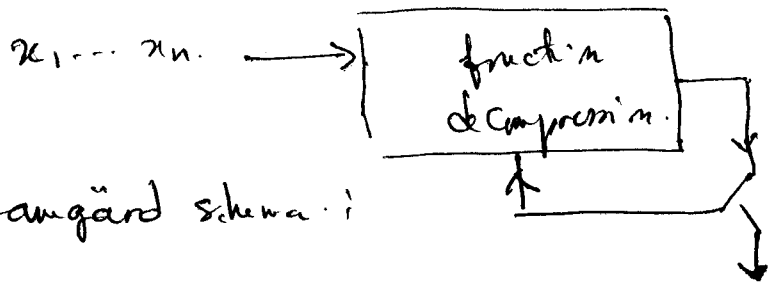
§ 4.2.2. Construction de fonctions de hachage.

Deux manières:

(A) Fonctions Dédiées: fonctions spécifiques pour le hachage.

(B) Fonctions Basé sur le B-cryptage: DES, AES,

Comme suit: x = message = $x_1 \cdots x_n$ divisé en bloc.



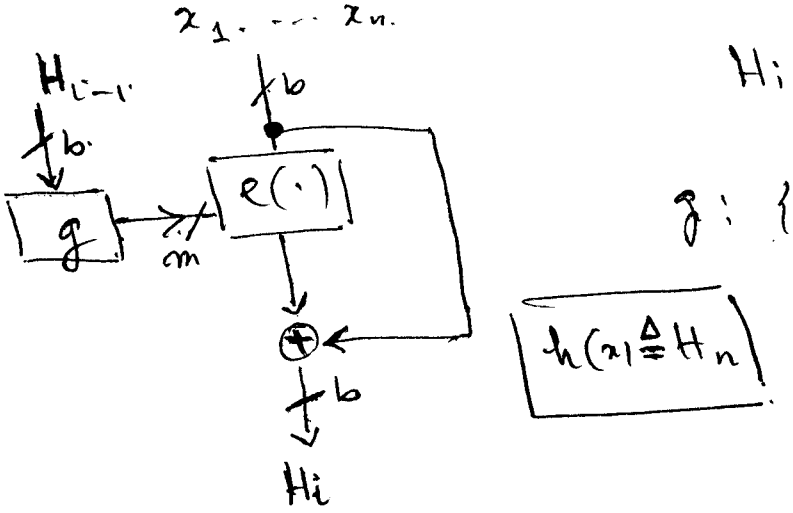
Merkle-Damgård schema:

(A.) Dédié :

- Famille MD4 ; spécialement MD5 (Rivest).
- Famille (SHA) : Secure Hach Algorithm pour remplacer MD4. ; SHA1 très utilisé actuellement.

voir page 307 de Paar et al où SHA1 est étudiée en détail. (resumée au réseau de Ficstel).

(B). $x = x_1 \dots x_n$ (blocs de taille fixe b)



$$H_i = e_{g(H_{i-1})}(x_i) \oplus x_i$$

$$g : \{0,1\}^b \longrightarrow \{0,1\}^m$$

$h(x) \triangleq H_n$

$e(\cdot)$ cryptage par DES, AES, ...

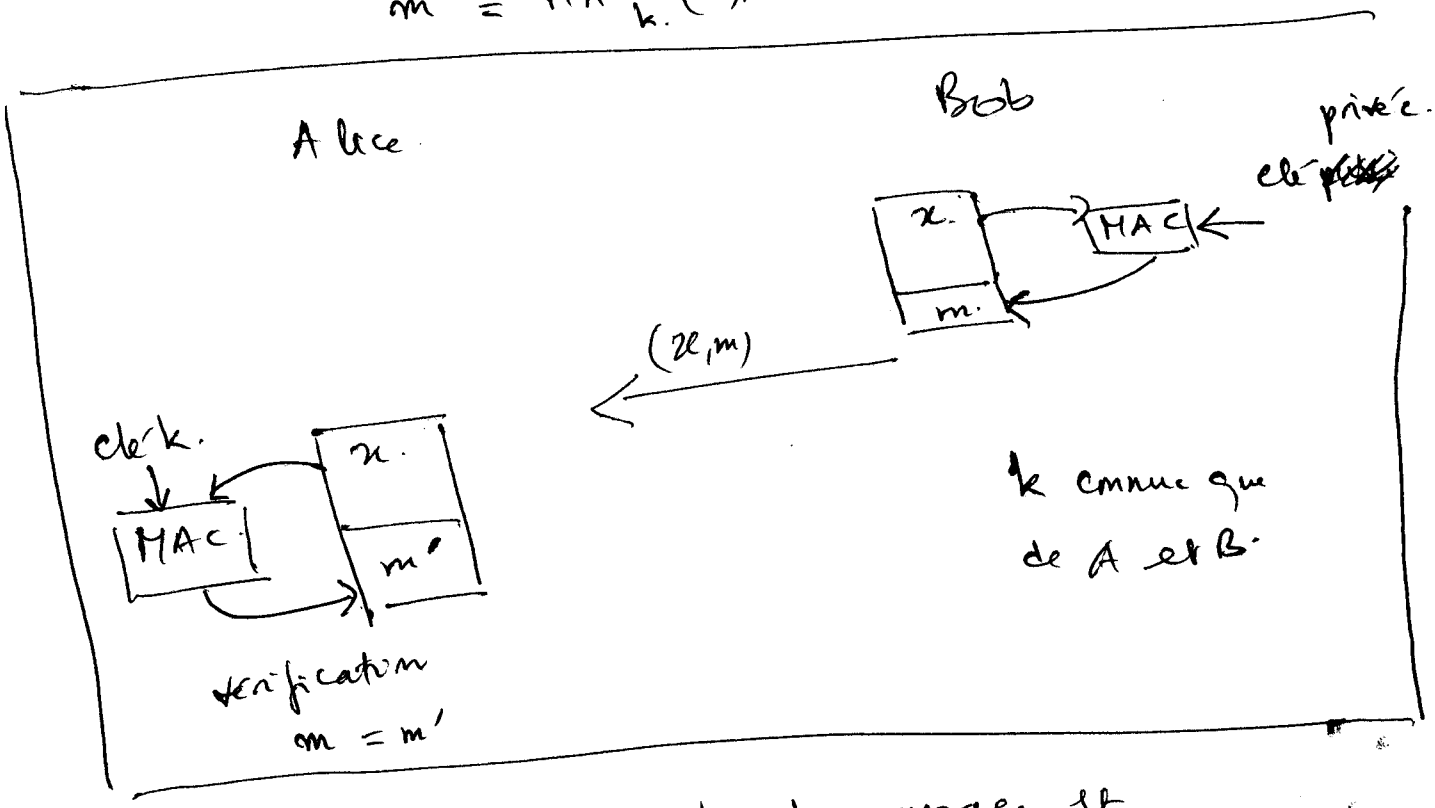
- Il existe plusieurs variantes (page 306 de Paar et al).

§ MAC : Message Authentification Codes.

§§ A) Propriétés et principes.

MAC ressemble à la signature numérique, en rajoutant un tag (empreinte) d'authentification au message x, mais il est basé pour le B-cryptage :

$$m = MAC_k(x).$$



⇒ • MAC assure l'intégrité des message et l'authenticité. En plus plus rapide que la cryptoclé publique. La non-répudiation n'est pas assurée.

• La construction est basée sur le B-cryptage et/ou hachage.

§§ B) Constructions de MAC.

1) HMAC : MAC à partir de hachages

HMAC utilise SHA-1.

La clé est hachée avec le message.

a) • préfixe MAC : pMAC :

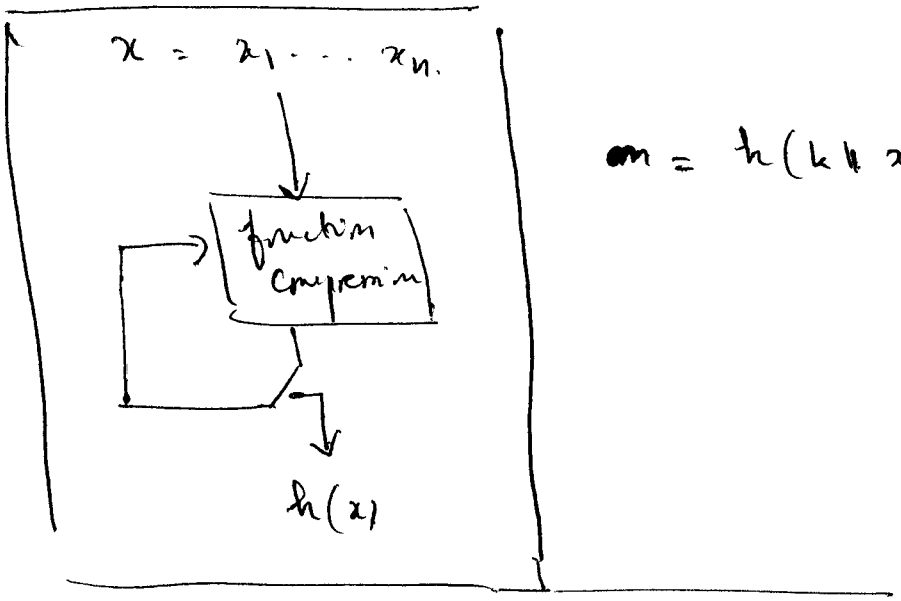
$$m = MAC_k(x) = h(k \| x).$$

b) • suffixe MAC : sMAC :

$$m = MAC_k(x) = h(x \| k).$$

a) Attaque de p.MAC : $m = h(k \| x)$.

On suppose que h est construit par (voir chap. hachage)



$m = h(k \| x_1, \ldots x_n)$.

A C B

$x = (x_1, \ldots, x_n)$

$m = h(k \| x_1, \ldots, x_n)$

Σ interception : (x, m) ←

x_0

$(x_1, \ldots, x_n, x_{n+1})$

$m_0 = h(m \| x_{n+1})$.

(x_0, m_0) ←

$m' = h(k \| x_1 \ldots x_n x_{n+1})$

$m' = m_0 \Rightarrow$

valide.

b) Attaque de DMAC : $m = h(x \| k)$

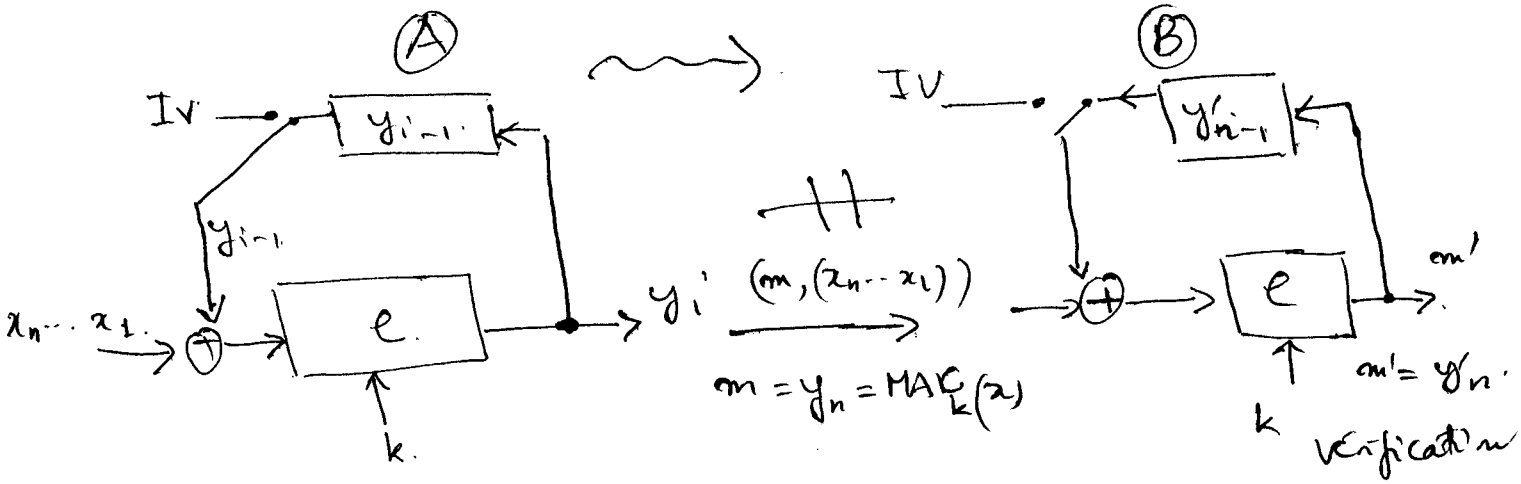
Supposons que O peut trouver x et x_0. $h(x) = h(x_0)$.

$m_0 = h(x \| k) = h(x_0 \| k)$: nature itérative de MAC. permet cette attaque.

2) MAC à partir de B-cryptage :

ex. mode CBC. (cipher Bloc chaining mode).

On obtient un mode CBC_MAC : $x = x_1 \cdots x_n$.



$$\begin{cases} y_1 = e_k(x_1 \oplus IV) \\ y_i = e_k(x_i \oplus y_{i-1}) \quad i \geq 2 \end{cases}$$

$m = MAC_k(x) = y_n$.

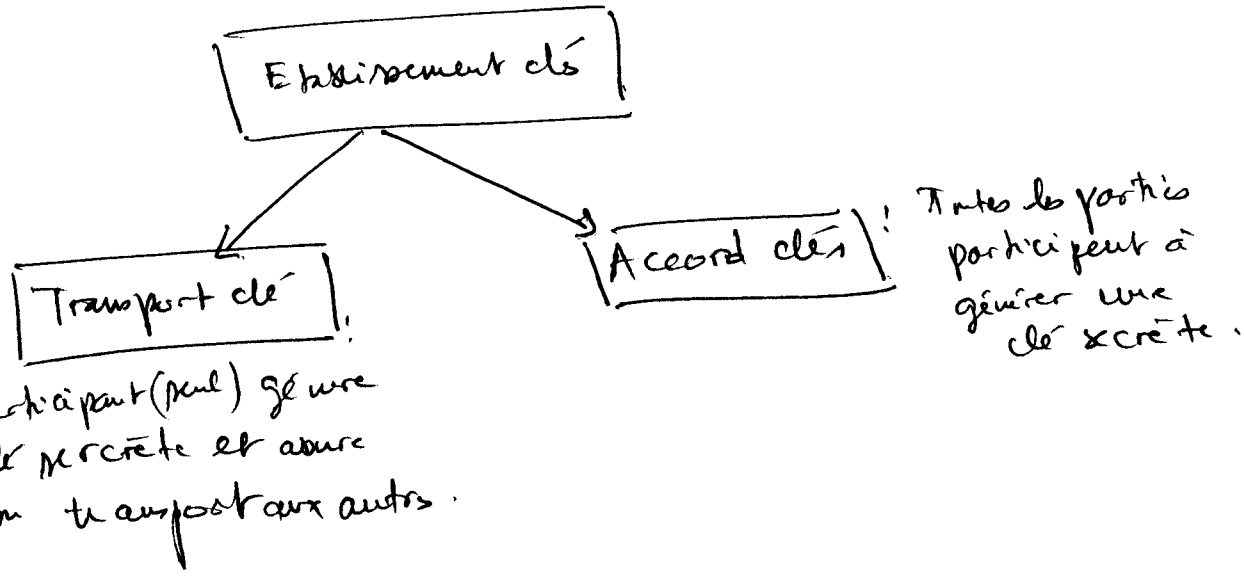
Avec le DES, utilisé par les banques.

- on peut utiliser Galois Counter MAC.

Chap 5. Gestions de clés.

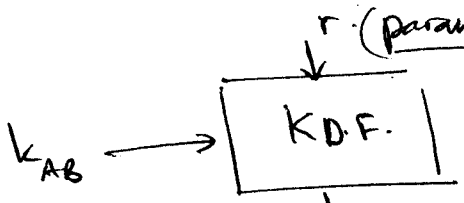
les techniques cryptographiques introduites dans les chap. précédents répondent aux 4 fonctions de sécurité citées (Chap. page). Reste quand même un point important qui est l'établissement de clés ; le protocole DH permet de répondre à cette question particulièrement.
Dans ce chap, on étudie l'établissement de clés dans les deux cas de figures (sym. et asym) et on introduit le concept important de certificats dans le cas asymétrique pour passer à l'attaque de l'homme-au-milieu.

§1. Généralités • L'établissement de clé consiste à créer des clés entre plusieurs partenaires (partage de secret). Selon les deux cas de figure :



• Clés éphémères et dérivation de clés

• Dans certains applications (Internet, mobile phone), la durée d'une clé est limitée. Au lieu de recalculer la clé secrète chaque fois, on dérive des clés en utilisant une fonction KDF. (Key derivation function) :



KDF est one-way.

Une réalisation possible est le protocole avec r = nonce.

(numerical value used once) généré par l'une des parties participantes :

KDF avec Nonce.

A	B
	• générer r nonce.
	$\xleftarrow{r}$
• dériver clé : $k_{ses} = e_{k_{AB}}(r)$	• dériver clé $k_{ses} = e_{k_{AB}}(r)$.

$e_{k_{AB}}$ peut être DES, AES, ...

Rq • on peut aussi utiliser $k_{ses} = HMAC_{k_{AB}}(r)$ au lieu de r, soit encore la valeur d'un compteur :

$$k_{ses} = HMAC_{k_{AB}}(cpt).$$

- Le problème de n utilisateurs.

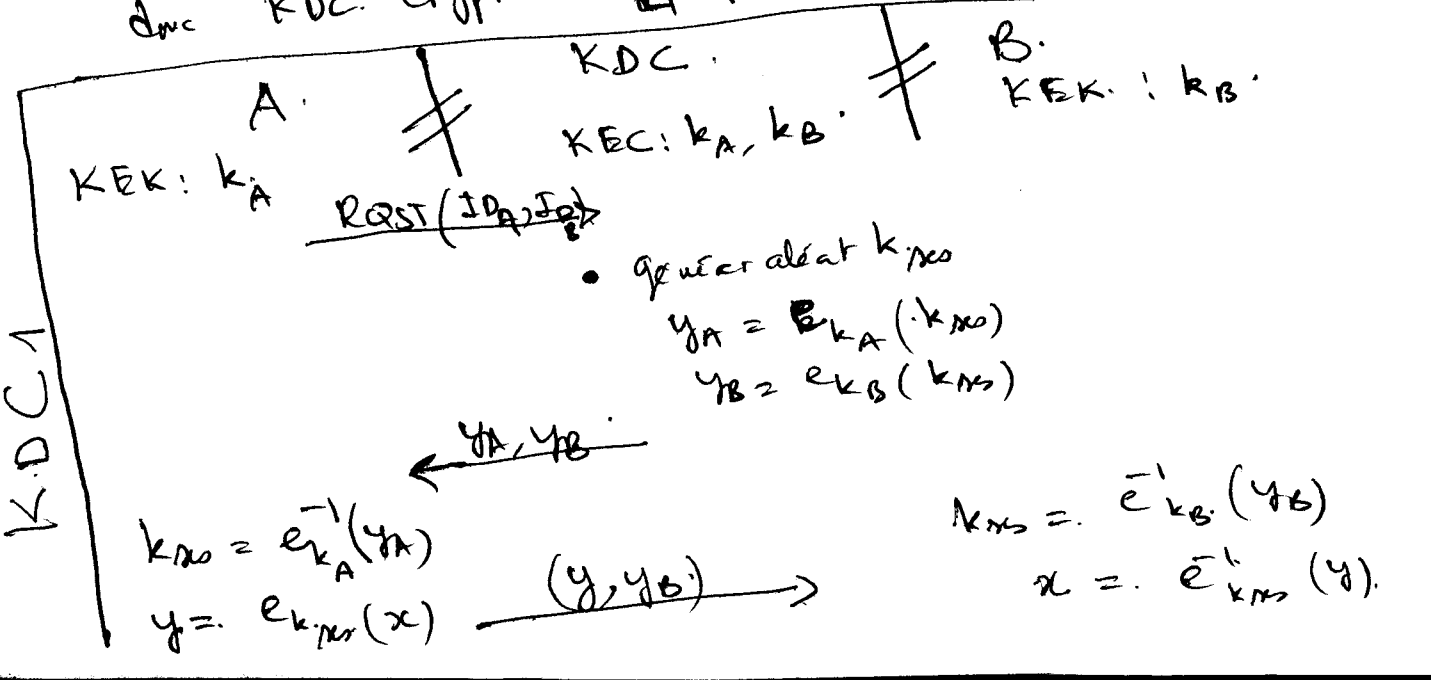
Ainsi le graphe de connexion est K_n. et deux participants qq. i et j partage une clé secrète k_{ij}. ($\Rightarrow \frac{n(n-1)}{2}$ clés).
D'où aussi pour utilisateur i de stocker $n-1$ clés.

§ 5.2. Etablissement de clés utilisant la cryptographie symétrique.

Même avec la crypto-symétrique, on peut réaliser le transport de clés (voir §1). à l'aide d'un centre de distribution (Ex: système embarqué; cell phone) de clés (KDC: Key Distri. Center).

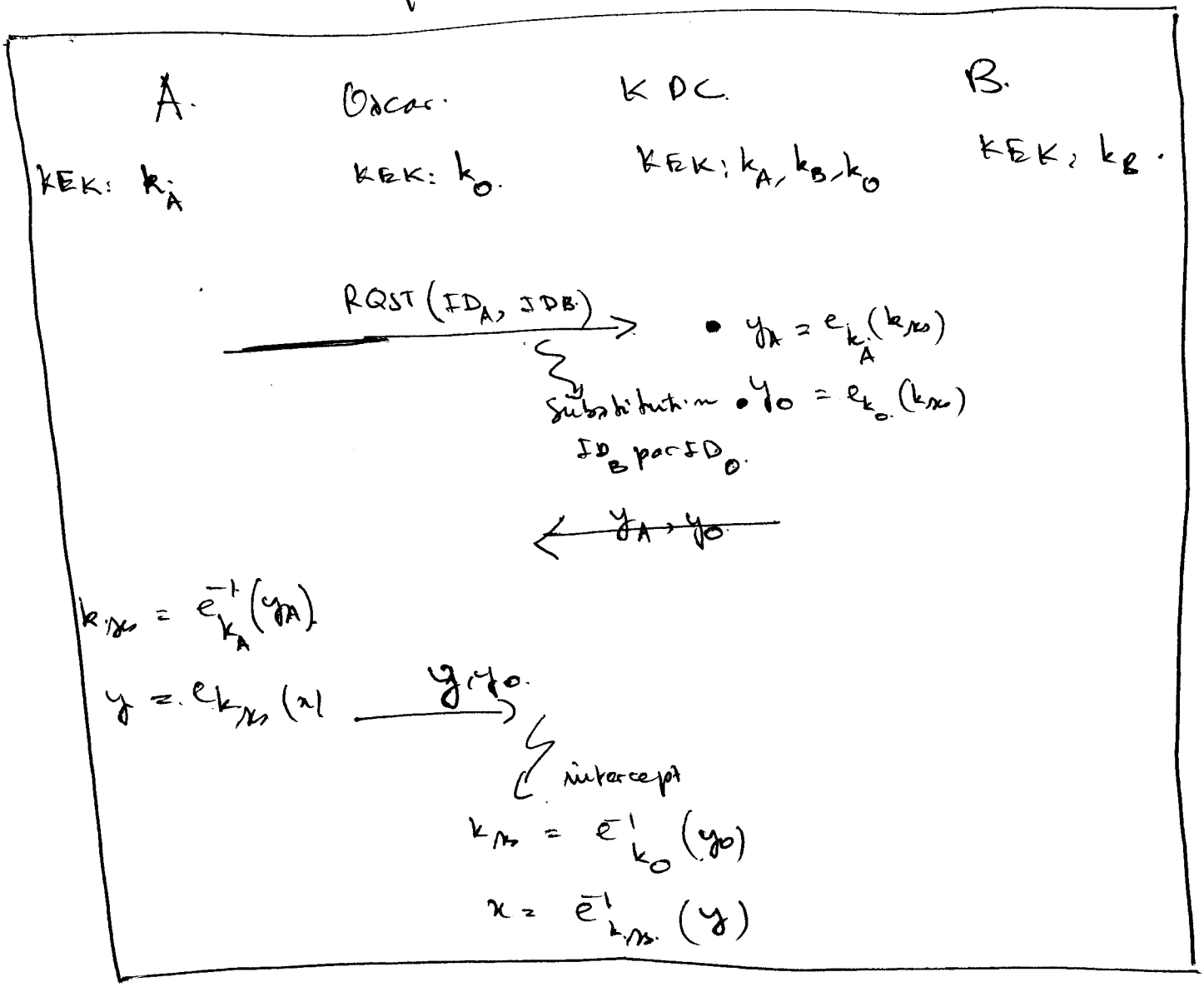
- K.D.C. !

Chaque utilisateur U partage au début une clé k_U notée KEK (Key Encryption Key) pour communiquer secrètement entre U et KDC.
Supposons que A demande une communication secrète avec B:
Le KDC répond par $y_A = e_{k_A}(k_{sess})$; $y_B = e_{k_B}(k_{sess})$;
donc KDC crypte le k_U par les demandeurs:



→ Remarque le Protocole KDC1 est fait pour empêcher une attaque par rejeu lorsque Oscar a accès à l'une de clés et au KDC. Entrée systématiquement k_{AB} à A et à B.

Une autre attaque de KDC1 est :



➡ Le système Kerberos permet de parer à ces attaques en confirmant la clé (voir page 340 de Paar et al), par utilisation de challenge — Réponse.

§ 5.3. Etablissement de clés par cryptographie Asymétrique

Rappelons :

protocole DH. d'échange de clés

A	B
• choisir aléato. $a = k_{prA}$	• choisir aléat. $b = k_{prB}$
• calculer $A = \alpha^a \bmod p$. $= k_{pubA}$	• calculer $k_{pubB} = \alpha^b \bmod p$.
$\xrightarrow{A}$	
	$\xleftarrow{B}$
• $k_{AB} = B^a \bmod p$.	• $k_{AB} = A^b \bmod p$.

(et l'attaque suivante (valable pour tout syst. asymétrique)

Attaque MIM. (man-in-the-middle).

O : Oscar

A	B.
• $a = k_{prA}$	• $b = k_{prB}$.
• $A = k_{pubA} = \alpha^a \bmod p$.	• $B = \alpha^b \bmod p$. $= k_{pubB}$.
$\xrightarrow{A}$ O intercepte $\tilde{A} = \alpha^o$.	$\xrightarrow{\tilde{A}}$
$\xleftarrow{\tilde{B}}$ O intercepte $\tilde{B} = \alpha^o$.	$\xleftarrow{B}$
• $k_{AO} = (\tilde{B})^a \bmod p$.	• $k_{BO} = (\tilde{A})^b \bmod p$.

• $k_{AO} = \tilde{A}^o \bmod p$

• $k_{BO} = \tilde{B}^o \bmod p$.

⇒ O peut intercepter et lire les messages entre A et B à leur insu.

La notion de certificat permet de parer à cette attaque MIM.

§§ 5.3.1. Certificats.

Le pb avec l'attaque MIM est que les clefs publiques ne sont pas authentiques. En pratique la clé publique est de la forme

$$k_A = (k_{pubA}, ID_A).$$ où ID_A : mail, nom, ...

que l'Oscar transforme en :

$$k_A = (k_{pubO}, ID_A)$$

D'où le titre le principe :

> Même si les protocoles à clé publique ne demande pas un canal sécurisé, ils exigent un canal d'authentification pour la distribution de clé

D'où en utilisant une signature

$$Cert_A = \left[(k_{pubA}, ID_A), sig_{k_{pr}}(k_{pubA}, ID_A)\right].$$

et le destinataire a un algorithme de vérification de la signature, qui est publique. La signature est faite par une 3ème partie appelé CA (Certification Authority).
CA génère et distribue des certificats à tous les utilisateurs du système, par deux façons :

A génère des clés et demande un certificat

A

A.

générer k_{pubA}, k_{prA}. $\xrightarrow{RQST(k_{pubA}, ID_A)}$

- vérifier ID_A
- $S_A = Sig_{prCA}(k_{pubA}, ID_A)$
- $Cert_A = [(k_{pubA}, ID_A), S_A]$.

$\xleftarrow{Cert_A}$

CA génère les clés et les certificats.

A

CA

$\xrightarrow{RQST(ID_A)}$

- vérifier ID_A
- générer k_{prA}, k_{pubA}
- $S_A = Sig_{prCA}(k_{pubA}, IDA)$
- $Cert_A = [(k_{pubA}, IDA), S_A]$.

$\xleftarrow{(Cert_A, k_{prA})}$

Rq. Noter que, quand même, ce protocole est fait une seule fois (SET UP) par des canaux sûrs. Sinon pb!

Protocole DH avec certificats.

A	B
• Calculer $k_{prA} = a$ (aléa)	• Calculer $k_{prB} = b$ (aléa)
$A = \alpha^a \bmod p$.	$B = \alpha^b \bmod p$.
$Cert_A = [(A, ID_A), A_A]$.	$Cert_B = [(B, ID_B), A_B]$
$\xrightarrow{Cert_A}$	
$\xleftarrow{Cert_B}$	
• vérifier certificat $Ver_{k_{pub,CA}}(Cert_B)$.	• vérifier certificat $Ver_{k_{pub,CA}}(Cert_A)$
• Calculer la clé de session $k_{AB} = B^a \bmod p$.	• Calculer clé session $k_{AB} = A^b \bmod p$.

§§.5.3.2. PKI et CA.

- PKI = Public-Key Infrastructure est l'ensemble des techniques pour établir un système cryptographique à clé publique y compris la certification, qui est une tâche lourde à mettre en œuvre.
- La structure d'un certificat contient les informations techniques et personnelles : # série, algo de signature, autorité de certifi, période de validité, ID, k_{pub} et paramètres, etc.