# Cryptographie et algorithmique pour la sécurité

Clément PERNET

M1 MAI, UFR-IMA, Université J. Fourier

# Plan

# Historique (rapide)

# Le chiffre de Vigénère

# Chiffre de Vigénère: rappel

Clé $K$ de taille $m$.

$$C_i = M_i + K_{i \mod m} \mod 26$$

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clair | L | A | V | I | E | E | S | T | B | E | L | L | E |
| Clé | B | O | N | J | O | U | R | B | O | N | J | O | U |
| Décalage | 1 | 14 | 13 | 9 | 14 | 20 | 17 | 1 | 14 | 13 | 9 | 14 | 20 |
| Chiffré | M | O | I | R | S | Y | J | U | P | R | U | Z | Y |

# Chiffre de Vigénère: rappel

Clé $K$ de taille $m$.

$$C_i = M_i + K_{i \mod m} \mod 26$$

| Clair | L | A | V | I | E | E | S | T | B | E | L | L | E |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clé | B | O | N | J | O | U | R | B | O | N | J | O | U |
| Décalage | 1 | 14 | 13 | 9 | 14 | 20 | 17 | 1 | 14 | 13 | 9 | 14 | 20 |
| Chiffré | M | O | I | R | S | Y | J | U | P | R | U | Z | Y |

### Cryptanalyse

1. Trouver la longueur de la clé
2. Casser la clé par une analyse fréquentielle

# Longueur *m* de la clé: Babbage 1854 et Kasiski 1863

Idée: si répétition d'une séquence dans $C$ à distance $\delta$

- Soit c'est le même source, codé sur la même partie de clé
- Soit c'est une coincidence (moins probable)

$\Rightarrow$la distance $\delta$ est probablement multiple de $m$

KQOWEFVJPUJUUNUKGLMEKJINMWUXFQMKJBGWRLFN
FGHUDWUUMBSVLPSNCMUEKQCTESWREEKOYSSIWCTUAX
YOTAPXPLWPNTCGOJBGFQHTDWXIZAYGFFNSXCSEYN
CTSSPNTUJNYTGGWZGRWUUNEJUUQEAPYMEKQHUIDU
XFPGUYTSMTFFSHNUOCZGMRUWEYTRGKMEEDCTVREC
FBDJQCUSWVBPNLGOYLSKMTEFVJJTWWMFMWPNMEMT
MHRSPXFSSKFFSTNUOCZGMDOEOYEEKCPJRGPMURSK
HFRSEIUEVGOYCWXIZAYGOSAANYDOEOYJLWUNHAME
BFELXYVLWNOJNSIOFRWUCCESWKVIDGMUCGOCRUWG
NMAAFFVNSIUDEKQHCEUCPFCMPVSUDGAVEMNYMAMV
LFMAOYFNTQCUAFVFJNXKLNEIWCWODCCULWRIFTWG
MUSWOVMATNYBUHTCOCWFYTNMGYTQMKBBNLGFBTWO
JFTWGNTEJKNEEDCLDHWTVBUVGFBIJG

KQOWEFVJPUJUUNUKGLMEKJINMWUXFQMKJBGWRLFN
FGHUDWUUMBSVLPSNCMUEKQCTESWREEKOYSSIWCTUAX
YOTAPXPLWPNTCGOJBGFQHTDWXIZAYGFFNSXCSEYN
CTSSPNTUJNYTGGWZGRWUUNEJUUQEAPYMEKQHUIDU
XFPGUYTSMTFFSHNUOCZGMRUWEYTRGKMEEDCTVREC
FBDJQCUSWVBPNLGOYLSKMTEFVJJTWWMFMWPNMEMT
MHRSPXFSSKFFSTNUOCZGMDOEOYEEKCPJRGPMURSK
HFRSEIUEVGOYCWXIZAYGOSAANYDOEOYJLWUNHAME
BFELXYVLWNOJNSIOFRWUCCESWKVIDGMUCGOCRUWG
NMAAFFVNSIUDEKQHCEUCPFCMPVSUDGAVEMNYMAMV
LFMAOYFNTQCUAFVFJNXKLNEIWCWODCCULWRIFTWG
MUSWOVMATNYBUHTCOCWFYTNMGYTQMKBBNLGFBTWO
JFTWGNTEJKNEEDCLDHWTVBUVGFBIJG

KQOWEFVJPUJUUNUKGLMEKJINMWUXFQMKJBGWRLFN
FGHUDWUUMBSVLPSNCMUEKQCTESWREEKOYSSIWCTUAX
YOTAPXPLWPNTCGOJBGFQHTDWXIZAYGFFNSXCSEYN
CTSSPNTUJNYTGGWZGRWUUNEJUUQEAPYMEKQHUIDU
XFPGUYTSMTFFSHNUOCZGMRUWEYTRGKMEEDCTVREC
FBDJQCUSWVBPNLGOYLSKMTEFVJJTWWMFMWPNMEMT
MHRSPXFSSKFFSTNUOCZGMDOEOYEEKCPJRGPMURSK
HFRSEIUEVGOYCWXIZAYGOSAANYDOEOYJLWUNHAME
BFELXYVLWNOJNSIOFRWUCCESWKVIDGMUCGOCRUWG
NMAAFFVNSIUDEKQHCEUCPFCMPVSUDGAVEMNYMAMV
LFMAOYFNTQCUAFVFJNXKLNEIWCWODCCULWRIFTWG
MUSWOVMATNYBUHTCOCWFYTNMGYTQMKBBNLGFBTWO
JFTWGNTEJKNEEDCLDHWTVBUVGFBIJG

| | | Valeurs possibles de $m$ | | | |
|---|---|---|---|---|---|
| Séquence | Distance | 2 | 3 | 5 | 19 |
| WUU | 95 | | | $\surd$ | $\surd$ |
| EEK | 200 | $\surd$ | | $\surd$ | |
| WXIZAY | 190 | $\surd$ | | $\surd$ | $\surd$ |
| NUOCZGM | 80 | $\surd$ | | $\surd$ | |
| DOEOY | 45 | | $\surd$ | $\surd$ | |
| GMU | 90 | $\surd$ | $\surd$ | $\surd$ | |

KQOWEFVJPUJUUNUKGLMEKJINMWUXFQMKJBGWRLFN
FGHUDWUUMBSVLPSNCMUEKQCTESWREEKOYSSIWCTUAX
YOTAPXPLWPNTCGOJBGFQHTDWXIZAYGFFNSXCSEYN
CTSSPNTUJNYTGGWZGRWUUNEJUUQEAPYMEKQHUIDU
XFPGUYTSMTFFSHNUOCZGMRUWEYTRGKMEEDCTVREC
FBDJQCUSWVBPNLGOYLSKMTEFVJJTWWMFMWPNMEMT
MHRSPXFSSKFFSTNUOCZGMDOEOYEEKCPJRGPMURSK
HFRSEIUEVGOYCWXIZAYGOSAANYDOEOYJLWUNHAME
BFELXYVLWNOJNSIOFRWUCCESWKVIDGMUCGOCRUWG
NMAAFFVNSIUDEKQHCEUCPFCMPVSUDGAVEMNYMAMV
LFMAOYFNTQCUAFVFJNXKLNEIWCWODCCULWRIFTWG
MUSWOVMATNYBUHTCOCWFYTNMGYTQMKBBNLGFBTWO
JFTWGNTEJKNEEDCLDHWTVBUVGFBIJG

| | | Valeurs possibles de *m* | | | |
|---|---|---|---|---|---|
| Séquence | Distance | 2 | 3 | 5 | 19 |
| WUU | 95 | | | $\surd$ | $\surd$ |
| EEK | 200 | $\surd$ | | $\surd$ | |
| WXIZAY | 190 | $\surd$ | | $\surd$ | $\surd$ |
| NUOCZGM | 80 | $\surd$ | | $\surd$ | |
| DOEOY | 45 | | $\surd$ | $\surd$ | |
| GMU | 90 | $\surd$ | $\surd$ | $\surd$ | |

# Longueur *m* de la clé: Friedman

Soit $n_A$ le nombre d'occurrences de A dans le texte de taille $n$
$X$, $Y$ deux lettres tirées au hasard.

$$P(X = A, Y = A) = \frac{C^2_{n_A}}{C^2_n} = \frac{\frac{n_A(n_A-1)}{2}}{\frac{n(n-1)}{2}} = \frac{n_A(n_A - 1)}{n(n-1)}$$

# Longueur $m$ de la clé: Friedman

Soit $n_A$ le nombre d'occurrences de A dans le texte de taille $n$
$X, Y$ deux lettres tirées au hasard.

$$P(X = A, Y = A) = \frac{C_{n_A}^2}{C_n^2} = \frac{\frac{n_A(n_A-1)}{2}}{\frac{n(n-1)}{2}} = \frac{n_A(n_A-1)}{n(n-1)}$$

**Indice de coïncidence**

Probabilité que deux lettres tirées au hasard soient égales

$$\mathrm{IC} = \sum_{\lambda=A}^{Z} \frac{n_\lambda(n_\lambda - 1)}{n(n-1)} = \sum_{\lambda=A}^{Z} f_\lambda \left( f_\lambda + \frac{f_\lambda - 1}{n-1} \right)$$

$\mathrm{IC} \to \sum_{\lambda=A}^{Z} f_\lambda^2$
$\Rightarrow$caractérisé par les fréquences (donc la langue).

# Indice de coïncidence

| Allemand | Anglais | Français | Aléatoire unif. |
|---|---|---|---|
| IC(g)=0,07187 | IC(e)=0,06577 | IC(f)=0,07405 | IC(a)=0,03846 |

# Indice de coïncidence

| Allemand | Anglais | Français | Aléatoire unif. |
|---|---|---|---|
| IC(g)=0,07187 | IC(e)=0,06577 | IC(f)=0,07405 | IC(a)=0,03846 |

## Propriété

- *IC est invariant par substitution mono-alphabétique*
- *Si $v$ est chiffré par Vigenère*

$$IC(v) \approx \frac{1}{m} IC(f) + (1 - \frac{1}{m}) IC(a)$$

$\Rightarrow m \approx \frac{IC(f) - IC(a)}{IC(v) - IC(a)}$

# Calculer la clé: test de Friedman

On connaît $m$ $\Rightarrow$on forme la matrice de dimension $m \times n/m$

$$A = \begin{bmatrix} c_0 & c_m & c_{2m} & \cdots \\ c_1 & c_{m+1} & c_{2m+1} & \cdots \\ \vdots & \vdots & \vdots & \\ c_{m-1} & c_{m-1+m} & c_{m-1+2m} & \cdots \end{bmatrix}$$

$\Rightarrow$Chaque ligne $L_i$ est générée par la même lettre de la clé: $K_i$

**Indice de coïncidence jointe**

Probabilité que les alphabets de $X$ et $Y$ correspondent: $\mathrm{MIC}(X, Y) = \sum_{\lambda=A}^{Z} P(X = \lambda)P(Y = \lambda)$

Pour tout $k = 0 \ldots 25$:

- Ajouter $k$ à $L_1$ (notée $L_1^k$)
- Calculer l'indice de coïncidence jointe $M_{i,k} = \mathrm{IC}(L_1^k, L_i)$
- Si le max de la ligne $i$ de $M$ est en col $k_i$, alors $K_i - K_0 = k_i$

$\Rightarrow$on a tous les coefficients de la clé $K_1, \ldots, K_{m-1}$ sauf $K_0$.
$\Rightarrow$analyse fréq. classique pour les 26 possibilités restantes.

# Plan

# Plan

# Analyse de complexité: généralités

- Coût des algorithmes
  - en temps de calcul (nombre d'opérations)
  - en place mémoire (espace de stockage)
- Exprimé en fonction de la taille des entrées
- Différents modèles de complexité:

  algébrique: nb d'ops dans le domaine (corps, anneau,...)
  binaire: nombre d'opérations sur des bits
- Différentes mesures:

  pire cas: la pire instance des entrées
  en moyenne: sur l'ensemble des instances des entrées

# Analyse de complexité: généralités

**Notations de Landau:**

- $f(n) = \mathcal{O}(g(n))$ ssi $0 \leq f(n) \leq Kg(n)$ pour $n \geq n_0$
- $f(n) = \Omega(g(n))$ ssi $g(n) = \mathcal{O}(f(n))$
- $f(n) = \Theta(g(n))$ ssi $f(n) = \mathcal{O}(g(n))$ et $g(n) = \mathcal{O}(f(n))$

# Analyse de complexité: généralités

## Notations de Landau:

- $f(n) = \mathcal{O}(g(n))$ ssi $0 \leq f(n) \leq Kg(n)$ pour $n \geq n_0$
- $f(n) = \Omega(g(n))$ ssi $g(n) = \mathcal{O}(f(n))$
- $f(n) = \Theta(g(n))$ ssi $f(n) = \mathcal{O}(g(n))$ et $g(n) = \mathcal{O}(f(n))$

## Notations poly-logarithmiques (*soft-O*)

$f(n) = \tilde{\mathcal{O}}(g(n))$ ssi $f(n) = \mathcal{O}\left(f(n) \log^e f(n)\right)$

Exemple: $n \log n \log\log n = \tilde{\mathcal{O}}(n)$

# Ordres de grandeur

- taille d'un entier $n$: $t_n = \lceil \log_2 n \rceil$ bits. $n = \mathcal{O}\left(e^{t_n}\right)$.

# Ordres de grandeur

- taille d'un entier $n$: $t_n = \lceil \log_2 n \rceil$ bits. $n = \mathcal{O}\left(e^{t_n}\right)$.
- Vitesse d'un PC: 3GHz $\Rightarrow 3 \times 10^9$ additions par seconde
  - Lumière du projecteur: 3m à $300\,000 km/s$ $\Rightarrow 10^{-8}$s
  - 30 additions effectuées avant que la lumière arrive à l'écran

# Ordres de grandeur

- taille d'un entier $n$: $t_n = \lceil \log_2 n \rceil$ bits. $n = \mathcal{O}\left(e^{t_n}\right)$.
- Vitesse d'un PC: 3GHz $\Rightarrow$ $3 \times 10^9$ additions par seconde
  - Lumière du projecteur: 3m à $300\,000 km/s$ $\Rightarrow$ $10^{-8}$s
  - 30 additions effectuées avant que la lumière arrive à l'écran
- Age de l'univers : 15 milliard $\times 365 \times 24 \times 3600 \approx 5.10^{17} s$
- Nombres d'électrons dans l'univers : $\approx 10^{64}$

# Ordres de grandeur

- taille d'un entier $n$: $t_n = \lceil \log_2 n \rceil$ bits. $n = \mathcal{O}\left(e^{t_n}\right)$.
- Vitesse d'un PC: 3GHz $\Rightarrow$3 $\times$ $10^9$ additions par seconde
  - Lumière du projecteur: 3m à 300 000$km/s$ $\Rightarrow$$10^{-8}$s
  - 30 additions effectuées avant que la lumière arrive à l'écran
- Age de l'univers : 15 milliard $\times$ 365 $\times$ 24 $\times$ 3600 $\approx$ $5.10^{17}s$
- Nombres d'électrons dans l'univers : $\approx 10^{64}$
- Complexité d'un algorithme pour des nombres de 128 bits.

| Complexité | $t_n$ | $t_n^2$ | $t_n^3$ | $t_n^4$ | $n$ |
|---|---|---|---|---|---|
| Nb d'ops | 128 | 16 384 | $2.10^6$ | $3.10^8$ | $10^{39}$ |

$\Rightarrow n = 10^{39}$ $\Rightarrow$$2.10^{12}$ fois l'âge de l'univers sur un PC à 1GHz!

# Arithmétique et complexité: $\mathbb{Z}$

## Précision fixée 32 ou 64 bits

$[0 \ldots 2^{32} - 1]$ ou $[-2^{31} + 1 \ldots 2^{31} - 1]$

- coût atomique: add,mul, sub: 1 cycle; div: $\approx$ 10 cycles
- petits entiers; petits corps/anneaux finis, ....

# Arithmétique et complexité: $\mathbb{Z}$

## Précision fixée 32 ou 64 bits

$[0 \ldots 2^{32} - 1]$ ou $[-2^{31} + 1 \ldots 2^{31} - 1]$

- coût atomique: add,mul, sub: 1 cycle; div: $\approx$ 10 cycles
- petits entiers; petits corps/anneaux finis, ....

## Multi-précision

En C/C++, GMP/MPIR: liste de mots non signés de 32 bits

addition: $\mathcal{O}(t_n)$

Multiplication : selon la taille

- $n < 32$ mots : Classique $\Rightarrow \mathcal{O}\left(t_n^2\right)$
- $32 < n < 256$ : Karatsuba $\Rightarrow \mathcal{O}\left(t_n^{1.585}\right)$
- Toom-Cook $\Rightarrow \mathcal{O}\left(t_n^{1.465}\right)$
- $n > 10000$ mots : FFT $\Rightarrow \mathcal{O}(t_n \log t_n \log\log t_n) = \tilde{\mathcal{O}}(t_n)$

# Arithmétique et complexité: $\mathbb{Z}/n\mathbb{Z}$

- $a \equiv b[m]$ : $a = b$ “modulo” $m$
- $\mathbb{Z}/n\mathbb{Z} = 0, 1, \ldots, n-1$

Opérations dans $\mathbb{Z}/n\mathbb{Z}$:

Addition:

```
c = a + b;
if (c > n) c = c - n;
```

Multiplication:

```
c = a * b;
c = c % n;
```

Division: cf Algorithme d’Euclide étendu

# Algorithme d'Euclide

**begin**
  $r_0 = a$;
  $r_1 = b$;
  **while** $r_i \neq 0$ **do**
    Calculer $r_{i+1}$ tq $r_{i-1} = r_i q_i + r_{i+1}$;
    $i = i + 1$;
**end**

- le dernier $r_i \neq 0$ est le pgcd de $a$ et $b$

# Algorithme d'Euclide

**begin**
$r_0 = a$;
$r_1 = b$;
$u_0 = 1, v_0 = 0$;
$u_1 = 0, v_1 = 1$;
**while** $r_i \neq 0$ **do**
Calculer $r_{i+1}$ tq $r_{i-1} = r_i q_i + r_{i+1}$;
$u_{i+1} = u_{i-1} - q_i u_i$;
$v_{i+1} = v_{i-1} - q_i v_i$;
$i = i + 1$;
**end**

- le dernier $r_i \neq 0$ est le pgcd de $a$ et $b$
- invariant $u_i a + v_i b = r_i$ for all $i$ $\Rightarrow$coefficients de Bezout

# Algorithme d'Euclide

$n = \max(t_a, t_b)$

- Algorithme classique : $\mathcal{O}\left(n^2\right)$
- Algorithme Half-GCD: $\mathcal{O}\left(n \log^2 n\right)$

# Algorithme d'Euclide

$n = \max(t_a, t_b)$

- Algorithme classique : $\mathcal{O}\left(n^2\right)$
- Algorithme Half-GCD: $\mathcal{O}\left(n \log^2 n\right)$

**Calcul de l'inverse modulaire $a^{-1} \mod n$**

Si $a \wedge b = 1$, alors $u_i a + v_i n = 1$
$\Rightarrow ua = 1 \mod n$
$\Rightarrow a^{-1} = u \mod n$.

**Corollaire**

*$\mathbb{Z}/p\mathbb{Z}$ est un corps ssi $p$ est premier*

# Plan

# Les grands types d'attaques

| | |
|---|---|
| Passives: | écoute sur le canal $\Rightarrow$confidentialité compromise |
| Actives: | modif. du message $\Rightarrow$menace l'intégrité, l'authenticité |

# Les grands types d'attaques

**Passives:** écoute sur le canal $\Rightarrow$confidentialité compromise

**Actives:** modif. du message $\Rightarrow$menace l'intégrité, l'authenticité

**Chiffré connu:** Oscar ne connaît que $C$

**Clair connu:** Oscar connaît $C$ et $M$

**Clair choisi:** Oscar choisit $M$ et obtient $C$

**Chiffré choisi:** Oscar choisit $C$ et obtient $M$

# Les grands types d'attaques

**Passives:** écoute sur le canal $\Rightarrow$confidentialité compromise

**Actives:** modif. du message $\Rightarrow$menace l'intégrité, l'authenticité

**Chiffré connu:** Oscar ne connaît que $C$

**Clair connu:** Oscar connaît $C$ et $M$

**Clair choisi:** Oscar choisit $M$ et obtient $C$

**Chiffré choisi:** Oscar choisit $C$ et obtient $M$

Algorithmes d'attaque:

**Attaque par force brute:** exploration de l'espace des clés

**Attaque par séquence connue:** une partie du message est connue (e.g. entête d'un e-mail,...)

**Attaque par séquence forcée:** forcer Alice à encoder une séquence connue

**Attaque par analyse différentielle:** faible différence des clairs

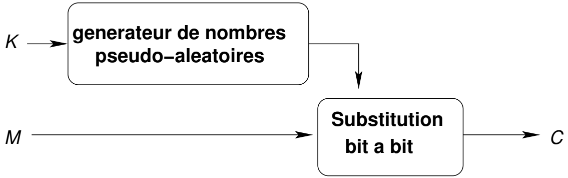
# Plan

# Entropie et théorie de l’information

# Plan

# Plan

# Chiffrement symétrique par flot



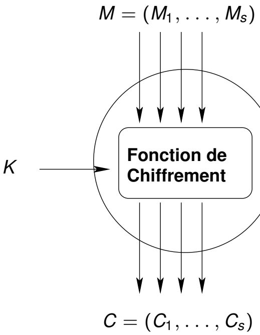

- traitement à la volée
- basé sur:
  - un **générateur pseudo-aléatoire** (e.g. LFSR)
  - un **mécanisme de substitution** bit à bit (e.g. XOR)

# Linear Feedback Shift Register: LFSR

# Plan

# Chiffrement par bloc

- $M = (M_1, \ldots, M_s)$ $s$ blocs de $n/s$ bits
- $C(C_1, \ldots, C_s)$ $s$ blocs de $n/s$ bits

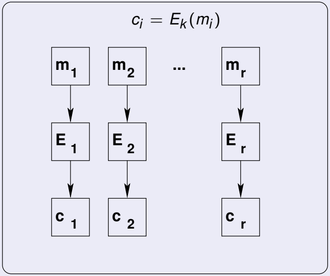



Deux niveaux:

- fonction de chiffrement $E$
- mode de chiffrement (*agencement* entre les blocs)

# Modes de chiffrement

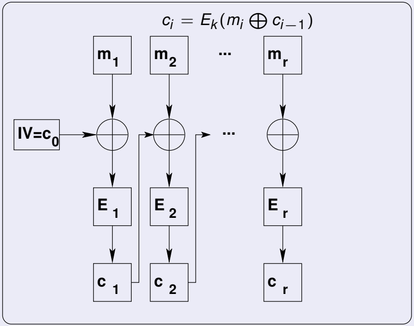
## ECB: Electronic Code Book




- un bloc est toujours codé identiquement
- aucune sécurité, pas utilisé

# Modes de chiffrement

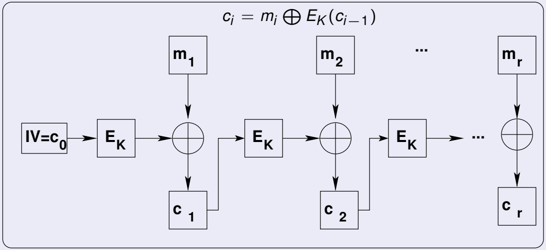
## CBC: Cipher Bloc Chaining




- $C_i = E_k(M_i \oplus C_{i-1})$
- $M_i = C_{i-1} \oplus D_k(C_i)$
- Le plus utilisé
- nécessite $D_K = E_k^{-1}$

# Modes de chiffrement

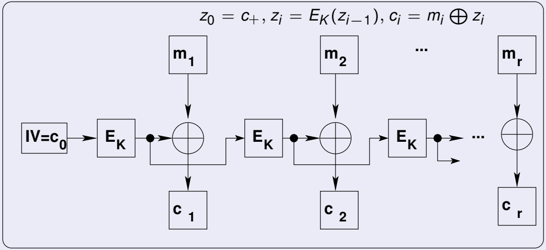
## CFB: Cipher Feedback




- $C_i = M_i \bigoplus E_K(C_{i-1})$
- $M_i = C_i \bigoplus E_k(C_{i-1})$
- ne nécessite pas $D_K = E_k^{-1}$
- moins sûr

# Modes de chiffrement

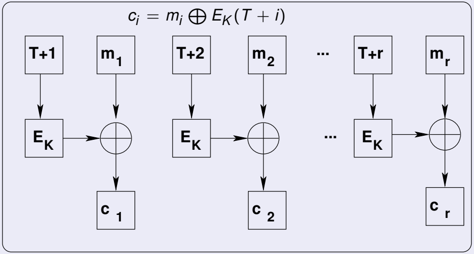
## OFB: Output Feedback




- variante totalement symétrique
- Encodage: $z_i = E_K(z_{i-1}); c_i = m_i \bigoplus z_i$
- Décodage: $z_i = E_K(z_{i-1}); m_i = c_i \bigoplus z_i$
- $\Rightarrow$moins de circuits (pour syst. embarqués, e.g. satellites)

# Modes de chiffrement

## CTR: Counter-mode encryption




- Totalement symétrique
- Introduit un compteur de valeur initiale
- Décodage: $m_i = c_i \bigoplus E_K(T + i)$
- calculs indépendants $\Rightarrow$parallélisation
- comme ECB, mais un même bloc n'est jamais codé pareil

# Plan

# DES: Data Encryption Standard

- 1977: Le NIST demande à IBM un standard de cryptographie industriel
- IBM dispose de *Lucifer* conçu en 1971 par H. Feistel
- puis modifié par la NSA (rendu plus robuste ?)

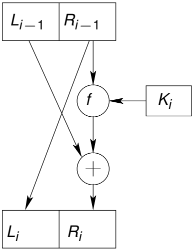
Structure:

- Blocs de taille 64 bits
- clé de 56 bits (64 avec 1 bit de parité par octet)

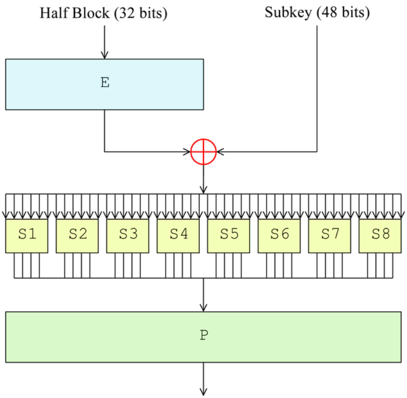
# DES: Data Encryption Standard

Structure générale:

1. Permutation initial IP:
   $(L_0, R_0) = IP(x)$
2. 16 tours de chiffrements (rondes):
   $\begin{cases} L_i & = & R_{i-1} \\ R_i & = & L_{i-1} \bigoplus f(R_{i-1}, K_i) \end{cases}$ où
   - $K_i$ est une clé de 48 bits dérivée de $K$ (**diversification**)
   - $f$ est la **fonction de Feistel**
3. $c = \mathrm{IP}^{-1}(R_{16}, L_{16})$



# Fonction de Feistel




- $E$: fonction d'expansion 32 $\longrightarrow$ 48 bits.
- $S_i$: boîte-S (SBox)
- $P$: permutation

# Diversification de la clé

- Permutation initiale des 56 bits de la clé: $(C_0, D_0) = P(K)$
- 16 rondes: $\left\{ \begin{array}{lcl} C_i & = & \mathrm{LS}_i(C_{i-1}) \\ D_i & = & \mathrm{LS}_i(D_{i-1}) \\ K_i & = & P_2(C_i, D_i) \end{array} \right.$ où $\mathrm{LS}_i$ est une rotation circulaire vers la gauche d'une ou deux position selon $i$, $P_2$, un permutation.

# Propriétés du DES

- après 5 rondes: chaque bit du chiffré dépend de chaque bit du clair
- après 16 rondes: statistiquement *plat* (analyse freq. inopérante)
- légère modification dans $K$ ou $M$ $\Rightarrow$forte modification dans $C$

  Avantages:
  - Implantation matérielle économique
  - Haut débit: 190Mo/s avec une puce bas de gamme

  Implanté dans
  - payment par cartes de crédit (UEPS)
  - protocoles d'authentification (Kerbeos)
  - messagerie électronique (PEM: Privacy enhanced Mail)

# Exercices

1. Montrer qu'il n'est pas nécessaire d'inverser $f$ pour déchiffrer une ronde de DES. En déduire l'algorithme de décodage.
2. Confientialité du DES
   - On rappelle qu'un chiffrement parfait vérifie $|K| > |M|$
   - Discuter de la confidentialité du DES pour un bloc

# Cryptanalyse de DES

Recherche exhaustive: (force brutale)

- A appliquer sur un texte clair connu
- Tester les $2^{56}$ clés, et retrouver le chiffré

Pré-calcul exhaustif:

- Stocker les chiffrés d'un texte choisi
- Identifier des fragments du chiffré pour déduire la clé

# Cryptanalyse de DES

Recherche exhaustive: (force brutale)

- A appliquer sur un texte clair connu
- Tester les $2^{56}$ clés, et retrouver le chiffré

Pré-calcul exhaustif:

- Stocker les chiffrés d'un texte choisi
- Identifier des fragments du chiffré pour déduire la clé

Cryptanalyse différentielle: [Biham Shamir 1990]

- Différence de cryptage entre deux textes similaires

  $\Rightarrow$information sur la clé probable

Cryptanalyse linéaire: [Matsui 1993]

- Relations linéaires sur certains bits du chiffré, du clair

  $\Rightarrow$interpolation de certains bits de la clé

# Cryptanalyse de DES

## Étude en 1996 sur le coût des attaques sur DES

| **Attaquant** | **Budget** | **Outil** | **Temps** |
|---|---|---|---|
| Hacker | 300$ | Logiciel | 38 ans |
| PME | 7 500$ | FPGA | 18 mois |
| Gde Entreprise | 225 K$ | ASIC | 19j, 3h |
| Multinationale | 7,5 M$ | ASIC | 6 min. |
| Gouvernement | 225 M$ | ASIC | 12s |

$\Rightarrow$ force brutale n'est plus impossible

Exemple:

- $2^{56} < 10^{17}$ clés à tester
- 1000 PC à 1GHz $\Rightarrow 10^{12}$ op/s

$\Rightarrow 10^5\text{s} \approx 27,7\text{h}$

# double-DES

Augmenter l'espace des clés.

Idée: Composer deux DES successifs

$$E(M) = E_2(E_1(M)), D(M) = D_1(D_2(C)).$$

# double-DES

Augmenter l'espace des clés.

Idée: Composer deux DES successifs

$$E(M) = E_2(E_1(M)), D(M) = D_1(D_2(C)).$$

**Attaque *Meet in the middle***

Connaissant $M$ et $C$, trouve $K$.

1. Calculer les $2^{56}$ chiffrés $X_i = E_1(M, K_i)$ — $2^{56}$
2. Trier les $(X_i)_i$ — $56 \times 2^{56}$
3. Calculer les $2^{56}$ décodage $Y_j = D_2(C, K_j)$ — $2^{56}$
4. Comparer les $Y_j$ avec les $X_i$ — $56 \times 2^{56}$
5. Quand $X_{i_0} = Y_{j_0}$ $\Rightarrow K_1 = K_{i_0}, K_2 = K_{j_0}$.

$\Rightarrow$seulement $112 = 2^{6,8}$ fois plus difficile (et non $2^{56}$).

$\Rightarrow$clé équivalente de $\approx 63$ bits

# triple-DES

Idée: Composer trois DES successifs

- DES-EEE:
  $E(M) = E_3(E_2(E_1(M))), D(M) = D_1(D_2(D_3(C)))$.
  $\Rightarrow$120 bits
- DES-EDE:
  $E(M) = E_1(D_2(E_1(M))), D(M) = D_1(E_2(D_1(C)))$.
  $\Rightarrow$112 bits
  $\Rightarrow$DES-EDE=DES si $K_1 = K_2$
  $\Rightarrow$clé moins longue est sécurité effective équivalente

# Plan

# AES: Advanced Encryption Standard

Nouveau standard créé par le NIST:

- 1997: Appel à candidatures
- 5 finalistes: Rijndael, Serpent, Twofish, RC6, Mars
- 2000: Selection de Rijndael

| | | Cycle/octet | |
|---|---|---|---|
| Algorithme | Taille de clé | Codage | Décodage |
| DES | 56 | 59 | 59 |
| 3-DES | 112 | 154 | 155 |
| IDEA | 128 | 56 | 56 |
| Grand Cru | 128 | 1250 | 1250 |
| RC6 | 256 | 18 | 17 |
| Rijndael | 256 | 34 | 35 |
| Serpent | 256 | 68 | 80 |
| Mars | 256 | 31 | 30 |
| Twofish | 256 | 29 | 25 |

# AES

- Opération de base: sur les octets $\mathbb{F}_{256} \equiv \mathbb{Z}_2[X]/(X^8 + X^4 + X^3 + X + 1)$.
- Représentation matricielle: $32 \times k$ bits = $4 \times k$ octets.

$$k = 4 : [a_0, a_1, a_2, \ldots, a_{16}] \to \begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{bmatrix} \ldots$$

où $a_0, a_1, \ldots$ sont des octets

Exemple:

- clé de 128 bits $\Rightarrow$matrice $4 \times 4$,
- clé de 192 bits $\Rightarrow$matrice $4 \times 6$,
- clé de 256 bits $\Rightarrow$matrice $4 \times 8$,

# AES

$N_k$: nb de colonnes de la matrice de la clé $N_b$: nb de colonnes de la matrice d'un bloc $N_r$: nb de rondes

| | $N_b$ | $N_k$ | $N_r$ |
|---|---|---|---|
| AES-128 | 4 | 4 | 10 |
| AES-192 | 4 | 6 | 12 |
| AES-256 | 4 | 8 | 14 |

# AES

$N_k$: nb de colonnes de la matrice de la clé $N_b$: nb de colonnes de la matrice d'un bloc $N_r$: nb de rondes

| | $N_b$ | $N_k$ | $N_r$ |
|---|---|---|---|
| AES-128 | 4 | 4 | 10 |
| AES-192 | 4 | 6 | 12 |
| AES-256 | 4 | 8 | 14 |

Opérations d'une ronde:

`SubBytes`: substitution non linéaire (SBox)

`ShiftRows`: Rotation de chaque $L_i$

`MixColumns`: Multiplication à gauche par une matrice

`AddRoundKey`: Addition bit à bit avec une clé de ronde $K_i$ obtenue par diversification

- Initialisation: `AddRoundKey`
- Ronde finale: identique, sans `Mixcolumns`

# AES

## Fonction `SubBytes`

- Pour chaque élément de la matrices $a_{i,j} \leftarrow f(a_{i,j})$
- $f$ non linéaire: $f(x) = Ax^{-1} + B$ dans $\mathbb{F}_{256}$, avec $A = X^7 + X^6 + X^5 + X^4 + 1, B = X^6 + X^5 + X + 1$.

## Fonction `ShiftRows`

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \longrightarrow \begin{bmatrix} a & b & c & d \\ f & g & h & e \\ k & l & i & j \\ p & m & n & o \end{bmatrix}$$

# AES

## Fonction `MixColumns`

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \leftarrow \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Point de vue algébrique:

- Chaque colonne est un polynôme de degré 4 de $\mathbb{F}_{256}[Y]$
- Multiplication par $[03]Y^3 + Y^2 + Y + [02] \mod Y^4 + 1$

Remarque: matrice $4 \times 8$ dans $\mathbb{F}_2$

$\Rightarrow$redondance $\Rightarrow$code correcteur d'erreur cyclique

- Détection jusqu'à 2 octets erronés
- Correction jusqu'à 1 octet

# AES

## `AddRoundKey`

- Addition bit à bit avec la clé $K_i$
- obtenue par diversification:

## Diversification de la clé

$K$: $4N_k$ octets $\longrightarrow$ $W$ :$4N_b(N_r+1)$ octets
$\Rightarrow N_r+1$ clé de ronde.

- $W_{1\ldots N_k} = K$ (les $N_k$ colonnes de $K$ sont copiés dans $W$)
- $W_i$ déduite de $W_{i-1}$ par rotations, SBox, et ajout de constantes

# Diversification de la clé (fin)

**begin**

$K_0 = [W_0|W_1|W_2|W_3] = K$;

**for** *i=1...10* **do**

$T = W_{4i-1} <<< 8$;

`/* rotation de 8 bits à gauche */`

$T = \texttt{SubBytes}(T)$;

`/* 4 applications de la SBox */`

$T = T \bigoplus (X^i \mod P_8)$;

$W_{4i} = W_{4i-4} \bigoplus T$;

$W_{4i+1} = W_{4i-3} \bigoplus W_{4i}$;

$W_{4i+2} = W_{4i-2} \bigoplus W_{4i+1}$;

$W_{4i+3} = W_{4i-1} \bigoplus W_{4i+2}$;

$K_i = [W_{4i}|W_{4i+1}|W_{4i+2}|W_{4i+3}]$;

**end**

# AES: récapitulatif

Cryptage:

```
begin
    A=M
    A=AddRoundKey(A, K_0)
    for i=1...9 do
        A=SubBytes(A)
        A=ShiftRows(A)
        A=MixColumns(A)
        A=AddRoundKey(A, K_i)
    A=SubBytes(A)
    A=ShiftRows(A)
    A=AddRoundKey(A, K_10)
    return C = A
end
```

Décryptage:

```
begin
    A=C
    A=AddRoundKey(A, K_10)
    A=InverseShiftRows(A)
    A=InverseSubBytes(A)
    for i=1...9 do
        A=AddRoundKey(A, K_i)
        A=InverseMixColumns(A)
        A=InverseShiftRows(A)
        A=InverseSubBytes(A)
    A=AddRoundKey(A, K_0)
    return M = A
end
```

# Sécurité de l'AES

- Propriété cryptanalytiques:
  - SBox: sans point fixe, ni opposé, ni inverse
  - `ShiftRows` diffuse les données en séparant les consécutifs
  - `MixColumns` chaque bit de sortie dépend de tous les bits d'entrée
- Rapide: implémentation FPGA $\Rightarrow$cryptage à 21.54 Go/s [Hodjat-Verbauwhede 2004]
- Sécurité:
  - Aucune attaque significative
  - Attaque théorique en $\mathcal{O}\left(2^{100}\right)$ sur AES-128 [Courtois,Pieprzyk, 2002, 05]
  - $2^{45}$ pour AES-256, sur 10 rondes (AES-256 simplifié) [Schneier 09]
  - ...

# Applications utilisant AES

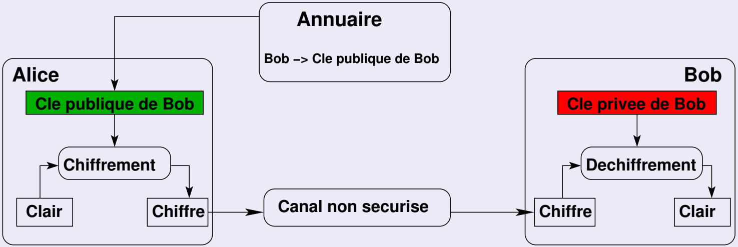
- OpenSSL
- SONET (Synchronus Optical NETwork)
- Routeurs internet
- Communications Satellites
- VPN
- Téléphones mobiles
- Transaction électroniques
- ...

# Plan

# Introduction

Rôle *asymétrique* des clés de cryptage/décryptage

## Communication chiffrée à clé publique



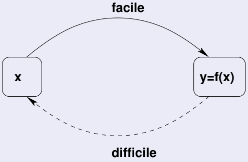

Analogie de la **Boîte aux lettres:**

- Toute personne peut écrire à Bob
- Seul Bob peut lire le courrier

# Fonctions à sens unique

- La clé privé $K_d$ est totalement déterminée par la clé publique $K_e$
  $\Rightarrow H(K_d|K_e) = 0$
- Ajouter la notion de *difficulté de calcul*

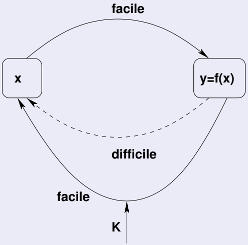
## Fonctions à sens unique



# Fonctions à sens unique

- La clé privé $K_d$ est totalement déterminée par la clé publique $K_e$
  $\Rightarrow H(K_d|K_e) = 0$
- Ajouter la notion de *difficulté de calcul*

## Fonctions à sens unique



# Plan

# Le théorème des restes chinois

## Théorème

*Si $m_1, \ldots, m_k$ sont premiers deux à deux,*

$$\mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z} \equiv \mathbb{Z}/(m_1 \ldots m_k)\mathbb{Z}.$$

Isomorphisme:

$$\begin{array}{rrcl} \varphi : & \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z} & \rightarrow & \mathbb{Z}/(m_1 \ldots m_k)\mathbb{Z} \\ & (x_1, \ldots, x_k) & \mapsto & \sum_{i=1}^{k} x_i \Pi_i Y_i \mod \Pi \end{array}$$

$$\text{où} \left\{ \begin{array}{lcl} \Pi & = & \prod_{i=1}^{k} m_i \\ \Pi_i & = & \Pi / m_i \\ Y_i & = & \Pi_i^{-1} \mod m_i \end{array} \right.$$

## Théorème (Autre formulation)

*Si $m_1, \ldots, m_k$ sont premiers deux à deux et $a_1, \ldots, a_k$ sont des résidus modulo resp. $m_1, \ldots, m_k$. Alors $\exists! A \in \mathbb{Z}_+, A < \prod_{i=1}^{k} m_i$, tel que $A = a_i[m_i]$ pour $i = 1 \ldots k$.*

# Analogie avec les polynômes

Évaluer $P$ en $a$ $\quad\leftrightarrow\quad$ Reduire $P$ modulo $X - a$

| **Polynômes** | **Entiers** |
|---|---|
| **Evaluation:** $P \mod X - a$ Évaluer $P$ en $a$ | $N \mod m$ "Évaluer" $N$ en $m$ |
| **Interpolation:** $P = \sum_{i=1}^{k} \frac{\prod_{j\neq i}(X-a_j)}{\prod_{j\neq i}(a_i-a_j)}$ | $N = \sum_{i=1}^{k} a_i \prod_{j\neq i} m_j (\prod_{j\neq i} m_j)^{-1[m_i]}$ |

# Indicatrice d'Euler

## Definition

- *Sous groupe multiplicatif de* $\mathbb{Z}/n\mathbb{Z}$*:*
  $(\mathbb{Z}/n\mathbb{Z})^* = \{x \in \mathbb{Z}/n\mathbb{Z}, x \wedge n = 1\}$
- *Indicatrice d'Euler:* $\varphi(n) = \#(\mathbb{Z}/n\mathbb{Z})^*$

## Propriété

- $p$ *premier* $\Rightarrow \varphi(p) = (p-1), \varphi(p^k) = (p-1)p^{k-1}$
- $m \wedge n = 1 \Rightarrow \varphi(mn) = \varphi(m)\varphi(n)$

Exemple: $n = \prod_{i=1}^{k} p_i^{\alpha_i}$ (décomposition en facteurs premiers)

$$\varphi(n) = \prod_{i=1}^{k} p_i^{\alpha_i - 1}(p_i - 1)$$

# Euler, Fermat

**Théorème (Euler)**

*Si* $a \wedge n = 1$*, alors* $a^{\varphi(n)} = 1 \mod n$.

**Théorème (Fermat)**

*Si* $p$ *est premier, alors* $a^{p-1} = 1 \mod p \; \forall a \in \mathbb{Z}/p\mathbb{Z}$.

# Logarithme discret

## Definition

*Un groupe est cyclique s'il est fini et monogène.*

## Théorème

$(\mathbb{Z}/n\mathbb{Z})^*$ *est cylcique si et seulement si* $n = 2, 4, p^\alpha, 2p^\alpha$, *avec* $p$ *premier impair.*

Soit $G = \{g^k, k \in \mathbb{Z}\}$ cyclique de taille $n$, engendré par $g$: La fonction $\begin{array}{cccc} E_g : & \{0 \ldots n\} & \to & G \\ & k & \mapsto & g^k \end{array}$ est une bijection.

On définit le logarithme discret $\log_g = E_g^{-1}$.

## Definition

*Pour tout* $x \in G \; \exists! n \in \{0 \ldots n-1\}, x = g^n$.
*L'entier* $n = \log_g x$ *est le logarithme discret de* $x$ *en base* $g$.

# Racines primitives

## Definition

*Tout générateur $g$ du sous-groupe multiplicatif d'un anneau $A$ est appelé racine primitive de $A$.*

## Propriété

*Pour $p$ premier, il y a $\varphi(p-1)$ racines primitives dans $\mathbb{Z}/p\mathbb{Z}$.*

## Théorème

*Si $g$ est une racine primitive de $\mathbb{Z}/n\mathbb{Z}$, alors*

$$g^x = g^y \ [n] \Leftrightarrow x = y \ [\varphi(n)]$$

# Plan

# Une fonction à sens unique: l'exponentiation modulaire

## Problème

**Données**: $n, m, x$
**Résultat**: $y = x^n \mod m$

**begin**
  $y = 1$
  **for** $i = 1 \ldots n$ **do**
    $y = y \times x \mod m$
**end**

$\Rightarrow \mathcal{O}(n)$ opérations dans $\mathbb{Z}/m\mathbb{Z}$

# Une fonction à sens unique: l'exponentiation modulaire

## Problème

**Données**: $n, m, x$
**Résultat**: $y = x^n \mod m$

**begin**
  **if** $n$=0 **then**
    **return** 1
  Calculer récursiv. $z = X^{\lfloor n/2 \rfloor}$
  **if** *n est pair* **then**
    **return** $z^2 \mod m$
  **else**
    **return** $z^2 \times x \mod m$
**end**

$\Rightarrow \mathcal{O}(\log_2 n)$ opérations dans $\mathbb{Z}/n\mathbb{Z}$

**begin**
  $y = 1$
  **for** $i = 1 \ldots n$ **do**
    $y = y \times x \mod m$
**end**

$\Rightarrow \mathcal{O}(n)$ opérations dans $\mathbb{Z}/m\mathbb{Z}$

# Une fonction à sens unique: l'exponentiation modulaire

**Problème**

**Données**: $n, m, x$
**Résultat**: $y = x^n \mod m$

```
begin
    if n=0 then
        return 1
    Calculer récursiv. z = X^⌊n/2⌋
    if n est pair then
        return z^2 mod m
    else
        return z^2 × x mod m
end
```

$\Rightarrow \mathcal{O}(\log_2 n)$ opérations dans $\mathbb{Z}/n\mathbb{Z}$

```
begin
    y = 1
    for i = 1 ... n do
        y = y × x mod m
end
```

$\Rightarrow \mathcal{O}(n)$ opérations dans $\mathbb{Z}/m\mathbb{Z}$

```
begin
    Soit n = [n_0, ..., n_{log_2 n}]
    l'écriture binaire de n
    h = x, y = 1
    for i = 1 ... log_2 n do
        if n_i = 1 then
            y = y × h mod m
        h = h^2 mod m
    return y
end
```

# Réciproque: le logarithme discret

### Problème

*Connaissant $y, x, m$, calculer $n$ tel que $y = x^n\ [m]$*

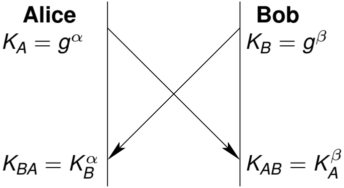
- Enumération de tous les $x^i$, pour $i = 1 \dots m$ $\Rightarrow \mathcal{O}(m)$
- Amélioration: BabyStep/GiantStep, Pollard $\Rightarrow \mathcal{O}\left(\sqrt{m}\right)$
- Meilleure complexité connue: $\mathcal{O}\left(m^{1/3}\right)$

### Fonction à sens unique

- $\mathrm{Exp}_x$: facile (complexité logarithmique)
- $\log_x$: difficile (complexité polynomiale)

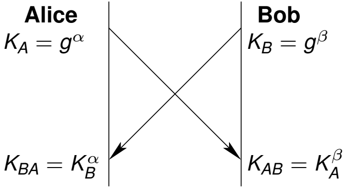
# Protocole d'échange de clés de Diffie Hellman

1. Choisir publiquement un groupe $(\mathbb{Z}/p\mathbb{Z})^*$ et un générateur $g$
2. Alice:
   - choisit un secret $\alpha \in (\mathbb{Z}/p\mathbb{Z})^*$
   - calcule $K_A = g^\alpha \mod p$
   - envoie $K_A$ à Bob
   - calcule $K_{BA} = K_B^\alpha \mod p$
3. Bob:
   - choisit un secret $\beta \in (\mathbb{Z}/p\mathbb{Z})^*$
   - calcule $K_B = g^\beta \mod p$
   - envoie $K_B$ à Alice
   - calcule $K_{AB} = K_A^\beta \mod p$



# Protocole d'échange de clés de Diffie Hellman

1. Choisir publiquement un groupe $(\mathbb{Z}/p\mathbb{Z})^*$ et un générateur $g$
2. Alice:
   - choisit un secret $\alpha \in (\mathbb{Z}/p\mathbb{Z})^*$
   - calcule $K_A = g^\alpha \mod p$
   - envoie $K_A$ à Bob
   - calcule $K_{BA} = K_B^\alpha \mod p$
3. Bob:
   - choisit un secret $\beta \in (\mathbb{Z}/p\mathbb{Z})^*$
   - calcule $K_B = g^\beta \mod p$
   - envoie $K_B$ à Alice
   - calcule $K_{AB} = K_A^\beta \mod p$




### Propriété

$$K_{AB} = K_{BA} = g^{\alpha\beta}$$

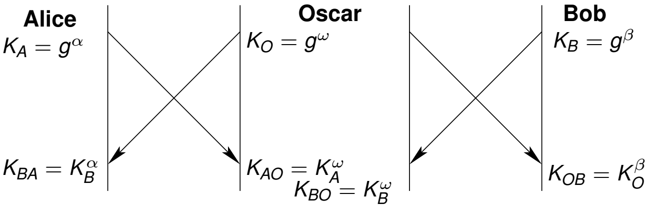
$\Rightarrow$partage la clé sans jamais transmettre $\alpha, \beta$.

# Sécurité du protocole de Diffie-Hellmann

- A partir de $g, p, K_A, K_B$, Oscar a besoin de calculer un logarithme discret pour connaître $\alpha, \beta$.

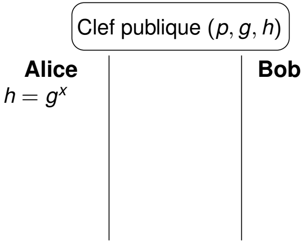
# Sécurité du protocole de Diffie-Hellmann

- A partir de $g, p, K_A, K_B$, Oscar a besoin de calculer un logarithme discret pour connaître $\alpha, \beta$.
- Faiblesse: attaque par *l'homme du milieu*



# Un code à clé publique: El Gamal

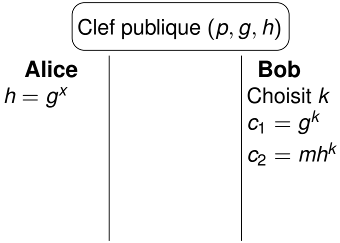
1. Alice:
   - choisit $(\mathbb{Z}/p\mathbb{Z})^*$ et un générateur $g$
   - choisit un secret $x$
   - calcule $h = g^x \mod p$
   - Clé publique: $(p, g, h)$



# Un code à clé publique: El Gamal

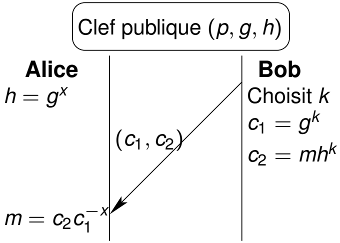
1. Alice:
   - choisit $(\mathbb{Z}/p\mathbb{Z})^*$ et un générateur $g$
   - choisit un secret $x$
   - calcule $h = g^x \mod p$
   - Clé publique: $(p, g, h)$

2. Bob: veut envoyer le message $m \in \mathbb{Z}/p\mathbb{Z}$
   - choisit aléatoirement $k$
   - calcule $c_1 = g^k \mod p$
   - calcule $c_2 = mh^k \mod p$
   - envoie $(c_1, c_2)$ à Alice



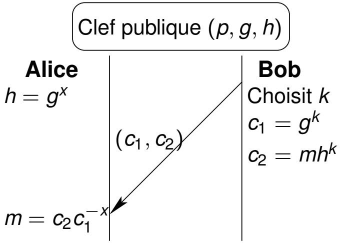
# Un code à clé publique: El Gamal

1. Alice:
   - choisit $(\mathbb{Z}/p\mathbb{Z})^*$ et un générateur $g$
   - choisit un secret $x$
   - calcule $h = g^x \mod p$
   - Clé publique: $(p, g, h)$
   - calcule $c_2 c_1^{-x} \mod p$
2. Bob: veut envoyer le message $m \in \mathbb{Z}/p\mathbb{Z}$
   - choisit aléatoirement $k$
   - calcule $c_1 = g^k \mod p$
   - calcule $c_2 = mh^k \mod p$
   - envoie $(c_1, c_2)$ à Alice



# Un code à clé publique: El Gamal

1. Alice:
   - choisit $(\mathbb{Z}/p\mathbb{Z})^*$ et un générateur $g$
   - choisit un secret $x$
   - calcule $h = g^x \mod p$
   - Clé publique: $(p, g, h)$
   - calcule $c_2 c_1^{-x} \mod p$
2. Bob: veut envoyer le message $m \in \mathbb{Z}/p\mathbb{Z}$
   - choisit aléatoirement $k$
   - calcule $c_1 = g^k \mod p$
   - calcule $c_2 = mh^k \mod p$
   - envoie $(c_1, c_2)$ à Alice




$$c_2 c_1^{-x} = mh^{-x} g^{kx} = mg^{-kx} g^{kx} = m \mod p$$

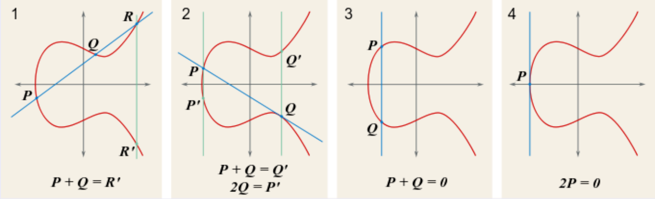
# Sécurité d'El Gamal

- Trouver le clair $\Rightarrow$casser le logarithme discret
- Généralisation: sur tout groupe fini

## Groupe de courbes elliptiques

$(E) : y^2 = ax^3 + b \mod p$

- Structure de groupe de points sur $(E)$.




- Clé publique: $(E, p, P)$, où $P$ est un point sur $(E)$.
- Alice: $h = xP$
- Bob: $(c_1, c_2) = (kP, m + kh)$

# Plan

# Le cryptosystème RSA: *Rivest Shamir Adelman*

**Théorème (Rivest Shamir Adelman 78)**

*Soit $p, q$ premiers, et $n = pq$. Alors*

$$\forall a \in \mathbb{Z}/n\mathbb{Z}\ a^{1+k(p-1)(q-1)} = a \mod n$$

**Preuve:**

- Si $a = 0 \mod p$, alors $a^{k\varphi(n)+1} = 0 = a \mod p$
- Sinon, $a$ est inversible modulo $p$ et $a^{p-1} = 1 \mod p$. Donc

$$a^{k(p-1)(q-1)+1} = a \mod p$$

- Idem $\mod q$: $a^{k(p-1)(q-1)+1} = a \mod q$
- Restes Chinois: $a^{k(p-1)(q-1)+1} = a \mod pq$

# Le code RSA

- Soit $e \in (\mathbb{Z}/n\mathbb{Z})^*$: $e \wedge \varphi(n) = 1 \quad \Rightarrow ed - k\varphi(n) = 1$

$$ed = 1 + k\varphi(n)$$

**Le code RSA:**

- Clé publique: $(e, n)$: encodage: $c = m^e \mod n$
- Clé privée: $(d, n)$: décodage: $c^d \mod n$

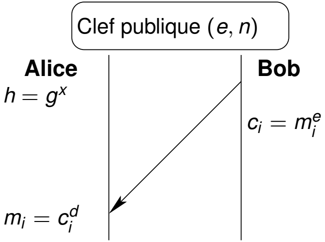
Théorème RSA: $c^d = m^{de} = m^{1+k\varphi(n)} = m \mod n$

**Attention:**

- $\varphi(n)$ doit rester privé
- $p, q$ doivent rester privés

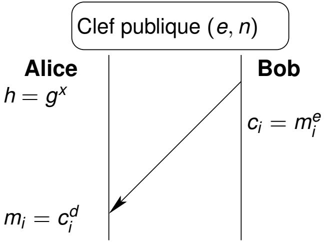
# Le code RSA

1. Alice:
   - choisit $p, q$ et calcule $n = pq$
   - choisit $e$ premier avec $\varphi(n)$
   - calcule $d = e^{-1} \mod \varphi(n)$
   - Clé publique: $(e, n)$



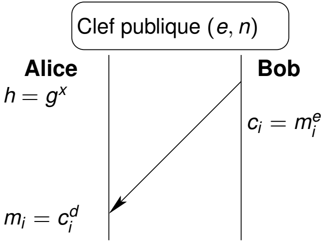
# Le code RSA

1. Alice:
   - choisit $p, q$ et calcule $n = pq$
   - choisit $e$ premier avec $\varphi(n)$
   - calcule $d = e^{-1} \mod \varphi(n)$
   - Clé publique: $(e, n)$
2. Bob: veut envoyer le message $m$
   - découpe $m = (m_1, \ldots, m_k)$ tq $m_i < n$
   - Pour chaque $m_i$
   - envoie $c_i = m_i^e \mod n$ à Alice



# Le code RSA

1. Alice:
   - choisit $p, q$ et calcule $n = pq$
   - choisit $e$ premier avec $\varphi(n)$
   - calcule $d = e^{-1} \mod \varphi(n)$
   - Clé publique: $(e, n)$
   - calcule $m_i = c_i^d \mod n$
2. Bob: veut envoyer le message $m$
   - découpe $m = (m_1, \ldots, m_k)$ tq $m_i < n$
   - Pour chaque $m_i$
   - envoie $c_i = m_i^e \mod n$ à Alice



# Algorithmique de RSA

- Calculer $p, q$ premiers $\Rightarrow$génération de premiers
- Calculer $n, \varphi(n) = (p-1)(q-1)$ $\Rightarrow$simple multiplication
- Calcul de $d = e^{-1} \mod \varphi(n)$ $\Rightarrow$Euclide Étendu
- Encodage: $c = m^e \mod n$ $\Rightarrow$exponentiation modulaire
- Décodage: $m = c^d \mod n$ $\Rightarrow$exponentiation modulaire

# Génération de nombre premiers

Principe:

- Tirer au hasard un entier
- Tester s'il est premier

## Tests de primalité

- Crible d'Eratosthène: **Déterministe, Exponentiel**
  - "Crible" tous les multiples de $2, 3, 5, 7, \ldots$ jusqu'à $\sqrt{n}$.
  - $\mathcal{O}(n)$ opérations $\Rightarrow$exponentiel
- [Miller Rabin]: **Probabiliste, Polynomial**
- [Agrawal, Kayal, Saxena 02]: **Déterministe, Polynomial**

# Test de primalité de Miller Rabin

Fermat: $p$ premier $\Rightarrow \forall a\ a^{p-1} = 1 \mod p$

- Tirer au hasard $a$
- Tester $a^{p-1} = 1 \mod p$

Idée: $p$ premier impair s'écrit $p = t2^s + 1$

$$\begin{aligned}
a^{p-1} - 1 &= (a^t)^{2^s} - 1 \\
&= ((a^t)^{2^{s-1}} - 1)((a^t)^{2^{s-1}} + 1) \\
&\dots \\
&= (a^t - 1)(a^t + 1)(a^{2t} + 1)\dots(a^{t2^{s-1}} + 1)
\end{aligned}$$

# Test de primalité de Miller Rabin

Fermat: $p$ premier $\Rightarrow \forall a\ a^{p-1} = 1 \mod p$

- Tirer au hasard $a$
- Tester $a^{p-1} = 1 \mod p$

Idée: $p$ premier impair s'écrit $p = t2^s + 1$

$$\begin{aligned} a^{p-1} - 1 &= (a^t)^{2^s} - 1 \\ &= ((a^t)^{2^{s-1}} - 1)((a^t)^{2^{s-1}} + 1) \\ &\dots \\ &= (a^t - 1)(a^t + 1)(a^{2t} + 1)\dots(a^{t2^{s-1}} + 1) \end{aligned}$$

$\Rightarrow$Si $n$ est premier:

- Soit $a^t = 1 \mod n$
- Soit $\exists i,\ a^{t2^i} = -1 \mod n$

# Miller Rabin

## Propriété

- *Si n est premier: le test indique premier*
- *Si n est composé: 3/4 des témoins indiquent composé*

*Tester k témoins:* $P[\textit{faux premier}] = (1/4)^k$

## Complexité

- 1 témoins: $\mathcal{O}\left(\log^3 n\right)$ ou $\tilde{\mathcal{O}}\left(\log^2 n\right)$
- $k$ témoins: $\mathcal{O}\left(k\log^3 n\right)$ ou $\tilde{\mathcal{O}}\left(k\log^2 n\right)$
- Sous GRH (hyp. de Riemmann Généralisée):
  $k = \mathcal{O}(\log n)$ suffit $\Rightarrow$polynomial, déterministe.

# Sécurité de RSA

## Difficulté de la factorisation de $n$

- Si on sait factoriser $n = pq$
  - On tire $\varphi(n) = (p-1)(q-1)$
  - On calcule $d = e^{-1} \mod \varphi(n)$
  - $m = c^d \mod n$

# Sécurité de RSA

## Difficulté de la factorisation de $n$

- Si on sait factoriser $n = pq$
  - On tire $\varphi(n) = (p-1)(q-1)$
  - On calcule $d = e^{-1} \mod \varphi(n)$
  - $m = c^d \mod n$
- Factorisation est difficile
- $\Rightarrow$Casser RSA *devrait* être difficile

# Sécurité de RSA

## Difficulté de la factorisation de $n$

- Si on sait factoriser $n = pq$
  - On tire $\varphi(n) = (p-1)(q-1)$
  - On calcule $d = e^{-1} \mod \varphi(n)$
  - $m = c^d \mod n$
- Factorisation est difficile
- $\Rightarrow$Casser RSA *devrait* être difficile
- Mieux: réductions “Casser RSA” $\Rightarrow$ “Factoriser”

# Réduction pour $e$ petit

- $ed = 1 \mod \varphi(n)$ $\Rightarrow ed - 1 = k\varphi(n)$
- $\varphi(n) = (p-1)(q-1) = pq + 1 - (p+q) = n + 1 - (p+q)$.
- Si $p, q \neq 2, 3$ $\Rightarrow p, q \leq n/4$
- Alors $k \leq 2e$

# Réduction pour $e$ petit

- $ed = 1 \mod \varphi(n) \quad \Rightarrow ed - 1 = k\varphi(n)$
- $\varphi(n) = (p-1)(q-1) = pq + 1 - (p+q) = n + 1 - (p+q)$.
- Si $p, q \neq 2, 3 \quad \Rightarrow p, q \leq n/4$
- Alors $k \leq 2e$

Si $e$ est petit: test exhaustif de tous les $k \in 0 \dots 2e$ possibles:

**begin**
  **for** *k=0...2e* **do**
    $S_k = n + 1 + \frac{1-ed}{k}$;
    **if** $S_k$ *n'est pas entier* **then**
      continue;
    **else**
      Résout $X^2 - S_k X + n = 0$;
      Tester si les solutions entières divisent $n$;
**end**

# Réduction pour $e$ petit

- $ed = 1 \mod \varphi(n) \quad \Rightarrow ed - 1 = k\varphi(n)$
- $\varphi(n) = (p-1)(q-1) = pq + 1 - (p+q) = n + 1 - (p+q)$.
- Si $p, q \neq 2, 3 \quad \Rightarrow p, q \leq n/4$
- Alors $k \leq 2e$

Si $e$ est petit: test exhaustif de tous les $k \in 0 \ldots 2e$ possibles:

**begin**
  **for** *k=0...2e* **do**
    $S_k = n + 1 + \frac{1-ed}{k}$;
    **if** $S_k$ *n'est pas entier* **then**
      continue;
    **else**
      Résout $X^2 - S_k X + n = 0$;
      Tester si les solutions entières divisent $n$;
**end**

Sécurité: Toujours choisir $e$ grand!

# Réduction pour $e$ grand

$$ed - 1 = k\varphi(n) = t2^s$$

- Soit $a$, premier avec $n$: $a^{k\varphi(n)} = (a^t)^{2^s} = 1 \mod n$
- ordre$(a^t) \in \{2^j, 0 \leq j \leq s\}$
- On pose $u = a^{2^{i-1}t}$. Alors
  $\exists i \in \{0 \dots s\}, \begin{cases} u & \neq & 1 \mod n \\ u^2 & = & 1 \mod n \end{cases}$

# Réduction pour $e$ grand

$$ed - 1 = k\varphi(n) = t2^s$$

- Soit $a$, premier avec $n$: $a^{k\varphi(n)} = (a^t)^{2^s} = 1 \mod n$
- ordre$(a^t) \in \{2^j, 0 \leq j \leq s\}$
- On pose $u = a^{2^{i-1}t}$. Alors
  $\exists i \in \{0 \ldots s\}, \begin{cases} u & \neq & 1 \mod n \\ u^2 & = & 1 \mod n \end{cases}$
- Mieux: $\exists i \in \{0 \ldots s\}, \begin{cases} u & \neq & \pm 1 \mod n \\ u^2 & = & 1 \mod n \end{cases}$
- $\begin{cases} u^2 - 1 & = & (u-1)(u+1) = 0 \mod n \\ u - 1 & \neq & 0 \mod n \\ u + 1 & \neq & 0 \mod n \end{cases}$

# Réduction pour $e$ grand

$$ed - 1 = k\varphi(n) = t2^s$$

- Soit $a$, premier avec $n$: $a^{k\varphi(n)} = (a^t)^{2^s} = 1 \mod n$
- ordre$(a^t) \in \{2^j, 0 \leq j \leq s\}$
- On pose $u = a^{2^{i-1}t}$. Alors
  $\exists i \in \{0 \dots s\}, \begin{cases} u & \neq & 1 \mod n \\ u^2 & = & 1 \mod n \end{cases}$
- Mieux: $\exists i \in \{0 \dots s\}, \begin{cases} u & \neq & \pm 1 \mod n \\ u^2 & = & 1 \mod n \end{cases}$
- $\begin{cases} u^2 - 1 & = & (u-1)(u+1) = 0 \mod n \\ u - 1 & \neq & 0 \mod n \\ u + 1 & \neq & 0 \mod n \end{cases}$
- Supp $(u-1) \wedge n = 1$.
  - Comme $n | (u-1)(u+1) \Rightarrow n | (u+1)$
  - $u + 1 \neq 0 \mod n$. Absurde
- $(u-1) \wedge n$ est un facteur non trivial de $n$.

# Réduction pour $e$ grand

$$ed - 1 = k\varphi(n) = t2^s$$

- Soit $a$, premier avec $n$: $a^{k\varphi(n)} = (a^t)^{2^s} = 1 \mod n$
- ordre$(a^t) \in \{2^j, 0 \leq j \leq s\}$
- On pose $u = a^{2^{i-1}t}$. Alors
  $\exists i \in \{0 \ldots s\}, \begin{cases} u & \neq & 1 \mod n \\ u^2 & = & 1 \mod n \end{cases}$
- Mieux: $\exists i \in \{0 \ldots s\}, \begin{cases} u & \neq & \pm 1 \mod n \\ u^2 & = & 1 \mod n \end{cases}$
- $\begin{cases} u^2 - 1 & = & (u-1)(u+1) = 0 \mod n \\ u - 1 & \neq & 0 \mod n \\ u + 1 & \neq & 0 \mod n \end{cases}$
- Supp $(u-1) \wedge n = 1$.
  - Comme $n | (u-1)(u+1) \Rightarrow n | (u+1)$
  - $u + 1 \neq 0 \mod n$. Absurde
- $(u-1) \wedge n$ est un facteur non trivial de $n$.

# Plan

# Attaques par modulo commun

- Contexte où $n$ est utilisé deux fois:
  - $(e_A, n), (d_A, n)$
  - $(e_B, n), (d_B, n)$
  - $e_a \wedge e_B = 1$
- Oscar veut envoyer un même message à Alice et Bob:
  - $c_A = m^{e_A} \mod n$
  - $c_B = m^{e_B} \mod n$
- Eve écoute $c_A$ et $c_B$ et connaît $e_A, e_B$

**Comment trouver M?**

# Attaques par modulo commun

- Contexte où $n$ est utilisé deux fois:
  - $(e_A, n), (d_A, n)$
  - $(e_B, n), (d_B, n)$
  - $e_a \wedge e_B = 1$
- Oscar veut envoyer un même message à Alice et Bob:
  - $c_A = m^{e_A} \mod n$
  - $c_B = m^{e_B} \mod n$
- Eve écoute $c_A$ et $c_B$ et connaît $e_A, e_B$

**Comment trouver M?**

Algorithme d'Euclide Etendu:

- $ue_A + ve_B = 1$
- $c_A^u c_B^v = m^{ue_A + ve_B} = m \mod n$

# Attaques factorielles

## Propriété

*Tous les facteurs premiers de $B!$ sont inférieurs à $B$*

- Soit $B$ tq $(p-1)|B!$ $\Rightarrow \mu(p-1) = B!$
- $a^{B!} = (a^{(p-1)})^{\mu} = 1 \mod p = 1 + \alpha p$
- $A = a^{B!} \mod n$
- $A - 1 = a^{B!} - 1 + \lambda pq = (\alpha + \lambda q)p$

# Attaques factorielles

## Propriété

*Tous les facteurs premiers de $B!$ sont inférieurs à $B$*

- Soit $B$ tq $(p-1)|B!$ $\Rightarrow \mu(p-1) = B!$
- $a^{B!} = (a^{(p-1)})^{\mu} = 1 \mod p = 1 + \alpha p$
- $A = a^{B!} \mod n$
- $A - 1 = a^{B!} - 1 + \lambda pq = (\alpha + \lambda q)p$
- $G = (A-1) \wedge n$
  - Soit $G = p$ $\Rightarrow$factorisé
  - Soit $A - 1$ est un multiple de $n$
    $\Rightarrow B! = k\varphi(n) = k(p-1)(q-1)$
    $\Rightarrow$tous les facteurs premiers de $p-1$ et $q-1$ sont $< B$.
    $\Rightarrow$choisir $B$ ni trop grand, ni trop petit

# Attaques factorielles

**begin**
  Choisir $C$ *pas trop grand*, mais $> B$ inconnu;
  Calculer $A = 2^{C!} \mod n$;
  G=pgcd(A-1,n);
  **if** $G = 1$ **then**
    Boucler avec $C$ plus grand
  **if** $G = \gamma n$ **then**
    Abandonner
**end**

Parade: Choisir $p, q$ tels que $p - 1, q - 1$ ont des facteurs premiers grands.

# Plan

# Factorisation

## Problème

1. *Ecrire $n$ sous la forme $n = \prod_{i=1}^{k} p_i^{\alpha_i}$, $p_i$ premiers*
2. *Plus faible: trouver $p > 1$ tel que $p|n$*
   $\Rightarrow$*calcul récursif des facteurs de $n/p$.*

- Problème difficile: pas d'algorithme polynomial ($\mathcal{O}\left(\log^{\gamma} n\right)$) connu
- $(p, q) \rightarrow p \times q$ est une fonction à sens unique.

# Algorithme naïf

- Prendre $r$ au hasard entre 0 et $n$
- $1/p$ chance d'avoir un multiple de $p$
- Si oui, alors $r \wedge n = p$
- Espérance du nombre de tirages: $p/2$ $\Rightarrow$pas mieux que le crible d'Eratosthène!

# Collision de suites récurrentes

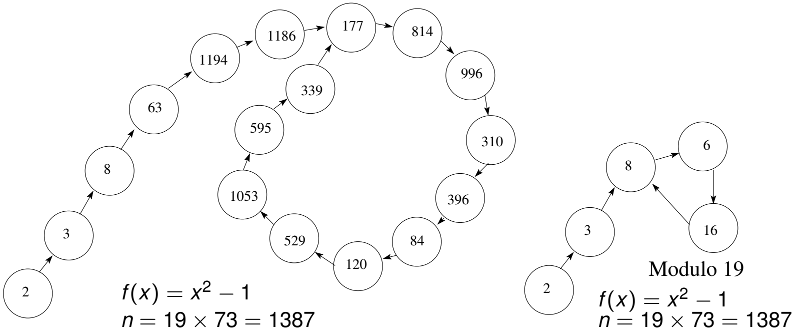
## Principe:

- Suite récurrente mod $n$: $X_0 = \text{rand}, X_{i+1} = f(X_i) \mod n$
- $\exists k, t$ tq $X_{t+k} = X_t \mod p$
- $X_{t+k} - X_t = \alpha p$
- $(X_{t+k} - X_t) \wedge n = p$ ou $pq$
- Si $X_{t+k} - X_t \neq 0 \mod n$ $\Rightarrow$on a le facteur $p$

# Collision de suites récurrentes

## Principe:

- Suite récurrente mod $n$: $X_0 = \text{rand}, X_{i+1} = f(X_i) \mod n$
- $\exists k, t$ tq $X_{t+k} = X_t \mod p$
- $X_{t+k} - X_t = \alpha p$
- $(X_{t+k} - X_t) \wedge n = p$ ou $pq$
- Si $X_{t+k} - X_t \neq 0 \mod n$ $\Rightarrow$on a le facteur $p$



# Rho Pollard

Approche naïve:

$X_0 = \text{rand}(), g = 1, i = 1;$
**while** $g = 1$ **do**
  $X_i = f(X_{i-1});$
  $j = 0;$
  **while** $p = 1$ *et* $j < i$ **do**
    $g = \text{pgcd}(X_j - X_i, n);$
    $j = j + 1;$
  $i = i + 1;$
**return** $g$

- $\mathcal{O}\left(p^2\right) = \mathcal{O}\left(n\right)$ pgcd's

# Rho Pollard

Approche naïve:

$X_0 = \text{rand}(), g = 1, i = 1;$
**while** $g = 1$ **do**
    $X_i = f(X_{i-1});$
    $j = 0;$
    **while** $p = 1$ *et* $j < i$ **do**
        $g = \text{pgcd}(X_j - X_i, n);$
        $j = j + 1;$
    $i = i + 1;$
**return** $g$

- $\mathcal{O}\left(p^2\right) = \mathcal{O}\left(n\right)$ pgcd's

**Principe de Rho Pollard**

Comparer tous les $X_j$ pour $j = 2^i + 1 \ldots 2^{i+1} - 1$ avec $X_{2^i}$

**begin**
    $y = \text{rand}(), g = 1, i = 1;$
    **while** $g = 1$ **do**
        **if** *i est une puissance de 2* **then**
            $x = y;$
        $y = f(y);$
        $g = \text{pgcd}(y - x, n);$
        $i = i + 1;$
    **return** $g$
**end**

# Rho Pollard

Approche naïve:

$X_0 = \text{rand}(), g = 1, i = 1;$

**while** $g = 1$ **do**

  $X_i = f(X_{i-1});$

  $j = 0;$

  **while** $p = 1$ *et* $j < i$ **do**

    $g = \text{pgcd}(X_j - X_i, n);$

    $j = j + 1;$

  $i = i + 1;$

**return** $g$

- $\mathcal{O}\left(p^2\right) = \mathcal{O}\left(n\right)$ pgcd's

**Principe de Rho Pollard**

Comparer tous les $X_j$ pour $j = 2^i + 1 \ldots 2^{i+1} - 1$ avec $X_{2^i}$

**begin**

  $y = \text{rand}(), g = 1, i = 1;$

  **while** $g = 1$ **do**

    **if** *i est une puissance de 2* **then**

      $x = y;$

    $y = f(y);$

    $g = \text{pgcd}(y - x, n);$

    $i = i + 1;$

  **return** $g$

**end**

- pire cas: au plus deux fois trop d'itérations
- Si $k > 0,5 + 1,18\sqrt{p}$ $\Rightarrow$proba ¿ 1/2 de trouver $p$
- peut être appliqué sur $p + 1$

# Crible quadratique

## Principe

- recherche de paires $x, y$ tq: $x^2 = y^2 \mod n$
  $\Rightarrow (x-y)(x+y) = 0 \mod n = \alpha pq$
- Avec de la chance: $(x-y) \neq \alpha \neq n$
  $\Rightarrow (x-y)$ est un facteur non trivial de $n$

Comment trouver de tels $x, y$?

$\Rightarrow$cribles quadratiques

# Crible quadratique

## Exemple (Factorisation de 7429)

$$
\begin{aligned}
87^2 &= 7429 + 140 \textit{ et } 140 = 2^2 \times 5 \times 7 \\
88^2 &= 7429 + 315 \textit{ et } 315 = 3^2 \times 5 \times 7
\end{aligned}
$$

# Crible quadratique

## Exemple (Factorisation de 7429)

$$\begin{aligned} 87^2 &= 7429 + 140 \textit{ et } 140 = 2^2 \times 5 \times 7 \\ 88^2 &= 7429 + 315 \textit{ et } 315 = 3^2 \times 5 \times 7 \end{aligned}$$

$$88^2 \times 87^2 = (2 \times 3 \times 5 \times 7)^2 \mod 7429$$

$$\begin{aligned} x &= 87 \times 88 = 227 \mod 7429 \\ y &= 2 \times 3 \times 5 \times 7 = 210 \mod 7429 \end{aligned}$$

$\Rightarrow$

$$7429 = 17 \times 437$$

# Crible quadratique

| | Exp. de 2 | Exp. de 3 | Exp de 5 | Exp. de 7 |
|---|---|---|---|---|
| $83^2$ | 2 | 3 | 1 | 0 |
| $87^2$ | 2 | 0 | 1 | 1 |
| $88^2$ | 0 | 2 | 1 | 1 |

# Crible quadratique

| | Exp. de 2 | Exp. de 3 | Exp de 5 | Exp. de 7 |
|---|---|---|---|---|
| $83^2$ | 2 | 3 | 1 | 0 |
| $87^2$ | 2 | 0 | 1 | 1 |
| $88^2$ | 0 | 2 | 1 | 1 |

- $87 \times 88$ est un carré ssi L2+L3 est paire
- trouver un vecteur $x$ tq $xM$ est pair
- trouver x tq $xM = 0 \mod 2$

# Crible quadratique

| | Exp. de 2 | Exp. de 3 | Exp de 5 | Exp. de 7 |
|---|---|---|---|---|
| $83^2$ | 2 | 3 | 1 | 0 |
| $87^2$ | 2 | 0 | 1 | 1 |
| $88^2$ | 0 | 2 | 1 | 1 |

- $87 \times 88$ est un carré ssi L2+L3 est paire
- trouver un vecteur $x$ tq $xM$ est pair
- trouver x tq $xM = 0 \mod 2$

**Exemple ( RSA-640:)**

*Cassé en 2005 en 4,5 mois*

- 36 000 000 *colonnes*
- $\approx 7 \times 10^9$ *coeffs non nuls*

**Exemple ( RSA-768:)**

*Cassé en déc. 2009 en 2 ans*

- 192 796 550 *colonnes*
- $\approx 27 \times 10^9$ *coeffs non nuls*

# Factorisation: ordres de grandeur

Rho Pollard: $\mathcal{O}\left(n^{1/4}\right)$, jusqu'à $\approx$ 30 chiffres

Courbes elliptiques: jusqu'à $\approx$ 50 chiffres

Cribles quadratiques: $\mathcal{O}\left(n^{1/6}\right)$ jusqu'à $\approx$ 238 chiffres

# Sécurité de RSA

Clés sûres:

- $e, d$ grands (cf cassage de RSA)

Nombres premiers robustes:

- Factorisation par courbes elliptiques: $p, q$ grands
- Attaques factorielles:
  - $p - 1$ doit avoir un grand facteur $r$
  - $p + 1$ doit avoir un grand facteur
  - $r - 1$ doit avoir un grand facteur
- Attaques par racines carrées: $p - q$ grand

# Algorithme de Gordon

**Données**: $b$ le nombre de bits
**Résultat**: $p$ un nombre premier robuste d'au moins $2b + 1$ bits
**begin**
  Générer deux nombres premiers $s, t$ de $b$ bits;
  Chercher $r$ premier de la forme $2kt + 1$;
  Calculer $l = 2(s^{r-2} \mod r)s - 1$;
  **return** *$p$ le plus petit premier de la forme $l + 2hrs$*
**end**

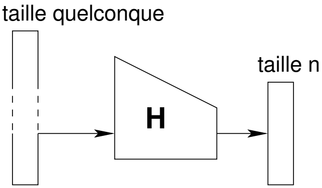
- $r - 1$ a pour facteur $t$
- Si $s \neq r$, alors $s^{r-1} = 1 \mod r$ $\Rightarrow l = 1 \mod r$ $\Rightarrow p - 1 = 0 \mod r$.
- $l = -1 \mod s$ $\Rightarrow p + 1 = 0 \mod s$.

Donc $r - 1, p - 1$ et $p + 1$ ont respectivement pour facteurs $t, r$ et $s$.

# Plan

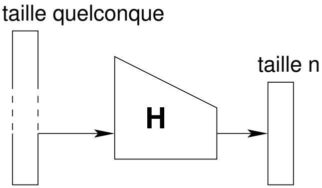
# Plan

# Fonctions de hachage




- $H : \{0,1\}^* \rightarrow \{0,1\}^n$ $\Rightarrow$résumé de taille fixe pour tout message de longueur quelconque
- Doit être rapide à calculer

# Fonctions de hachage




- $H : \{0,1\}^* \rightarrow \{0,1\}^n$ $\Rightarrow$résumé de taille fixe pour tout message de longueur quelconque
- Doit être rapide à calculer

Rôle:

- Identification
- Ne permet pas de déterminer d'autres propriétés

Motivations:

- Intégrité: MDC (Modification Detection Code)
- Authentification et intégrité: (Message Authentication Code)

# Fonctions de Hachage

## Propriétés des fonctions de hachage

Uniforme: $P[H(M) = i] = (1/2)^n$

(1) Résistance à la pré-image: Connaissant $y = H(x)$: difficile de trouver $x$

(2) Résistance à la deuxième pré-image: Connaissant $x$, difficile de trouver $x'$ tel que $H(x) = H(x')$

(3) Resistance aux collisions: Dur de trouver $x$ et $x'$ tels que $H(x) = H(x')$.

## Exercice

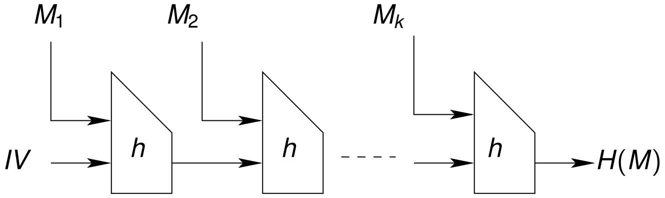
*Montrer que (3) $\Rightarrow$ (2) $\Rightarrow$ (1)*

# Fonction de hachage de Merkle Damgård

Basé sur une fonction de compression:

$$h : \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$$

Construction du hachage par itérations:




- $h$ résistante aux collisions $\Rightarrow H$ aussi [Merkle Damgård]

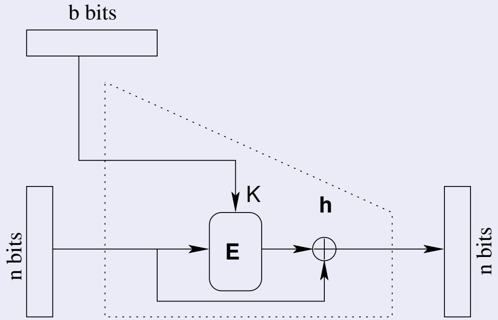
**Exercice**

*Prouver [Merkle Damgård]*

# Fonctions de compression cryptographiques

Principe: utiliser une fonction de chiffrement à clé secrète pour *compresser*.

## Construction de Davis-Meyer

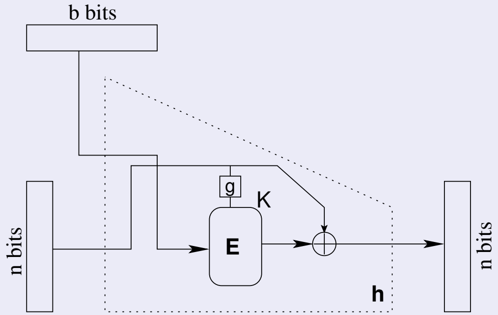



- Attaque de la résistance à la pré-image [Drew Dean 99]

# Fonctions de compression cryptographiques

Principe: utiliser une fonction de chiffrement à clé secrète pour *compresser*.

## Construction de Miyaguchi-Preneel




- $g$: fonction d'adaptation de la taille de la clé

# Attaque sur les fonctions de hachage

## Le paradoxe des anniversaires

- Population de $k$ personnes, $n$ jours dans l'année
- Nombre de distributions d'anniv. différents: $A_n^k = \frac{n!}{(n-k)!}$
- $P[\text{deux anniv. le même jour}] = 1 - \frac{A_n^k}{n^k}$

- Si $k > 1.18\sqrt{n}$ $\Rightarrow$Proba $> 50\%$ d'avoir une collision

# Attaque sur les fonctions de hachage

## Le paradoxe des anniversaires

- Population de $k$ personnes, $n$ jours dans l'année
- Nombre de distributions d'anniv. différents: $A_n^k = \frac{n!}{(n-k)!}$
- $P[\text{deux anniv. le même jour}] = 1 - \frac{A_n^k}{n^k}$

## Exemple

k=9: *P=9,5%*

k=23: *P=50,7%*

k=70: *P=99,9%*

- Si $k > 1.18\sqrt{n}$ $\Rightarrow$Proba $>$ 50% d'avoir une collision

# Attaque de Yuval sur les fonctions de hachage

But: Corrompre un message $M$ en $M'$ sans être détecté par la fonction de hachage: $h(M) = h(M')$.

## Attaque de Yuval

**Données**: $h$: fonction de hachage
**Données**: $X$: message légitime
**Données**: $Y$: message frauduleux
**Résultat**: $X' \approx X$, $Y' \approx Y$ tels que $h(X') = h(Y')$
**begin**
  Générer $t = 2^{\frac{m}{2}}$ modifications mineures $X_i$ de $X$;
  **foreach** $X_i$ **do**
    Calculer $h(X_i)$;
  Générer des modifications mineures $Y_j$ de $Y$ jusqu'à ce que $h(Y_{j_0}) = h(X_{i_0})$;
  **return** $(X_{j_0}, Y_{j_0})$
**end**

# Attaque de Yuval sur les fonctions de hachage

But: Corrompre un message $M$ en $M'$ sans être détecté par la fonction de hachage: $h(M) = h(M')$.

## Attaque de Yuval

**Données**: $h$: fonction de hachage
**Données**: $X$: message légitime
**Données**: $Y$: message frauduleux
**Résultat**: $X' \approx X$, $Y' \approx Y$ tels que $h(X') = h(Y')$
**begin**
  Générer $t = 2^{\frac{m}{2}}$ modifications mineures $X_i$ de $X$;
  **foreach** $X_i$ **do**
    Calculer $h(X_i)$;
  Générer des modifications mineures $Y_j$ de $Y$ jusqu'à ce que $h(Y_{j_0}) = h(X_{i_0})$;
  **return** $(X_{j_0}, Y_{j_0})$
**end**

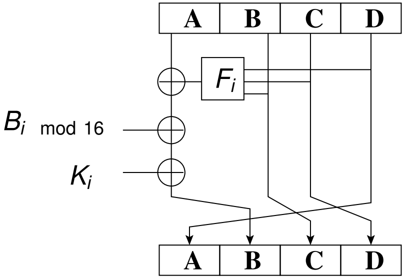
Répudiation: envoyer $X$ et soutenir qu'on a envoyé $Y$.

Fiabilité des fonctions de hachage: taille de bloc $> 128$ bits

# MD5

- Blocs $B$ de $b = 512$ bits, décomposé en 16 sous-blocs $B_i$
- Empreinte sur $n = 128$ bits.
- 64 rondes, variable d'état $(A, B, C, D)$ de $4 \times 32 = 128$ bits

1 ronde de MD5:




où

- Si $i = 0 \mod 4$: $F_i = (B \text{ AND } C) \text{ OR } (\overline{B} \text{ AND } D)$
- Si $i = 1 \mod 4$: $F_i = (D \text{ AND } B) \text{ OR } (\overline{D} \text{ AND } C)$
- Si $i = 2 \mod 4$: $F_i = B \oplus C \oplus D$
- Si $i = 3 \mod 4$: $F_i = C \oplus (B \text{ OR } \overline{D})$

# Sécurité de MD5

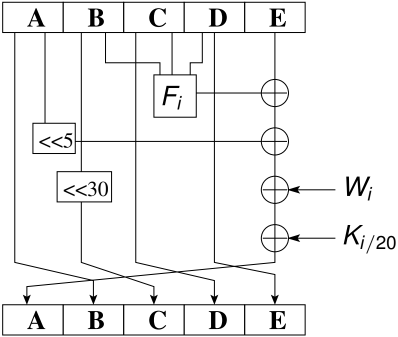
- A priori: $\sqrt{2^n} = 2^{64}$ calculs

# Sécurité de MD5

- A priori: $\sqrt{2^n} = 2^{64}$ calculs
- Mais attaques possibles en $2^{42}$ voir $2^{30}$
  $\Rightarrow$plus considéré comme fiable

# SHA-1

- Blocs $B$ de $b = 512$ bits, décomposés en 16 sous-blocs $(B_i)_{i=0\ldots15}$
- Puis étendus en 80 blocs $(W_i)_{i=0\ldots79}$
- 4 constantes $(K_i)_{i=0\ldots3}$
- Empreinte sur $n = 160$ bits.
- $4 \times 16 = 64$ rondes, variable d'état $(A, B, C, D, E)$ de $5 \times 32 = 160$ bits




- Si $i = 0 \mod 4$: $F_i = (B \text{ AND } C) \text{ OR } (\overline{B} \text{ AND } D)$
- Si $i = 1 \mod 4$: $F_i = B \oplus C \oplus D$
- Si $i = 2 \mod 4$: $F_i = (B \text{ AND } C) \oplus (\overline{B} \text{ AND } D) \oplus (\overline{C} \text{ AND } D)$
- Si $i = 1 \mod 4$: $F_i = B \oplus C \oplus D$

# Sécurité de SHA-1

Recherche de collisions:

- Par force brute (Yuval): $2^{80}$ faisable en théorie
- [Wang& Al. 05]: $2^{69}$, [Yao & Al. 07]: $2^{63}$

$\Rightarrow$plus considéré comme fiable

# Sécurité de SHA-1

Recherche de collisions:

- Par force brute (Yuval): $2^{80}$ faisable en théorie
- [Wang& Al. 05]: $2^{69}$, [Yao & Al. 07]: $2^{63}$

$\Rightarrow$plus considéré comme fiable

Extension: SHA-256. Empreinte de 256 bits
$\Rightarrow$pas encore d'attaques praticable connues

# Plan

# Authentification: signatures

Signatures RSA: encoder un message (ou son résumé, de préférence), par une clé privée RSA.

# Authentification: signatures

**Signatures RSA:** encoder un message (ou son résumé, de préférence), par une clé privée RSA.

**Signatures DSS:** basé sur le log discret. Clé publique $y$, clé privée $x$.

**begin**
- choisir $q$, premier de 160 bits;
- trouver $p = kq + 1$ premier de 512 à 1024 bits;
- **while** $g = 1$ **do**
  - choisir $a$ et calculer $g = a^k \mod p$;
- choisir $x$ de 160 bits;
- **return** $y = g^x \mod p$

**end**

1. Alice choisit $k$ aléatoire $< q$
2. Alice calcule et envoi
   - $r = (g^k \mod p) \mod q$
   - $S = k^{-1}\mathrm{SHA1}(Message) + xr \mod q$
3. Bob vérifie la signature ssi: $v = r$
   - $w = s^{-1} \mod q$
   - $u = \mathrm{SHA1}(Message)$
   - $t = rw \mod q$
   - $v = (g^u y^t \mod p) \mod q$

# Certificats numériques

- Associer une clé à son propriétaire:
  $\Rightarrow$certificat numérique: association identité $\leftrightarrow$ clé
- Authentifier un certificat:
  $\Rightarrow$les certificats sont signés par une autorité
- Trouver un certificat:
  $\Rightarrow$annuaires publics maintenus
- Sécurité:
  $\Rightarrow$auto-signatures des autorités
  $\Rightarrow$architectures à clé publiques PKI