



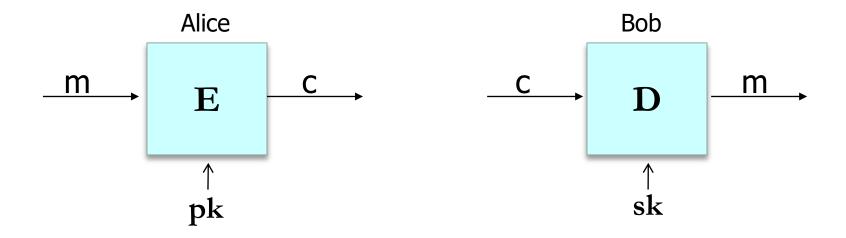
## CRYPTOGRAPHIE À BASE DE SAC À DOS

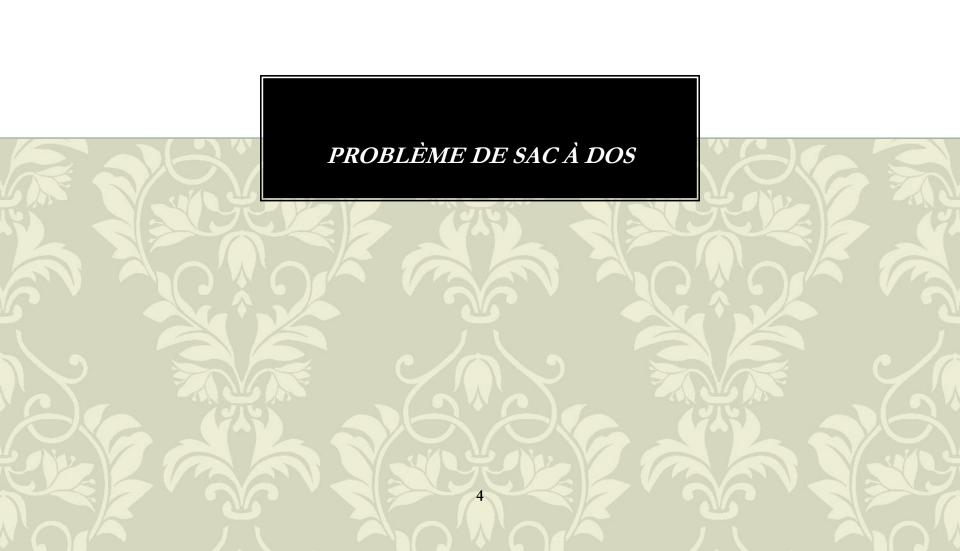
Réalisé par: Loubna EL BACHIRI Tarik RAHMATALLAH Jugé par: Mme. Hanane El Bakkali Mr. Hussain BENAZZA

- 1 Cryptographie publique
  - Problème général de sac à dos
    - 3 Cryptosystème de Merkle-Hellman
    - 4 Réduction de réseau
  - 5 Cryptanalyse du système de Merkle-Hellman
- 6 Cryptosystème de Chor-Rivest

## CRYPTOGRAPHIE PUBLIQUE

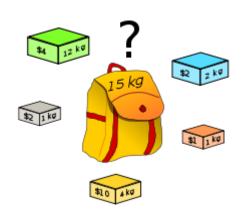
Bob génère une paire de clés(PK, SK) et donne PK à Alice





## PROBLÈME GÉNÉRAL DE SAC À DOS

Comment remplir le sac de façon à maximiser la valeur totale des objets tout en respectant la capacité du sac ?



Problème NP-complet!

## CRYPTOSYSTÈME DE MERKLE ET HELLMAN

- Défini par Merkle et Hellman en 1978.
- > Cryptosystème asymétrique.
- Basé sur le problème de la somme des sous ensembles.

## CRYPTOSYSTÈME DE MERKLE ET HELLMAN

#### Génération de clés

- Choisir un entier n suffisamment grand (taille du bloc)
- Choisir une suite (a1, a2,..., an) super croissante.
- Choisir un entier N supérieur à la somme des bi
- Choisir un entier A inférieur à N et premier avec N
- Calculer les ai = A<sup>-1</sup>\*bi mod M, les trier en ordre croissant et garder la permutation.

La clé publique est la suite des bi tandis que la clé privée est composée de N, A et (a1, ..., an)

## CRYPTOSYSTÈME DE MERKLE ET HELLMAN

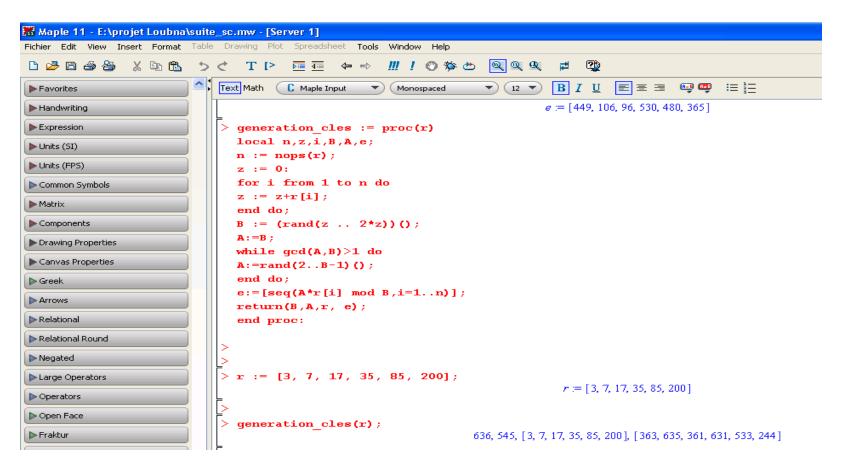
#### Chiffrement

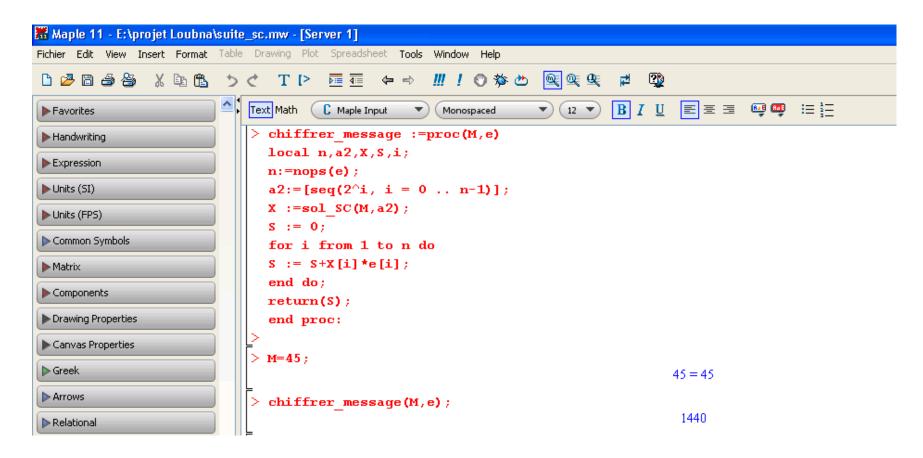
- Le message chiffré est écrit en binaire sous la forme m1,m2...mn avec mi = 0 ou 1.
- On calcule le chiffré d'un message comme suit : c=∑mi\*bi

#### Déchiffrement

- Calculer  $d = A^{-1} \times c \mod N$ .
- Puis calculer les e1, e2, ...., en tels que d=∑ei\*ai. Il s'agit en fait de résoudre un problème de sac à dos très simple (suite super croissante)

Ei=Mi pour tout i





```
▶ Canvas Properties

▶ Greek

▶ Arrows

▶ Relational

▶ Relational Round

▶ Negated

▶ Large Operators

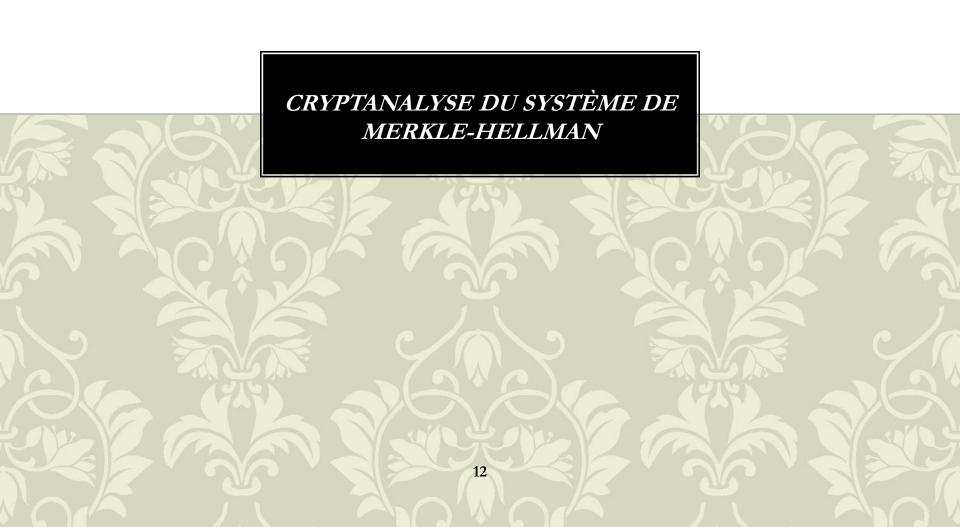
▶ Operators

▶ Open Face

▶ Fraktur
```

```
> dechiffrer_cryptogramme := proc(S,A,B,a)
local S1,X1,M,i;
S1:=(S*modp(1/A, B)) mod B;
X1 := sol_SC(S2,a);
M := 0:
for i from 0 to n-1 do
M := M+X1[i+1]*2^i
end do:
return(M);
end proc:
> dechiffrer_cryptogramme(S,A,B,a);
```

45



## **RÉDUCTION DE RÉSEAU**

Soit B = (b1,..., bn) une famille de vecteurs linéairement indépendants de Rn. Le réseau engendré par la base B est l'ensemble :

$$L(B) = \{ \sum_{i=1}^{n} xibi \mid xi \in \mathbb{Z} \}$$

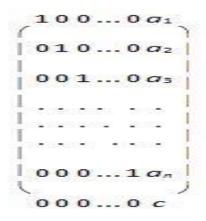
- L'idée de la réduction de réseau est de calculer une nouvelle base engendrant le même réseau que (b1,...,bn) mais dont les vecteurs sont plus courts et plus orthogonaux.
- ➤ Une base (b1, b2,..., bn) est LLL-réduite si, la base (b1\*, b2\*,..., bn\*) produite par la méthode d'orthogonalisation de Gram-Schmidt vérifie :

$$|\mu i,j| \le 1/2 \text{ pour } 1 \le j \le i \le n$$
 (1)

$$3/4 \mid |b \mid i-1^*||^2 \le |b \mid i^* + \mu \mid i, i-1 \mid b \mid i-1^*||^2 \quad \text{pour } 1 \le i \le n$$
 (2)

#### CRYPTANALYSE DU SYSTÈME DE MERKLE ET HELLMAN

- Attaquer le cryptosystème de Merkle et Hellman revient a résoudre l'équation:  $\sum_{i=1}^n mi*ai=c$
- On considère le réseau engendré par les lignes de la matrice suivante :



### CRYPTANALYSE DU SYSTÈME DE MERKLE ET HELLMAN

Le plus court vecteur doit vérifier les conditions suivantes:

• 
$$| m_{n+1} | = 0$$

• | mi | 
$$\leq 1$$
 pour  $1 \leq i \leq n$ 

• 
$$\sum_{i=1}^{n} mi*ai = c$$



```
#produit scalaire
prod_scal:= proc (x, y)
add(x[i]*y[i], i = 1 .. nops(x));
end proc:

= > u := [1, 45, 8, -36]; v := [49, 8, -5, 4];
prod_scal(u, v);

u := [1, 45, 8, -36]
v := [49, 8, -5, 4]
225
```

```
▶ Expression
Units (SI)
Units (FPS)
Common Symbols
Matrix
▶ Components
Drawing Properties
Canvas Properties
Greek
> Arrows
Relational
▶ Relational Round
▶ Negated
Large Operators
Operators
```

```
gramschmidt := proc(M)
local i, j, n, B, P, u, k:
with(LinearAlgebra):
n := nops(M):
B \coloneqq [\ ]: P \coloneqq [\ ]:
for i from 1 to n do
\mathbf{u} \coloneqq [\operatorname{seq}(0, \mathbf{k} = 1 .. \mathbf{n})]:
for j from 1 to i-1 do
u[j] := prod\_scal(M[i], B[j])/prod\_scal(B[j], B[j]):
end do:
u[i] := 1:
P \coloneqq [op(P), u]:
B := [op(B), [seq(M[i][k]-add(u[j]*B[j][k],
j = 1 ... i-1), k = 1 ... n)]]:
end do:
 return (B, P);
end proc:
with(linalg):
M := [[2, 1, 0], [1, 4, -2], [0, -2, -1]];
gramschmidt(M);
                                                                    M = [[2, 1, 0], [1, 4, -2], [0, -2, -1]]
                                      \left[ [2,1,0], \left[ -\frac{7}{5}, \frac{14}{5}, -2 \right], \left[ \frac{10}{23}, -\frac{20}{23}, -\frac{35}{23} \right], \left[ [1,0,0], \left[ \frac{6}{5}, 1, 0 \right], \left[ -\frac{2}{5}, -\frac{6}{23}, 1 \right] \right]
```

(1)

```
▶ Expression
▶ Units (SI)
Units (FPS)
Common Symbols
▶ Matrix
▶ Components
Drawing Properties
▶ Canvas Properties
▶ Greek
> Arrows
▶ Relational
Relational Round
▶ Negated
Large Operators
Operators
Den Face
Fraktur
▶ Script
Miscellaneous
```

```
redfaible := proc(M)
    local i, j, k, U, N;
   N := M;
    U := gramschmidt(M)[2];
   for i from 2 to nops(M) do
   for j from i-1 by -1 to 1 do
   N[i] := N[i] - \text{round}(U[i][j]) * N[j];
   for k from 1 to j do
   U[i][k] := U[i][k] - \text{round}(U[i][j]) * U[j][k]
    end do
    end do
    end do:
   return N:
    end proc:
   lovasz := proc(M)
   local i, n, G;
   n := nops(M);
    G := gramschmidt(M);
   for i to n-1 do
   if \, evalb(scal(G[1][i+1],G[1][i+1])
    < (3/4 - G[2][i+1][i]^2 * scal(G[1][i], G[1][i]))
    then return [false, i]
    end if:
    end do:
   return [true, 0];
    end proc:
> myLLL := proc(M)
   local N, B, x:
    N := redfaible(M):
    B := lovasz(N):
    while not(B[1]) do
   \mathbf{x} \coloneqq \mathbf{N}[\mathbf{B}[2]]:
   N \coloneqq redfaible(subsop(B[2] = N[B[2] + 1], B[2] + 1 = x, N)):
    B := lovasz(N):
    end do:
    return N:
    end proc:
```

## CRYPTOSYSTÈME DE CHOR-RIVEST

#### Génération de clés

- choisir un nombre premier P et un nombre entier h , et créer un corps fini  $GF(P^h) = Zp[x]/f(x)$  où f(x) est un polynôme de degré h qui est irréductible dans Z/pZ .soit g(x) un élément primitif de Z/pZ.
- ✓ Pour chaque i  $\in \mathbb{Z}/p\mathbb{Z}$  calculer les logarithmes discrets ai =  $\log_{g(x)}(x+i)$ .
- ✓ Choisir un entier aléatoire d avec  $0 \le d \le P^h$  -2.
- ✓ Calculer ci = (ai+d) mod  $P^h$ -1 pour  $0 \le i \le p$ -1.
- ✓ la clé publique est([c0,c1,.....cp-1],P,h) et la clé privée est [f(x),g(x),d].

## CRYPTOSYSTÈME DE CHOR-RIVEST

#### chiffrement

 Les messages sont des suites binaires de longueur P avec exactement h 1

$$(\sum_{i=0}^{p-1} \min = h).$$

Le cryptogramme est obtenu par :

$$C = \sum_{i=0}^{p-1} mici (mod P^h - 1).$$

#### déchiffrement

- Calculer  $r = (C-hd) \mod (P^h-1)$ .
- Calculer  $u(x) = g(x)^r \pmod{f(x)}$ .
- Calculer s(x) = u(x) + f(x).
- Factoriser s(x) en facteurs linéaires  $S(x) = \prod_{j=1}^{h} (x + tj)$

#### **CONCLUSION**

- après la parution de la méthode de Merkle-Hellman, il a été remarqué que des répétitions pouvaient apparaître.
- L'algorithme LLL permet de déchiffrer quasi systématiquement le message codé.
- la méthode de Chor-Rivest ne semble pas encore avoir été attaquée.
- non plus utilisée car pour avoir une résistance comparable à du RSA 512 bits, il faudrait une clé publique de l'ordre de 36000 bits.

# Mercí pour votre attention





## CRYPTOGRAPHIE À BASE DE SAC À DOS

Réalisé par: Loubna EL BACHIRI Tarik RAHMATALLAH Jugé par: Mme. Hanane El Bakkali Mr. Hussain BENAZZA