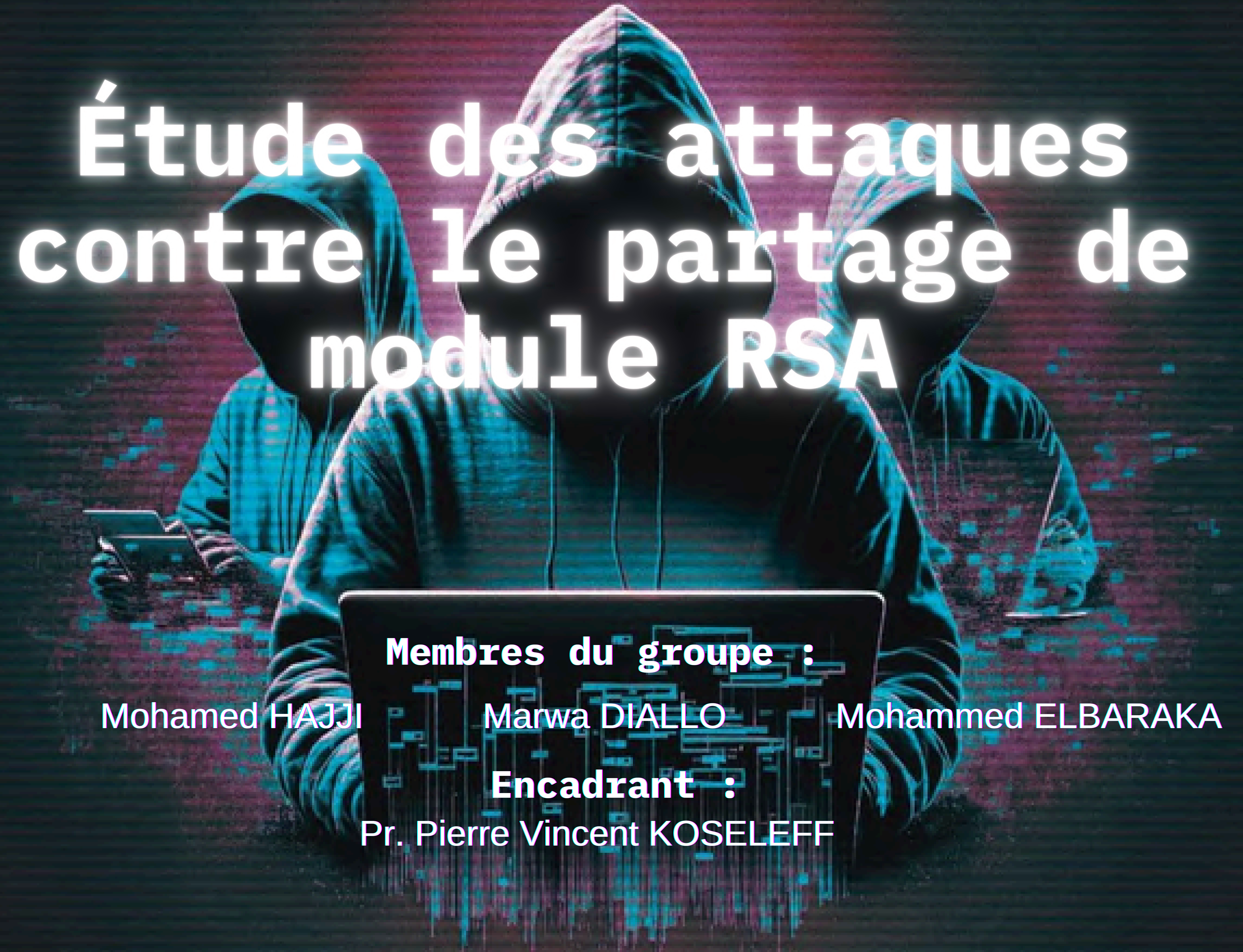


Étude des attaques contre le partage de module RSA

Membres du groupe :

Mohamed HAJJI Marwa DIALLO Mohammed ELBARAKA

Encadrant :

Pr. Pierre Vincent KOSELEFF

PLAN

01

PRINCIPES DE CHIFFREMENT À TRAVERS L'HISTOIRE

02

ATTAQUES CONTRE LE PARTAGE DU MODULE RSA

03

CHANCES DE RÉUSSITE D’UN ESPION

04

PERSPECTIVES

PRINCIPES DE CHIFFREMENT À TRAVERS L'HISTOIRE

PRINCIPES DE CHIFFREMENT À TRAVERS L'HISTOIRE

Chiffrement par Substitution

Chiffrement Polyalphabétique

Chiffrement informatique

- Diffie-Hellman
- RSA
- ElGamal

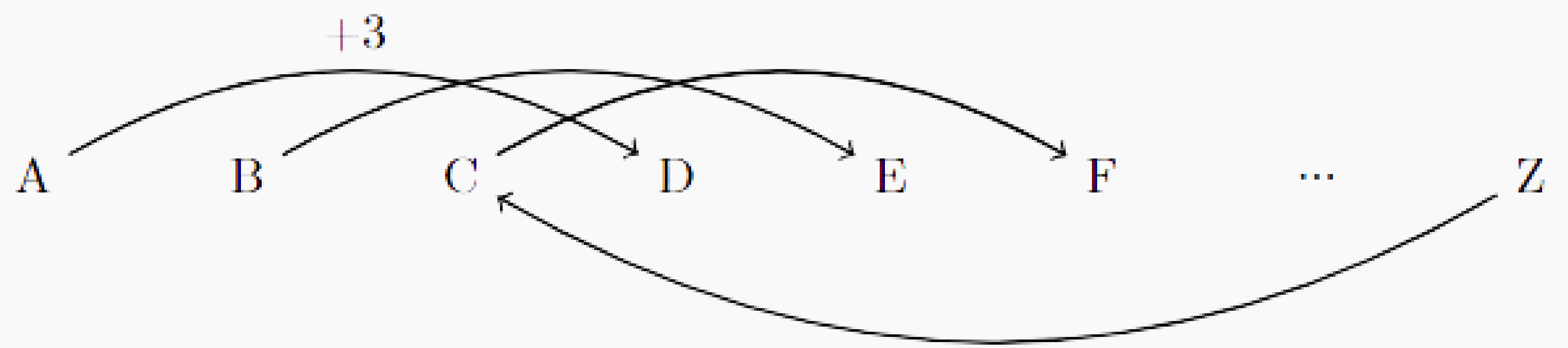
PRINCIPES DE CHIFFREMENT À TRAVERS L'HISTOIRE

Chiffrement par Substitution : Méthode de César

La méthode consiste à décaler chaque lettre du message clair d'un certain nombre de positions dans l'alphabet.

Par exemple, avec un décalage de 3 :

Texte en clair	Texte chiffré
A	D
B	E
C	F
...	...
Z	C



Facile à mettre en oeuvre mais facile à casser également !

PRINCIPES DE CHIFFREMENT À TRAVERS L'HISTOIRE

Chiffrement Polyalphabétique :Méthode de Vigenère

$$M = m_1 m_2 \ldots m_n$$

$$K = k_1 k_2 \ldots k_n$$

$$C_i = (M_i + K_i) \mod 26$$

Le tableau de Vigenère

	A	B	C	D	E	F	G	H	...	Z
A	A	B	C	D	E	F	G	H	...	Z
B	B	C	D	E	F	G	H	I	...	A
C	C	D	E	F	G	H	I	J	...	B
...		...	...	..	..		..	...	...	
Z	Z	A	B	C	D	E	F	G	...	Y

PRINCIPES DE CHIFFREMENT À TRAVERS L'HISTOIRE

Chiffrement Polyalphabétique :Méthode de Vigenère

Exemple :

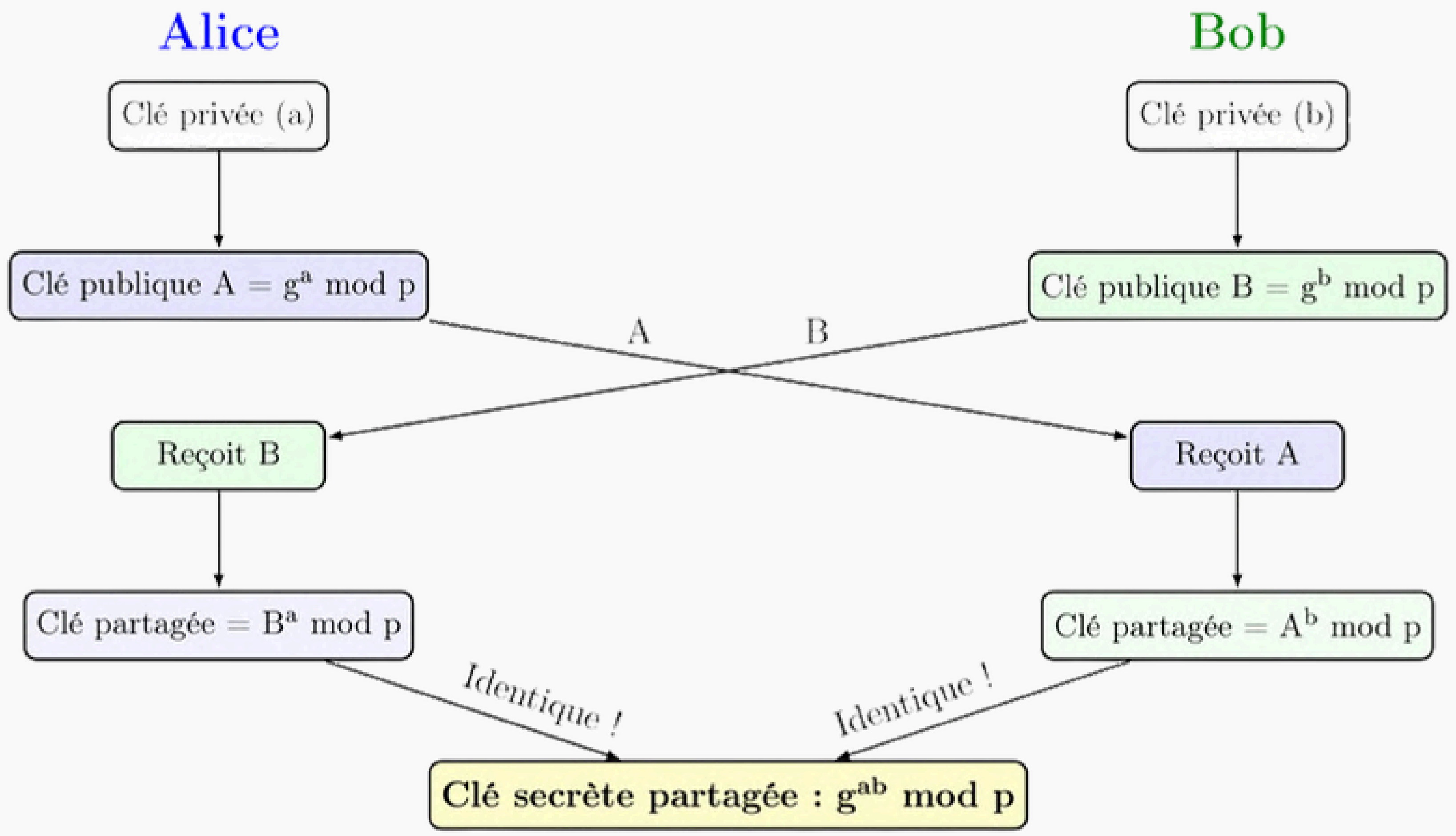
Message clair	B	O	N	J	O	U	R
Clé	C	L	E	F	C	L	E
Message chiffré	D	Z	R	O	Q	F	T

Simple et efficace pour des messages courts mais vulnérable par analyse par fréquence si la clé est courte !

PRINCIPES DE CHIFFREMENT À TRAVERS L'HISTOIRE

Chiffrement informatique : Diffie-Hellman

1. Tout le monde connaît deux nombres publics :
   un grand nombre premier (**p**) et un générateur (**g**).
2. Chacun choisit un nombre secret :
   Alice choisit **a** et Bob choisit **b**.
3. Ils calculent chacun un nombre public :
   — Alice calcule **A = g^a mod p**.
   — Bob calcule **B = g^b mod p**.
4. Ils échangent leurs nombres publics :
   — Alice envoie **A** à Bob.
   — Bob envoie **B** à Alice.
5. Chacun calcule le code secret commun :
   — Alice calcule **B^a mod p**.
   — Bob calcule **A^b mod p.**



PRINCIPES DE CHIFFREMENT À TRAVERS L'HISTOIRE

Chiffrement informatique : RSA

Pour chiffrer un message **M:**
On calcule **$C = M^e \mod n$**
où **C** est le message chiffré.

Génération des clés

- 2 nombres premiers **p** et **q** , puis calcule le module **$n = p \times q$**
- On calcule **$\phi(n) = (p - 1)(q - 1)$** (indicatrice d'Euler).
- **e** un entier tel que **$1 < e < \phi(n)$** et **$\text{pgcd}(e, \phi(n)) = 1$**.
- On calcule **d** tel que **$e \cdot d \equiv 1 \mod \phi(n)$**.

Chiffrement

Déchiffrement

Pour déchiffrer un message **C** :
On calcule **$M = C^d \mod n$**
où **M** est le message en question

La clé publique est (n, e) et la clé privée est (ϕ(n), d).

PRINCIPES DE CHIFFREMENT À TRAVERS L'HISTOIRE

Chiffrement informatique : ElGamal

1. Bob choisit des paramètres publics :
2. Alice veut envoyer un message **M** à Bob :
– Alice récupère la clé publique de Bob **(p, g, B)**.
– Alice choisit un entier aléatoire secret
k ∈ {1, . . . , p − 1}, appelé clé éphémère.
3. Alice chiffre le message **M** :
– Alice calcule le chiffré (**C1 , C2**) :

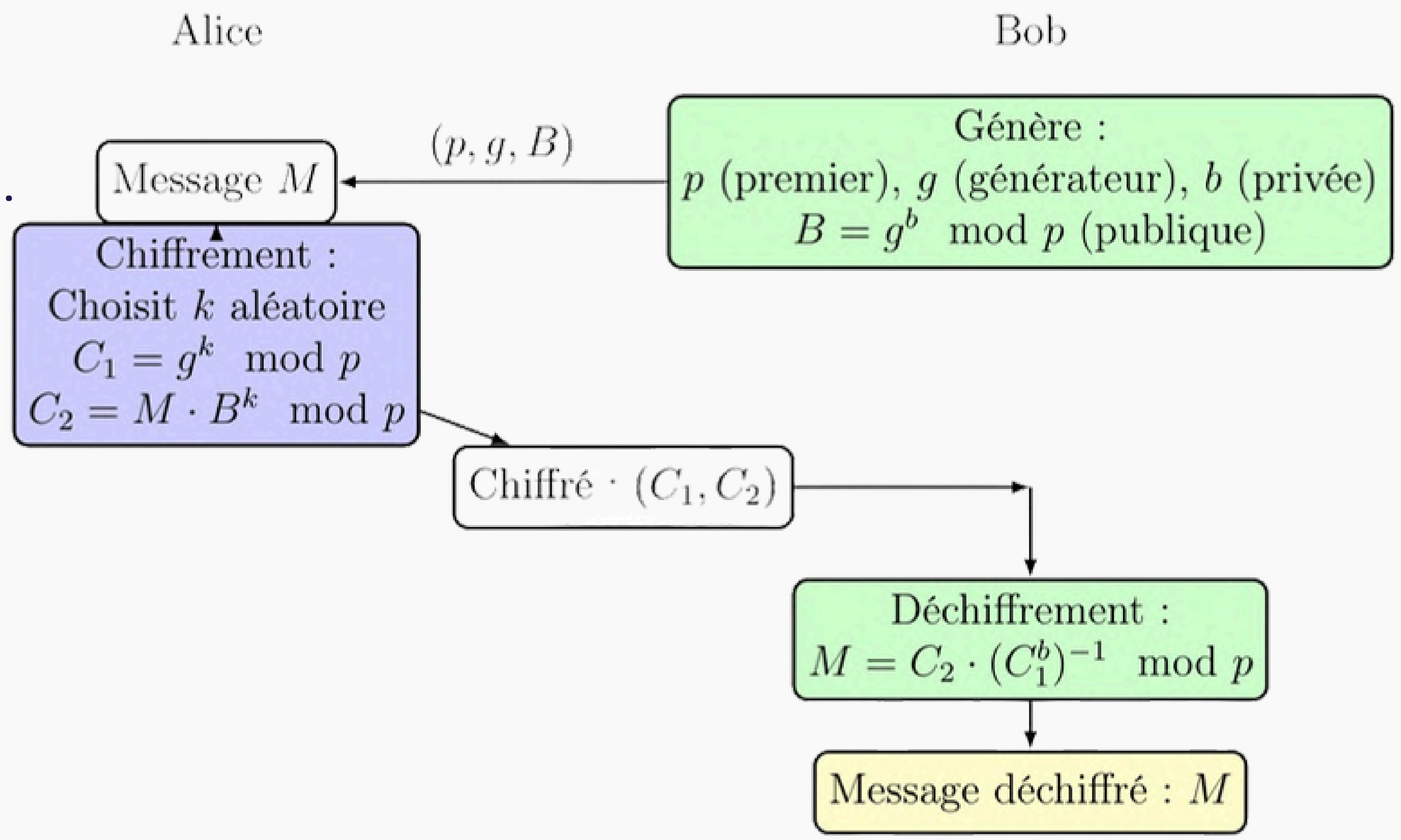
C1 ≡ g^k (mod p)

C2 ≡ M · B^k (mod p)

4. Alice envoie le chiffré à Bob :
– Alice envoie le chiffré (**C1 , C2**) à Bob.
5. Bob déchiffre le message :
– Bob utilise sa clé privée b pour déchiffrer :

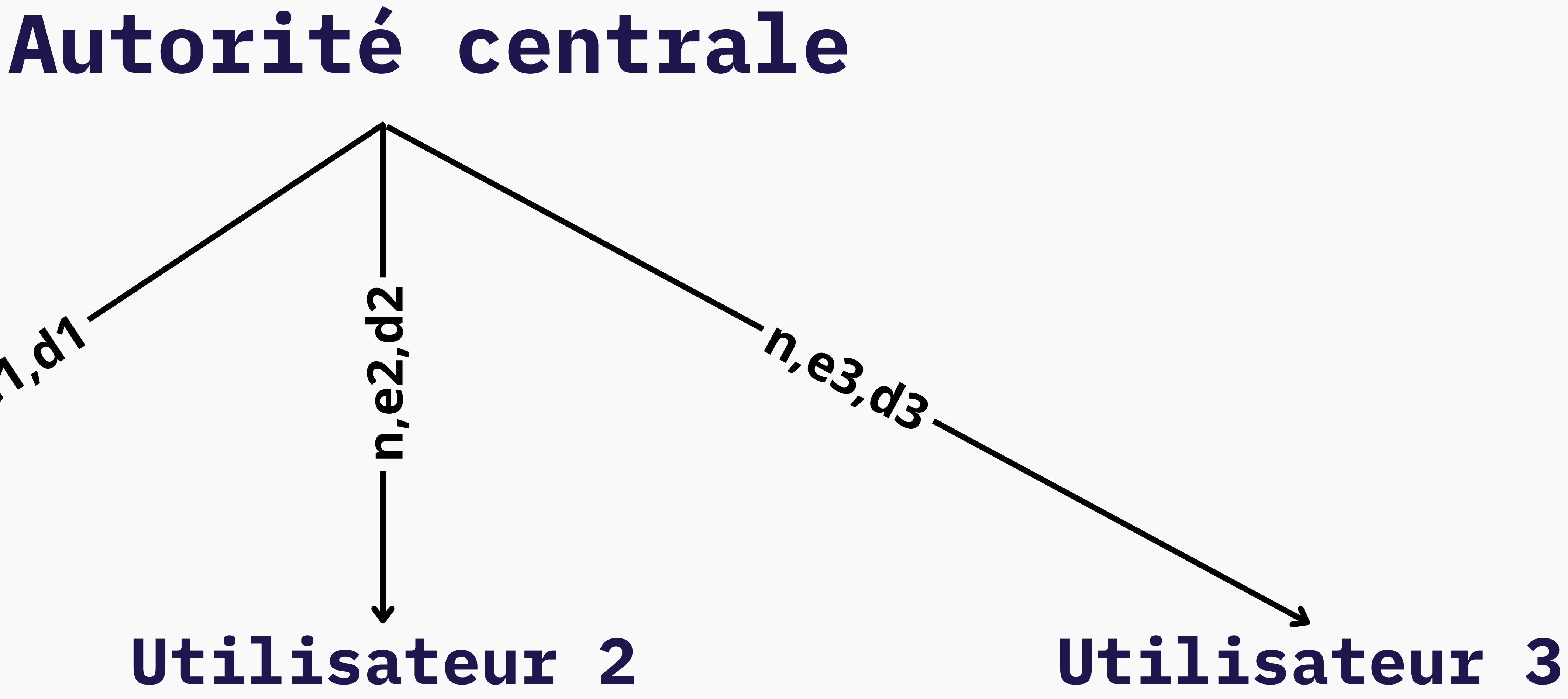
M ≡ C2 · (C1^b)−1 (mod p)

≡ C2 · C1^(−b) (mod p)



Partage du module RSA

Partage du module RSA



Partage du module RSA

1. Complexité de génération de clés

```
def find_large_prime (bits) :
    while True :
        p = ZZ(getrandbits (bits))
        if p.is_prime()==True:
            return p
%time prime = find_large_prime (2048)
print (prime)
```

Listing 3.1 – Simulation sur Sagemath pour trouver un nombre premier

CPU times : user 19.2 s, sys : 0 ns, total : 19.2 s Wall time : 19.3 s

Simulation pour bits = 3000:

CPU times : user 2min 29s, sys : 31 ms, total : 2min 29s Wall time : 2min 30s

Partage du module RSA

2. Attaques éventuelles

- **Cas 1 :** $ed - 1 = \phi(n)$
- **Cas 2 :** $ed - 1 = \alpha\phi(n)$ **avec** $\alpha \in \mathbb{Z} \setminus \{1\}$
- **Cas 3 :** $(e_1, e_2) = 1$ **et** $m_1 = m_2$

Partage du module RSA

2.Attaques éventuelles

2.1. Cas où $ed - 1 = \phi(n)$

$$n \text{ et } \phi(n) \text{ connus} \implies \text{factorisation de } n \text{ possible.}$$

Résolution de l'équation

$$X^2 + (\phi(n) - n - 1)X + n = 0$$

$$\Downarrow$$

$$p, q$$

Partage du module RSA

2.Attaques éventuelles

2.1. Cas où $ed - 1 = \phi(n)$

```
p=next_prime(randint(10^(2-1),10^2-1))
q=next_prime(randint(10^(2-1),10^2-1))
n=p*q
phi_n=(p-1)*(q-1)
for e in range(phi_n) :
    a=ZZ(e).xgcd(phi_n)
    if a[0]==1 and a[2]==-1:
        break
d=a[1]
p,q,n,phi_n,e,d
```

Listing 3.3 – Simulation de généréation d'un triplet de clés

(53, 29, 1537, 1456, 31, 47)

```
f=e*d-1
g=f-n-1
x = var('x')
equation = x^2 + g*x +n== 0
solutions = solve(equation, x)
solutions
```

Listing 3.4 – Simulation du calcul de p et q par l'utilisateur

[x == 53, x == 29]

Partage du module RSA

2.Attaques éventuelles

2.2. Cas où $ed - 1 = \alpha\phi(n)$ avec $\alpha \in \mathbb{Z} \setminus \{1\}$

$\phi(n) = (p-1)(q-1)$ avec $p \neq \mathrm{q} \gg 2$ donc $\phi(n)$ est divisible par 4.

$$\implies ed - 1 = 2^s t \quad \text{avec } s \geq 2 \text{ et } t \text{ impair}$$

On choisit au hasard $y \in \mathbb{Z}/n\mathbb{Z} \setminus \{0\}$. Si $(y,n) \neq 1$, alors (y,n) est égal à p ou q car

$$(y,n) \mid n \iff (y,n) \mid pq$$

En prenant $(p,(y,n)) = 1$ $(y,n) \mid q$ et donc $(y,n) = q$

Partage du module RSA

2.Attaques éventuelles

2.2. Cas où $ed - 1 = \alpha\phi(n)$ avec $\alpha \in \mathbb{Z} \setminus \{1\}$

Sinon, $y \in (\mathbb{Z}/n\mathbb{Z})^*$. On a donc $y^{\phi(n)} \equiv 1 \pmod{n}$, ce qui implique

$$y^{\alpha\phi(n)} = y^{2^s t} \equiv 1 \pmod{n}.$$

Si $y^t \not\equiv 1 \pmod{n}$, alors $\exists\, j$ minimal dans $\{1, 2, \dots, s\}$ tel que $y^{2^j} \equiv 1 \pmod{n}$

Objectif : Trouver une racine carrée de 1 différente de 1 et de - 1

Partage du module RSA

2.Attaques éventuelles

2.2. Cas où $ed - 1 = \alpha\phi(n)$ avec $\alpha \in \mathbb{Z} \setminus \{1\}$

Enoncé du lemme :

Soit $x \in \mathbb{Z}/n\mathbb{Z}$ tel que $x \not\equiv \pm 1 \pmod{n}$ et $x^2 = 1 \pmod{n}$.

Alors $(x-1, n)$ est un diviseur non trivial de n.

Démonstration :

$$x^2 \equiv 1 \pmod{n} \implies (x-1)(x+1) \equiv 0 \pmod{n}$$

$$\implies (x-1)(x+1) \equiv 0 \pmod{p} \quad \text{et} \quad (x-1)(x+1) \equiv 0 \pmod{q}$$

$$\implies ((x-1) \equiv 0 \pmod{p} \text{ et } (x+1) \equiv 0 \pmod{q}) \quad \text{ou}$$

$$((x+1) \equiv 0 \pmod{p} \text{ et } (x-1) \equiv 0 \pmod{q}) \quad \text{car } x \not\equiv \pm 1 \pmod{n}$$

D'où $(x-1, n)$ et $(x+1, n)$ sont des diviseurs non triviaux de n (différents de 1 et de n).

Partage du module RSA

2. Attaques éventuelles

2.2. Cas où $ed - 1 = \alpha\phi(n)$ avec $\alpha \in \mathbb{Z} \setminus \{1\}$

Remarque : Résolution par le théorème Chinois

$$x^2 \equiv 1 \pmod{n},\ x \not\equiv \pm 1 \pmod{n} \Leftrightarrow \begin{cases} (x \equiv 1 \pmod{p} \text{ et } x \equiv -1 \pmod{q}) \\ \text{ou} \\ (x \equiv -1 \pmod{p} \text{ et } x \equiv 1 \pmod{q}) \end{cases}$$

$\Rightarrow$ **Trouver une racine carrée non triviale dans l'ensemble quotient est aussi difficile que factoriser n**

Partage du module RSA

2.Attaques éventuelles

2.2. Cas où $ed - 1 = \alpha\phi(n)$ avec $\alpha \in \mathbb{Z} \setminus \{1\}$

```
p=31
q=67
n=p*q
Zn=IntegerModRing(n)
l=[]
for i in Zn :
    if i!=ZZ(-1) and i!=ZZ(1) and i^2==ZZ(1):
        a=n.gcd(i-1)
        l.append(a)
l
```

Listing 3.5 – Simulation de la factorisation de n par le lemme précédent

[67, 31]

```
Zn=IntegerModRing(n)
a=[[1,-1],[-1,1]]
h=[p,q]
solutions=[]
l=[]
for i in range (len(a)):
    x=crt(a[i],h)
    solutions.append(Zn(x))
    l.append(n.gcd(x-1))
solutions,l
```

Listing 3.6 – Simulation de la factorisation de n en utilisant le lemme et le théorème chinois

([1272, 805], [31, 67])

Donc si $y^t = 1$ ou $y^{2^{j-1}t} \equiv -1 \pmod{n}$, on doit tirer un nouveau y.

Partage du module RSA

2.Attaques éventuelles

2.2. Cas où $ed - 1 = \alpha\phi(n)$ avec $\alpha \in \mathbb{Z} \setminus \{1\}$

Décryptage d'un message par un utilisateur espion

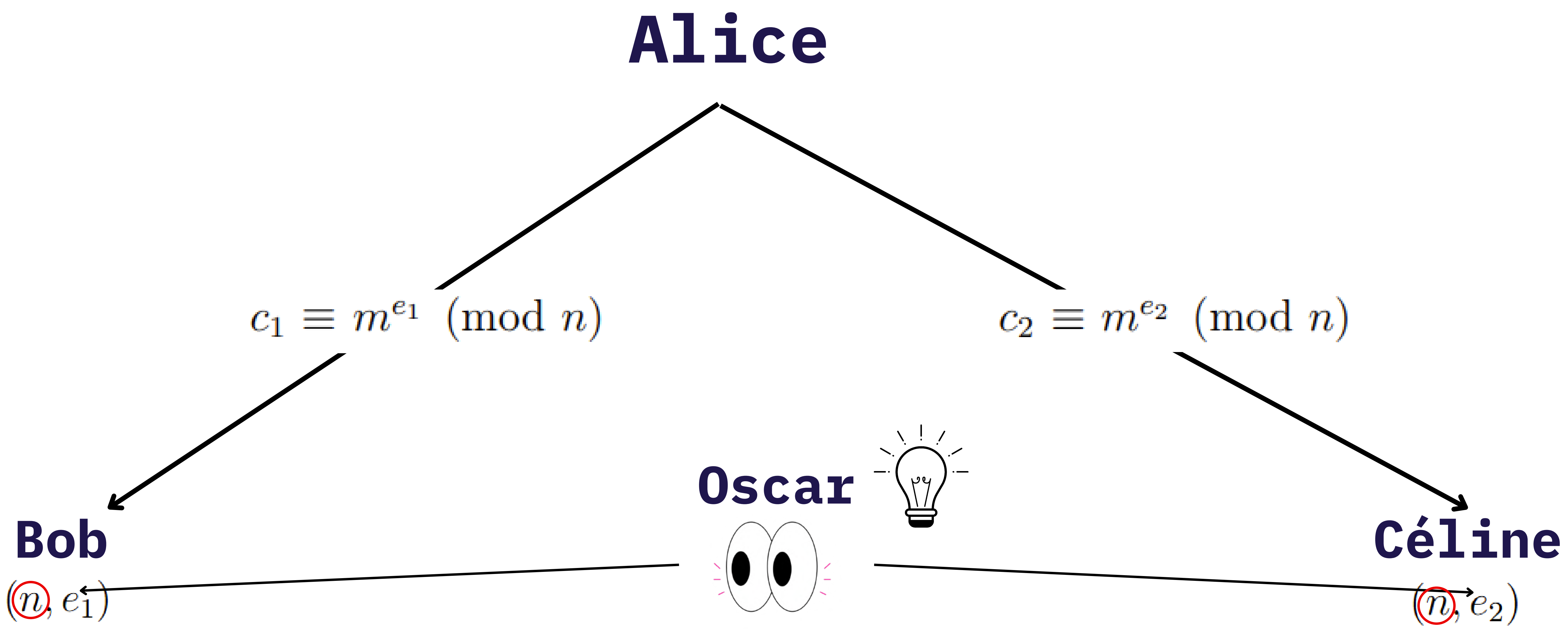
```
a = generate_shared_rsa(2048)
n, e1, d1 = a[0] #Cles de l'utilisateur 1
n, e2, d2 = a[1] #Cles de l'utilisateur 2
def attack(c):
    #Factorisation de n par l'utlisateur 1 (espion) et calcul de
       phi_n qui lui permettera de trouver les cles de
       dechiffrement de tous les autres utilisateurs
    p_trouve = factor_from_private_key(n, e1, d1)
    q_trouve = n / p_trouve
    phi_n=(p_trouve-1)*(q_trouve-1)
    d22=e2.xgcd(phi_n)[1] # Calcul de la cle de dechiffrement de
       l'utilisateur 2
    m=c^d2
    return phi_n,d22,m
b=10^60
Zn=IntegerModRing(n)
b=Zn(b) # message qu'Alice veut envoyer a l'utlisateur 2
c2=b^e2 # message crypte
phi_n,d22,m=attack(c2)
Zphi_n=IntegerModRing(phi_n)
print(Zphi_n(d2)-Zphi_n(d22),b-m)
```

(0, 0)

Partage du module RSA

2. Attaques éventuelles

2.3. Cas particulier où $(e_1, e_2) = 1$ et $m_1 = m_2$



Partage du module RSA

2. Attaques éventuelles

2.3. Cas particulier où $(e_1, e_2) = 1$ et $m_1 = m_2$

Démonstration :

On a $ue_1 + ve_2 = 1$, (théorème de Bezout)

Donc $$c_1^u \times c_2^v \equiv ((m^{e_1})^u \pmod{n} \times (m^{e_2})^v \pmod{n}) \pmod{n}$$
$$\equiv m^{ue_1+ve_2} \pmod{n}$$
$$\equiv m \pmod{n}$$

Partage du module RSA

2.Attaques éventuelles

2.3. Cas particulier où $(e_1, e_2) = 1$ et $m_1 = m_2$

Simulation de l'attaque

```
import time
e1=l[0] ; e2=l[1]
m=3*2^(1023)
Zn=IntegerModRing(n)
m=Zn(m)
c1=m^e1
c2=m^e2
start =time.time()
u,v = e1.xgcd(e2)[1],e1.xgcd(e2)[2]
w=c1^u*c2^v
end=time.time()
m,w,m-w,end-start
```

(26965397022934738615939577861835371004269654684 1...,
26965397022934738615939577861835371004269654684134...,
0,
0.0055768489837646484)

CHANCES DE RÉUSSITE

Quelles sont nos chances de réussite?

L'algorithme précédent s'appuie sur le choix aléatoire de $y \in \mathbb{Z}/n\mathbb{Z} \setminus \{0\}$.

On arrive à factoriser **n** dans les cas suivants :

Cas 1 : si on trouve $\mathrm{pgcd}(y, n) \neq 1$

- La probabilité **P** de réalisation de ce cas est égale à $\dfrac{p+q-2}{n-1}$

  Pour p = q = 10^{100}, comparables aux tailles réelles utilisées, on obtient : **P** $\approx 2.10^{-100}$

- La proba de gagner la loto est de l'ordre de 10^{-7}
- La proba d'être frappé par la foudre en un an est de l'ordre de 10^{-6}

➩ Ce cas est **extrêmement** improbable.

Quelles sont nos chances de réussite?

Cas 2 : si **y** vérifie les conditions suivantes :

a) $y^t \not\equiv 1 \pmod{n}$

b) $u^{2^j-1} \not\equiv -1 \pmod{n}$ avec $\mathrm{u} = \mathrm{y}^t$

➡ Quelle est la probabilité que **y** vérifie les deux conditions ?

Probabilité que y vérifie a)

Supposons qu'on a trouvé **y** vérifiant : $y^t \equiv 1 \pmod{n}$

Soit $\epsilon_1 \in \mathbb{Z}/n\mathbb{Z}$ tel que $\begin{cases} \epsilon_1 \equiv 1 \pmod{p} \\ \epsilon_1 \equiv -1 \pmod{q} \end{cases}$ (Th. des restes chinois)

Considérons les éléments suivants : $y, -y, \epsilon_1 y$, et $-\epsilon_1 y$, on a donc :

$$\begin{cases} (-y)^t \equiv (-1)^t \pmod{p} \equiv -1 \pmod{p} \\ (-y)^t \equiv (-1)^t \pmod{q} \equiv -1 \pmod{q} \end{cases} \implies (\text{-y})^t \not\equiv 1 \pmod{n}$$

$$\begin{cases} (\epsilon_1 y)^t \equiv (\epsilon_1)^t \pmod{p} \equiv 1 \pmod{p} \\ (\epsilon_1 y)^t \equiv (\epsilon_1)^t \pmod{q} \equiv -1 \pmod{q} \end{cases} \implies (\epsilon_1 y)^t \not\equiv 1 \pmod{n}$$

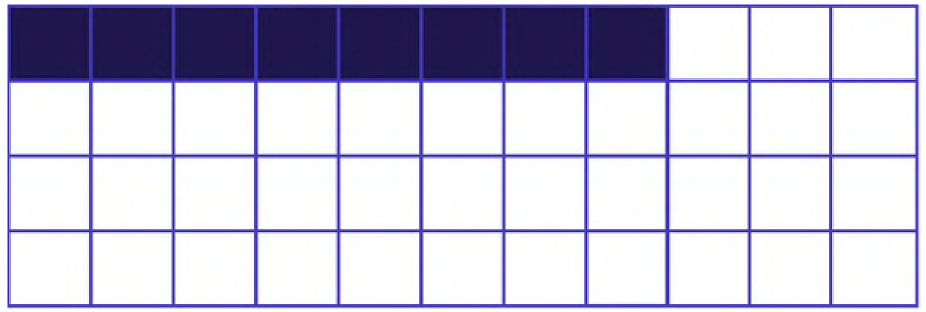
$$\begin{cases} (-\epsilon_1 y)^t \equiv (-\epsilon_1)^t \pmod{p} \equiv -1 \pmod{p} \\ (-\epsilon_1 y)^t \equiv (-\epsilon_1)^t \pmod{q} \equiv 1 \pmod{q} \end{cases} \implies (\text{-}\epsilon_1 y)^t \not\equiv 1 \pmod{n}$$

Condition a)

D'où on tire : $y^t \equiv 1 \pmod{n} \implies \begin{cases} (\epsilon_1 y)^t \not\equiv 1 \pmod{n} \\ (-y)^t \not\equiv 1 \pmod{n} \\ (-\epsilon_1 y)^t \not\equiv 1 \pmod{n} \end{cases}$

D'où $P(y^t \equiv 1 \pmod{n}) = \frac{1+1+1+\dots}{4+4+4+\dots+R} \leq \frac{1}{4}$

selon le principe des clusters :

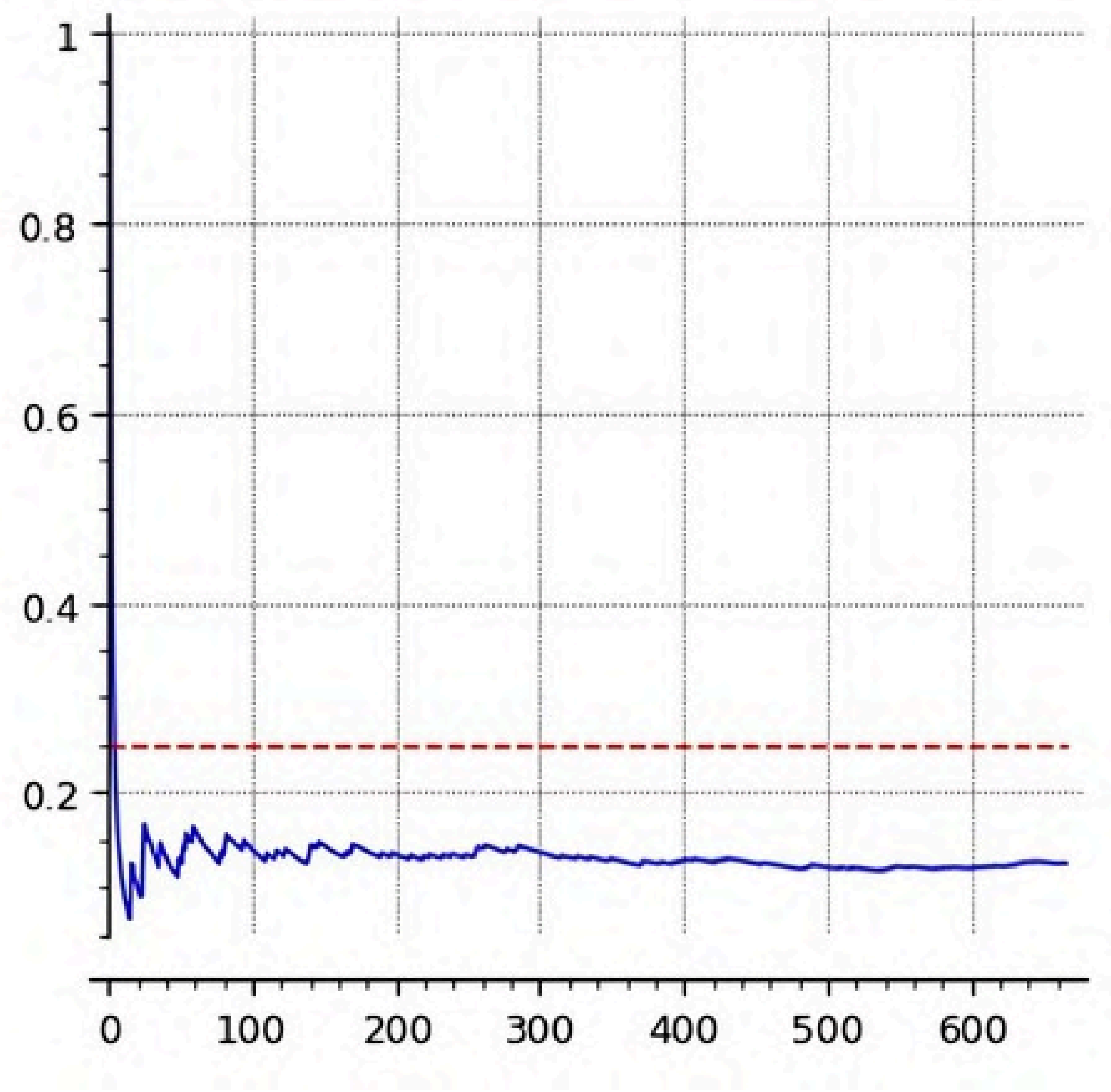


...

Condition a)

Approximation de la probabilité par la méthode de Monte-Carlo :

Probabilité



Nombre de
y testés

Probabilité que y vérifie b)

Supposons qu'on a trouvé **y** vérifiant : $u^{2^{j-1}} \equiv -1 \pmod{n}$

Soit $\varepsilon \in (Z/nZ)^*$ tel que : $\begin{cases} \mathrm{ord}_p(\varepsilon) = 2^j \\ \varepsilon \equiv 1 \pmod{q} \end{cases}$

Posons $y_0 = y\varepsilon$ et $u_0 = (y_0)^t = u\varepsilon^t$, on a alors :

$$\begin{aligned} u_0^{2^{j-1}} &\equiv u^{2^{j-1}} \varepsilon^{2^{j-1}t} \pmod{p} \\ &\equiv -\varepsilon^{2^{j-1}t} \pmod{p} \end{aligned}$$

Comme 2^j ne divise pas $2^{j-1}t$ (car t est impair), $\varepsilon^{2^{j-1}t} \not\equiv 1 \pmod{p}$

Donc $u_0^{2^{j-1}} \not\equiv -1 \pmod{p}$, comme $u_0^{2^{j-1}} \equiv 1 \pmod{q}$

Alors : $u_0^{2^{j-1}} \not\equiv -1 \pmod{n}$

Condition b)

Approximation de la probabilité par la méthode de Monte-Carlo :

Probabilité

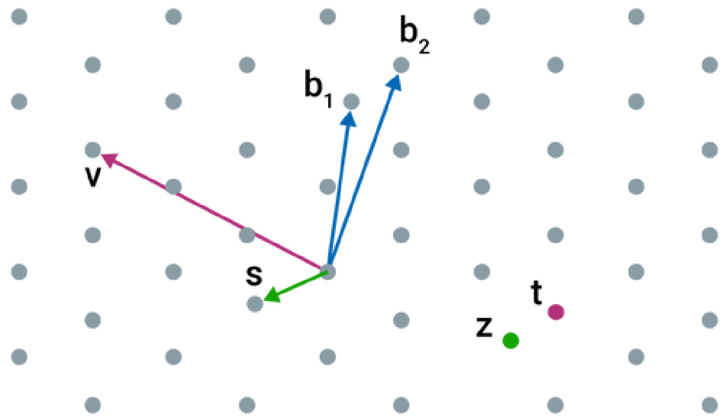
Nombre de
y testés

Conclusion :

En moyenne, en O(1) tirages aléatoires de y, la stratégie décrite fournit une factorisation de n, mettant en évidence la vulnérabilité intrinsèque du partage de module. Par conséquent, cette pratique est à proscrire absolument

Perspectives

Ordinateurs quantiques



Puces quantiques
(QRNG, QKD)

Cryptographie basée sur les réseaux
(Lattice-based Cryptography)

"En cryptographie, il n'y a pas de sécurité parfaite, seulement des probabilités d'échec très faibles."

"Le message le plus sûr est celui qui n'existe pas."

Merci de votre attention

GAME

WHO WILL BE THE FASTEST SPY ?

GAME

Autorité centrale

91,35,d1

91,5,d2

91,49,d3

91,41,65

Utilisateur 1

e1=59

c1=28

Utilisateur 2

e2=31

c2=84

Utilisateur 3

e3=41

c3=63

Utilisateur 4 (Espion)

e4=65

d4=41

n=91