Faiblesses de configuration sur RSA

Sommaire

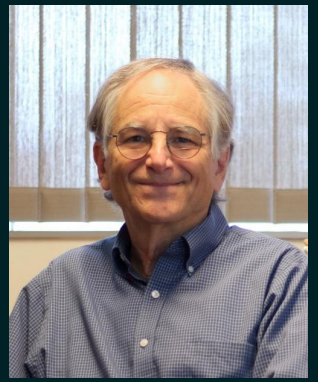
Introduction

Création

Algorithme de chiffrement créé en 1977

Breveté en 1983

Déclassifié en 1997



Rappel algorithmique

PGCD étendu et équation diophantienne : 121*u + 71*v = 1

121 = 71*1 + 50 ⇔ 50 = 121 - 71*1
71 = 50*1 + 21 ⇔ 21 = 71 - 50*1
50 = 21*2 + 8 ⇔ 50 - 21*2
21 = 8*2 + 5 ⇔ 5 = 21 - 8*2
8 = 5*1 + 3 ⇔ 3 = 8 - 5*1
5 = 3*1 + 2 ⇔ 2 = 5 - 3*1
3 = 2*1 + 1 ⇔ 1 = 3 - 2*1

1 = 3 - 2
1 = 3 - (5 - 3) ⇔ 1 = 2*3 - 5
1 = 2*(8 - 5) - 5 ⇔ 1 = 2*8 - 3*5
1 = 2*8 - 3*(21 - 8*2) ⇔ 1 = 8*8 - 3*21
1 = 8*(50 - 21*2) - 3*21 ⇔ 8*50 - 19*21
1 = 8*50 - 19*(71 - 50) ⇔ 27*50 - 19*71
1 = 27*(121 - 71) - 19*71 ⇔ 121*27 - 46*71

u = 27 et v = -46
est une solution

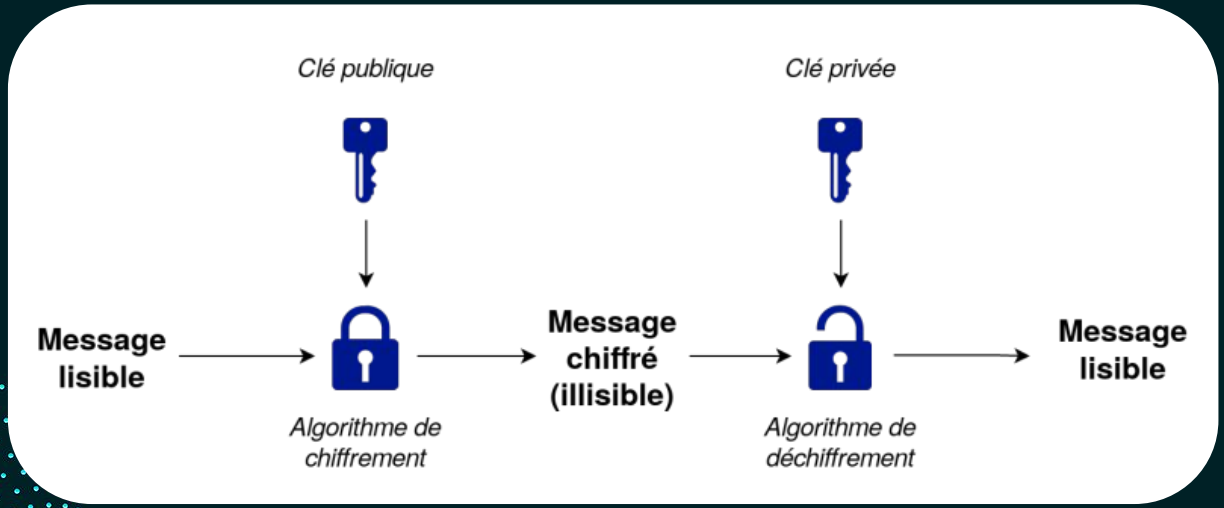
Fonctionnement

Type de chiffrement

Chiffrement asymétrique

Clé publique pour chiffrer

Clé privée pour déchiffrer

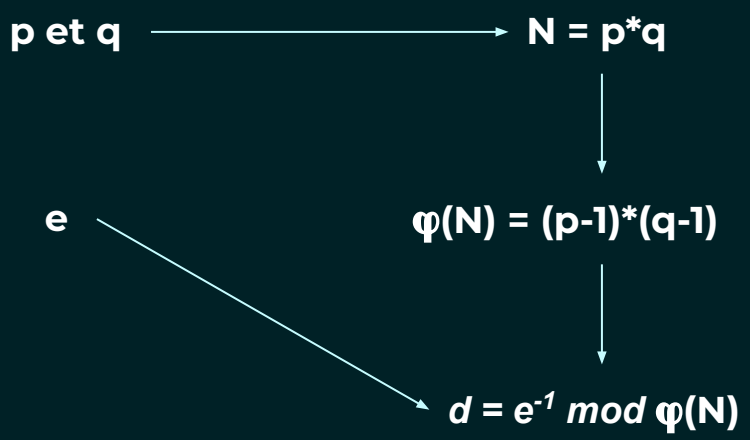


Fonctionnement

Arithmétique

Clé publique (N,e)

Clé privée (N,d)



e premier avec φ(N) ⇔ pgcd (e,φ(N)) = 1

Arithmétique

Chiffrer M ⇒ **$C = m^e \bmod (N)$**

Déchiffrer C ⇒ **$M = C^d \bmod (N)$**

Exemple :

M = 10, N = 39, e = 4, p = 3, q = 13

Chiffrement :

$C = 10^5 \bmod (39) = 4$

Déchiffrement :

$\varphi(39) = (3-1)*(13-1) = 24$

$d = 5^{-1} \bmod \varphi(39) = 5$

$M = 4^5 \bmod (39) = 10$

Implémentation

Mise en place en python

```python
from Cryptodome.Util.number import *

m = bytes_to_long(b"Mais c'etait sur en fait !!!")

p = getPrime(2048)
q = getPrime(2048)
n = p*q
e = 65537

c = pow(m,e,n)

phi = (p-1)*(q-1)
d = inverse(e,phi)

print(long_to_bytes(pow(c,d,n)))
```

Certificat clé publique/privée

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAmeL3lL+MSp/z5lM2+hfI
hGAenybsOKFzHmV29+OSGEBCb1mzSM2l0BpVSPWp8/f8C2GWQQMLjulaKsjCCn17
mA3YMaJCDi8PklzHB6rPHVc2yKzhZAkMQ9lVXxasBBsULra1Ybn0Ii4BjeGDjr6V
p5c1gpxZMzcjJJG47qGJdbtVgKa+yKobg/dTCkTaJK8eeFaRWYURdCCzh6ai6PRt
vRGE579YvBdtUABjiI098v4iFhiVpNfIzvPS2I/JgUKHD5AkhxcbhcwT94APCzqT
WHTn2S9gSVcXfiq319+L1IT8OI4F56LMzW0bIn1CiYr0KrQe7O3osABctyoxDI84
lQIDAQAB
-----END PUBLIC KEY-----
```

Où sont les nombres ? encodés en base 64

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAmeL3lL+MSp/z5lM2+hfIhGAenybsOKFzHmV29+OSGEBCb1mz
SM2l0BpVSPWp8/f8C2GWQQMLjulaKsjCCn17mA3YMaJCDi8PklzHB6rPHVc2yKzh
ZAkMQ9lVXxasBBsULra1Ybn0Ii4BjeGDjr6Vp5c1gpxZMzcjJJG47qGJdbtVgKa+
yKobg/dTCkTaJK8eeFaRWYURdCCzh6ai6PRtvRGE579YvBdtUABjiI098v4iFhiV
pNfIzvPS2I/JgUKHD5AkhxcbhcwT94APCzqTWHTn2S9gSVcXfiq319+L1IT8OI4F
56LMzW0bIn1CiYr0KrQe7O3osABctyoxDI84lQIDAQABAoIBAA8xe/05Ljo98aA3
bhGiIH+FV/ChsKfktM92+BiAZO+1BHom8o8JEspplysFajq2rLFY+LNBgDYjbu4T
dSkA+59nGoXH/dqVpeVu+I2WvS0p67RDcVtlk0rtncKuI9yh6cu06VKVP/fR8ABY
mBsTUQZDZRqYoivu0S7unkIdO8ueRKDtVgmHo7VOg48LMT9ZpDeCfySwVHpjx8pz
uEJx0vcvvNViL+/PFMNWd3/LPY83jzmkc3ul1k8587/b+ktygk+KXI1Y2qXw18ad
ziEabC2Qxq1U7LgGSjODapoRLsFaLojWWTuF2w533QxJJ/2mYZbD0nI/37umX11O
dkcYXhECgYEAxiurg1IdScUI/jfkKDNF9txs2FJ+o061MbGV36mjMmxcRSh/nM2S
b4gt611qiOgED+joYkmwT2FDpkHf1maGcTOlqSURiQcS6w0M3vc47dFX4ahXT+j7
YPBoj2fZ2kPIwrxL1wYZSd2NLNUGNyyI845SOL4Fte6/Ojvp71D1dhkCgYEAxssP
4t1Kj0e3H/Y2ghalIaD9EEd1p5WsTl+wZVDDBZYxxsZZk52wbhEgHtyhegPKjoTL
0kL802ooGBoSaV6idRaLEb0A2DIHSl5y68EjIGUR+ebf72vkD+W+l7ZlSaYQFdZa
DNiX19qIdC6OrExbOt0Dlu7wWWNsfxgXG1BhDd0CgYEAgRqiOoOcoRrEqRowgiAl
TJZ+MtBLwIjecumHApe5wnfmBQhFWxpydEb744KQjXDadKMAAkiPnKd5My9nQsy4
ebd0DaYgrmz1Xjy6TegyJn1kC5doip72Zkk5VXQK8a+Hc5bIiDNo9j5oYvTFf9x7
vHDUEDN79PL9kirGVSsW0sECgYAJo8svax+0dvvQVvYptMNpMA9VfMDXMZDbdBRv
tUzftnrPcsrhN6JYlAgEtxo1en2WD6Z6mbj834b2ND10Upx6JQ+1Wn7x/OsRlibo
e8MiTQRkI+f0Cj1S8AcgAD5FJgJdrL1NKiuft38qcx+EB2fdIN6NsSSCkSmvuBFW
x70NGQKBgHyLqjCPebQuFbJ3XJ+i+FvUw3Nr6Iwcpi83NdEwoRpSXwtk0T7C0yV3
9LhvInrA9yCFQl6ePGuu8x9hEWF4zrG7ThowrplWgvutYqdmp0407c19znn3XhriZQrVUS8YmML1DlGKHK2GB5NxItZ0CFZOA085uQBoxMZowqkWq387
-----END RSA PRIVATE KEY-----
```

```
RSAPrivateKey ::= SEQUENCE {
    version           Version,
    modulus           INTEGER,  -- n
    publicExponent    INTEGER,  -- e
    privateExponent   INTEGER,  -- d
    prime1            INTEGER,  -- p
    prime2            INTEGER,  -- q
    exponent1         INTEGER,  -- d mod (p-1)
    exponent2         INTEGER,  -- d mod (q-1)
    coefficient       INTEGER,  -- (inverse of q) mod p
    otherPrimeInfos   OtherPrimeInfos OPTIONAL
}
```

Utilisation

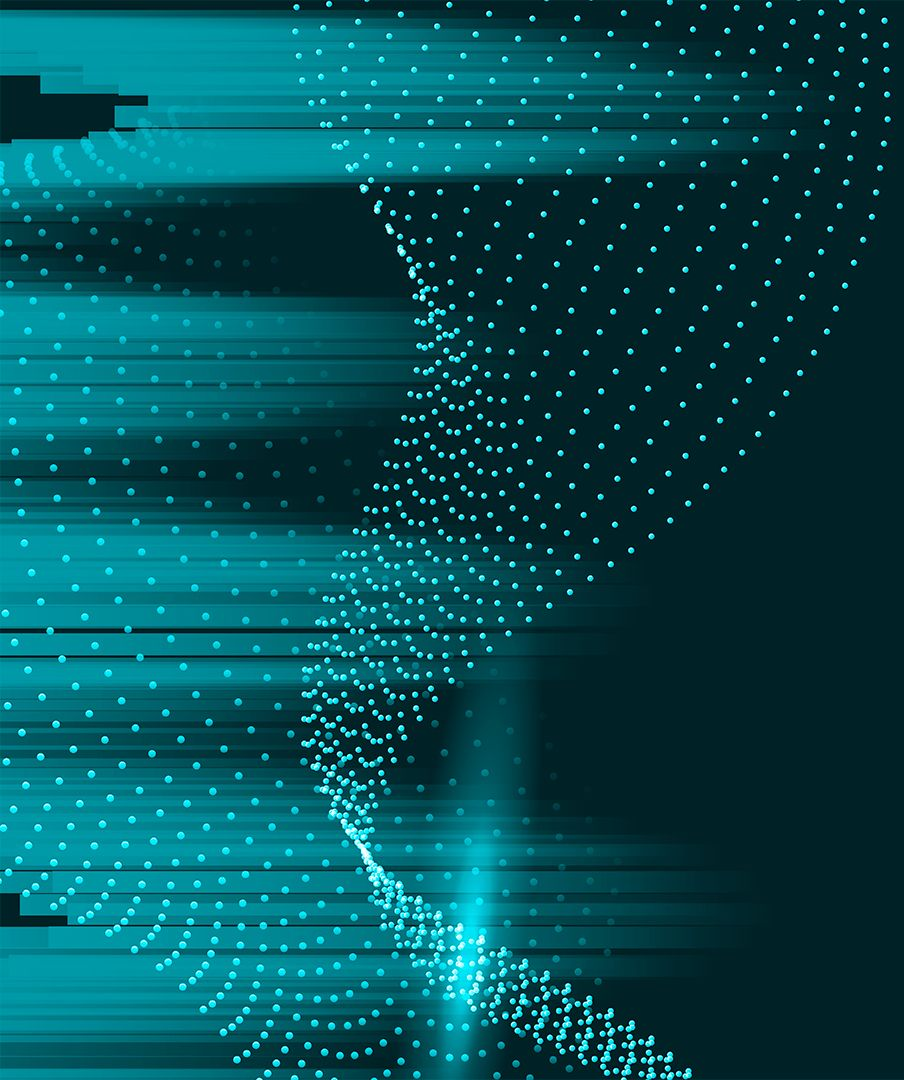
Connection SSH (Secure Shell) sans mot de passe

Sécurisation d’échanges bancaires

Vérification d’échanges avec signature du RSA (pas dans cette présentation)

...

Attaques



Factorisation

Décomposition de N

Oracle de déchiffrement

Homomorphisme sur RSA

Exposant Faible

Racine n-ième

Module Commun

Arithmétique power

Factorisation

Soit N = 91

Que valent p et q ? 13 et 7

Attaque facilement envisageable pour des tailles de N de moins de 39 chiffres

100 000 000 000 000 000 000 000 000 000 000 000 000

Oracle de déchiffrement (CTF like)

Oracle: algorithme permettant de déchiffrer des messages

Homomorphisme du RSA:

C(a*b) = C(a)*C(b) et M(a*b) = M(a)*M(b)

En décomposant C en produit de deux facteurs on revient donc à une forme C(a*b).

C(a*b) ne peut pas être déchiffré par l'oracle mais C(a) et C(b) oui.

On obtient M(a) et M(b) donc on retrouve M(a*b)

Exposant Faible

On pose N = 4512 … 157 (très grand nombre)

Soit e = 3 et m = 12

$c = 12^3 \bmod 4512 \ldots 157 = 1728 \bmod 4512 \ldots 157$

Or c << N (“Nous n’avons pas fait un tour du modulo”)

Donc $1728 \approx m^3 \Leftrightarrow m = \sqrt[3]{1728}$

Il faut toujours prendre e assez grand. Souvent e = $2^{16} + 1 = 65537$

Module Commun - 1

Soit N1 = 39 et N2 = 15

N1 et N2 ont un facteur commun qui est 3

Pour le trouver on utilise le PGCD tel que p = pgcd(N1,N2)

N1 = p*q ⇔ q = N1/p

N2 = p*r ⇔ r = N2/p

Module Commun - 2

Qui a dit qu'il n'y avait que p en commun ?

$c1 = m^{e1} \bmod N$

$c2 = m^{e2} \bmod N$

Avec l'algorithme d'Euclide étendu on trouve s1 et s2 tel que:

$s1*e1 + s2*e2 = 1 \bmod N$ (équation diophantienne)

On a donc $c^{s1} * c^{s2} = m^{(e1*s1)} * m^{(e2*s2)} \bmod N$

$\Leftrightarrow m^{(e1*s1 + e2*s2)} \bmod N = m \bmod N$

Encore d’autres attaques

Fractions continues, Håstad, ...

Conclusion

Conclusion ...

Cryptosystème sécurisé grâce à l'utilisation astucieuse des propriétés mathématiques

Majorité des failles sont dû à des erreurs d'implémentation souvent humaine:

- réutilisation de facteurs

- nombre trop petits

...

... et limites

Cependant la puissance grandissante des ordinateurs et l'arrivée du quantique dans l'informatique met à mal le RSA.

Factorisation de plus en plus simple

Sources

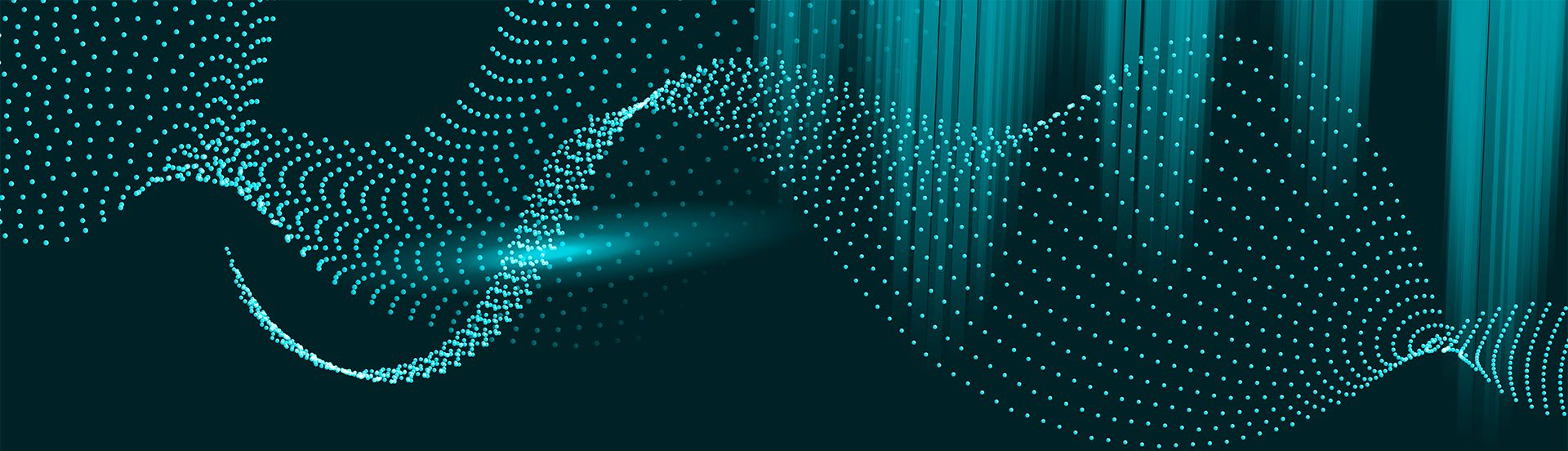
https://www.root-me.org/

BlackRaven#8794

https://www.youtube.com/watch?v=RyMmKoSSPN8
(série de vidéo d'Express sur le RSA)

https://cryptohack.org/

Antonin



Des Questions