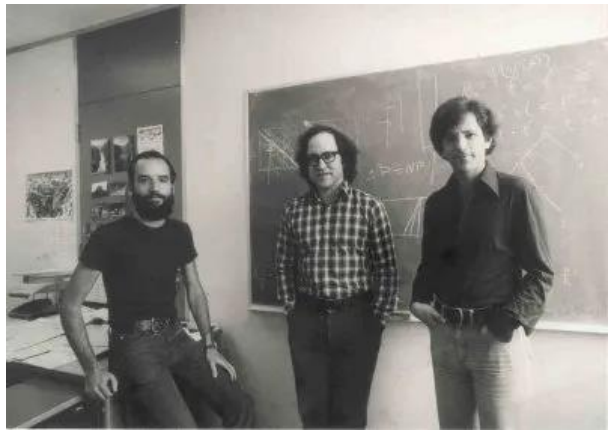
RSA Rivest Shamir Adleman



- Créé en 1977 par : Ron Rivest, Adi Shamir et Leonard Adleman
- Algorithme de chiffrement asymétrique le plus connu.
  - Chiffré avec une clé et déchiffré avec l'autre.
  - RSA crée un paire de clé « commutative ».

RSA

Facteur

Nombres qui divisent un nombre sans laisser de reste

Facteurs de **12** :
1 2 3 4 6 12

Facteurs de **7** :
1 7

Nombre premier

Nombre dont les facteurs sont 1 et lui-même

2 3 5 7 11
13 29 37 61

Nombre divisible par seulement 1 et lui même

Semi-premier

Nombre dont les facteurs sont des nombres premiers

Facteurs du semi-premier 21 :
1 **3** **7** 21

produit de deux nombres premiers

Modulo

Reste d'une division

13 MOD 5 = 3

21 MOD 5 = 1

25 MOD 5 = 0

RSA : Exemple

- **Génération de clés :**
  - Sélection de 2 nombres premiers : (P,Q)
  - Calcul du produit : N=(P×Q)
  - Calcul de l'indicatrice d'Euler : T = (P-1) ×(Q-1)
  - Sélection d'un exposant public : (E)
    - Doit être un nombre premier
    - Doit être inférieur à l'ind. d'Euler
    - Ne doit pas être un facteur de l'ind. d'Euler
  - Sélection d'un exposant privé : (D)
    - Produit de D et de E divisé par T.
      Le reste doit donner 1.
    - (D × E) MOD T = 1.
- La **clé privée** est constituée du couple (N,D) et la **clé publique** du couple (N,E).

premier	**P Q**	**7 19**
produit	**N**	**133**
Euler	**T**	**108**
Exp public	**E**	**29**
Exp privé	**D**	**41**

RSA : Exemple

- **Chiffrement et déchiffrement :**
  - **Chiffrement :**

    Message ^ E mod N = texte chiffré
  - **Déchiffrement :**

    Texte chiffré ^ D mod N = Message

Message
60

premier	P Q	7 19
produit	N	133
Euler	T	108
Exp public	E	29
Exp privé	D	41

- **Chiffrement avec clé publique et déchiffrement avec clé privée**

  (60 ^29) mod 133 = 86

  (86 ^41) mod 133 = 60

- **Chiffrement avec clé privée et déchiffrement avec clé publique**

  (60 ^41) mod 133 = 72 Message ^ D mod N = texte chiffré

  (72 ^29) mod 133 = 60 Texte chiffré ^ E mod N = Message

Utilisé pour signer (authentification)

RSA EN PYTHON AVEC LA BIBLIOTHÈQUE SYMPY

```
import random
import sympy

# Sélection de deux nombres premiers
   P = sympy.randprime(10, 50)
   Q = sympy.randprime(10, 50)

   # Calcul du produit N
   N = P * Q

   # Calcul de l'indicatrice d'Euler
   T = (P - 1) * (Q - 1)

 # Choisir un exposant public E (doit être premier et non diviseur de T)
   E = 3
   while sympy.gcd(E, T) != 1:
      E += 2  # Prendre un autre nombre premier

   # Calcul de l'exposant privé D
   D = pow(E, -1, T)  # Calcul de l'inverse modulaire
```

Sélection de 2 nombres premiers : (P,Q)

Calcul du produit : N=(P×Q)

Calcul de l'indicatrice d'Euler : T = (P-1) ×(Q-1)

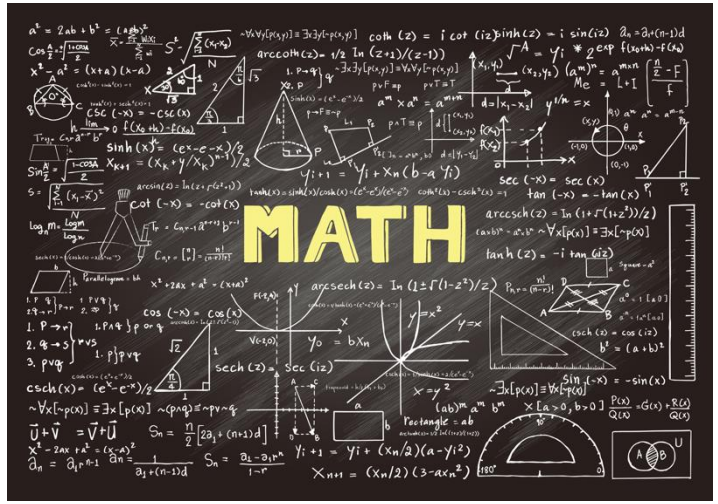
Sélection d'un exposant public : (E)

- Doit être un nombre premier
- Doit être inférieur à l'ind. d'Euler
- Ne doit pas être un facteur de l'ind. d'Euler

Produit de D et de E divisé par T.
Le reste doit donner 1.

(D × E) MOD T = 1.

RSA EN PYTHON AVEC LA BIBLIOTHÈQUE SYMPY



pow(message, E, N) ← Message ^ E mod N = texte chiffré

pow(ciphertext, D, N) ← Texte chiffré ^ D mod N = Message

GÉNÉRER UNE PAIRE DE CLÉS RSA AVEC OPENSSL

1. Générer une clé privée RSA (2048 bits) :

openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048

2. Extraire la clé publique depuis la clé privée :

openssl rsa -pubout -in private_key.pem -out public_key.pem

C'est du base 64 :

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAw8mWTBzlLjOQUc6u2Tjx
C37excmFc/DinUiG34HYTeQ4oSzg6jCISG6OF0j+BKIdTR433MowqZ/nhbYMggPg
LYxotK32rNz6q+MGXd8vy2DZGuXjNazcGU6pdezwPc74wPBLvVJ+uHELFkD4uhNn
I1OLXuZVRJ6R6LN9ArFwt2/7pwJMlpjZfUvcKRZ3SZ9dy2/tn+/f4xTqzNj0mNkB
HKEI22vNwvuGac4EMYQEnd2g7xsr39lZxEXURBLJvvUFjSOmdh65+29xSOlz3xh+
iOO0ZQE040JAXVt/spxqGMNGAHeYWDfbK+ux14lQKffiCZoUI5YkOfvXBOOR3Wk6
YwIDAQAB
-----END PUBLIC KEY-----
```

```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDDyZZMHOUuM5BR
zq7ZOPELft7FyYVz8OKdSIbfgdhN5DihLODqMIhIbo4XSP4Eoh1NHjfcyjCpn+eF
tgyCA+AtjGi0rfas3Pqr4wZd3y/LYNka5eM1rNwZTql17PA9zvjA8Eu9Un64cQsW
QPi6E2cjU4te5lVEnpHos30CsXC3b/unAkyWmNl9S9wpFndJn13Lb+2f79/jFOrM
2PSY2QEcoQjba83C+4ZpzgQxhASd3aDvGyvf2VnERdREEsm+9QWNI6Z2Hm7b3FI
6XPfGH6l47RlATTjQkBdW3+ynGoYw0YAd5hYN9sr67HXiVAp9+IJmhQjliQ5+9cE
45HdaTpjAgMBAAECggEARx8wsN+CRWFH2N/q1lAACDwDMvg8uVxVBeu1yvWBHPJ9
u9bwsUIGD8HRbhX+6LH3UO9cPaFZQhUrZoA2VPiRdBpgYtobFLqFePFUe786vnel
BkdiXTo5DhxsJMF4mAtag8QGz3RrQhcWfvQFJpsryUJtC4FCklXEDTM+gz7wAkJS
DWC3FbYmCX0PjLSwSrzc2PZ8AFgwpD7SSjFaVU1YajqDfAggwjiyMIeSc4ViknKl
LpZtRBumfANpmoBmKgb8n/h6E4bKKRW2AAJt+Y+ixHUBtr0z0ep8kCY7AXIzsiV2
sxgfQePveqLheswkjiYPrtaZpugjkbnJHCXHoNZGuQKBgQDjlgD2U2GqnFLijjAP
yzbDhwsorHknkLAXGkaPsIN1IfHqdM4VKW4J2d14VEPsBvybnd3X7I1OCq1SO+Cn
f0UVVxL1nucl87HZHt2jvC90lCjOlXDOXulHAIG4whhKKHdkzxU81gSOaLK7ABGX
NLD6QDkit2C021HAjN3V7W9yPwKBgQDcq7rS2sdPZnNcYoTMTMU8D49UpabnrIsP
McvMrdctzs0fg2gLLh+5j/l1/cqH7d6a9J3AvrGPUJTo5cDlZGWYSIF4HiryWO2d
89L+XT1RhyLxeE2g0blOacc3LGtsW4OlqFOcclki3Lmf0emP68Z7g6x9Rcd9ZLFv
Fg/+urHm3QKBgQChzPFeiFuEzEN0t0WpmpvygdsDKlpOExY9/uvRJG8hHANXf8nm
7VKEFf1c4LpaA0gkGzuTu+GVfKmncRxCdbli0FqjOoVzVbyV7RT6tO4hYAIJ2A86
FAaeG9BNK4deHKGg4JxCcDtir1bTO+NMEa8ZNnL9NiDf9XSxdCf7Y18GxQKBgQCY
lLmaQCleGaXclFZkeRTLwaqGif/NIzYH7JORVAZ8KwkUKCVRzbOr+AwydKiMsoIA
PPGzK2fUw5AjNXHaa0AAOuoipyFk7u5mEAitTsUuUTVtJ5TthPYPbP6NfM5OXzy4
15nwi/RsMZi8FEXd+NYF43bqK8yTPIOfXu7yYB7QlQKBgClKRNnuIpehKlqVYdZa
fX2R1nn+z1FLrnqgLuOfmt0AbuApixeFY2TOHL0kEKq8F5zkuUHO7Y2DqwCqOgdg
YTbeOYKyeS2JIou3cvqDbl5szZk5Q1/uMoVzBdaASVryz6WyzBjWpPGlYItQm1HE
7kmY3tlpZPUXu+WiOCC64rXG
-----END PRIVATE KEY-----
```

🔐 Chiffrer avec une clé publique RSA :

openssl rsautl -encrypt -pubin -inkey public_key.pem -in message.txt -out message.enc

- encrypt : Mode chiffrement
- pubin : Indique qu'on utilise une **clé publique** pour chiffrer
- inkey public_key.pem : Clé publique utilisée pour chiffrer
- in message.txt : Fichier contenant le message à chiffrer
- out message.enc : Fichier contenant le message chiffré

🔒 Déchiffrer avec la clé privée RSA

openssl rsautl -decrypt -inkey private_key.pem -in message.enc -out message_dechiffre.txt

- decrypt : Mode déchiffrement
- inkey private_key.pem : Utilisation de la clé privée pour déchiffrer
- in message.enc : Fichier chiffré en entrée
- out message_dechiffre.txt : Fichier contenant le message déchiffré

Pour lire la clé publique au format texte : **openssl rsa -pubin -in public_key.pem -text -noout**

publicExponent: 65537

privateExponent:
02:2e:66:12:6d:63:68:29:ed:98:66:b6:db:d2:ba:
84:1d:8a:42:de:2d:2e:f8:c8:52:1a:34:fe:11:af:
7d:a3:5d:dd:06:38:f7:ef:ff:4b:6c:e2:a9:c5:19:
f2:7c:12:04:b5:58:59:7e:6d:b7:1a:ef:ce:8e:c5:
6f:c5:d8:a4:bf:ae:a3:7b:17:13:0b:0c:b1:48:d4:
ce:c3:11:4e:e3:28:8a:6d:fb:88:ce:cc:fa:7c:3f:
6d:cf:5e:a5:39:68:af:b9:cf:35:54:0d:8f:33:63:
64:c7:85:ae:de:f5:a8:8d:93:7e:1e:28:06:2e:84:
e7:1b:c8:70:0d:74:07:95:bf:45:6a:84:74:7f:33:
55:b7:78:27:1c:cb:e8:41:ce:ee:c5:ed:f4:24:f5:
90:5f:fa:e7:dc:8b:08:3d:5b:60:45:95:6f:70:d1:
6d:53:ac:71:3c:b1:32:45:90:7a:73:90:df:05:b9:
0f:87:5d:dc:72:60:a2:02:26:eb:4b:5e:f3:02:de:
c5:3c:00:2f:72:cc:7b:ee:10:30:e5:ba:75:d5:8b:
41:30:8b:fa:68:51:d0:c8:b5:6d:78:2f:02:e9:4b:
0b:6c:2c:0e:d6:35:4a:0a:6f:44:99:84:75:8d:91:
54:cb:07:2d:22:da:3d:e4:a2:07:1a:9a:58:37:f0:
2f

premier	**P Q**
produit	**N**
Euler	**T**
Exp public	**E**
Exp privé	**D**

Pour lire la clé privée au format texte : **openssl rsa -in private_key.pem -text -noout**

Prime1:
00:bf:b8:22:7f:70:58:ae:72:8d:e8:0c:03:b7:d5:
ec:79:0a:f7:32:1c:ac:c6:5f:d7:e1:48:17:3f:df:
34:93:63:a6:b5:4c:72:d8:b0:2f:7c:81:7a:12:69:
bc:41:4f:a0:6a:26:dc:b0:36:55:3d:65:b6:54:75:
44:a1:6e:1c:f9:73:8d:ef:84:ba:1b:e9:d8:05:bb:
46:14:25:d9:e1:45:f6:3a:0a:95:93:ae:14:82:89:
be:c3:12:7d:96:06:87:f3:96:fc:a7:ee:f6:49:31:
f7:5b:d9:d9:87:5f:18:be:bb:d0:30:00:a6:b3:30:
35:da:1d:67:ad:cc:1c:1b:cf

Prime2:
00:fa:94:02:9d:60:b2:1b:3c:61:26:c2:0f:78:b5:
34:26:82:b7:e3:7f:14:6c:c9:cb:00:84:ac:f5:0f:
6b:c8:e2:53:35:d6:33:bf:be:9e:9f:68:fd:ac:e9:
29:36:d5:85:e2:da:e1:b9:34:61:03:55:bb:8c:3f:
6b:45:3d:26:af:4b:2e:31:fe:94:ab:8c:11:ab:01:
e5:f4:d2:b4:3a:4e:89:07:a7:cf:3e:12:56:e3:4a:
32:3b:05:1d:8e:96:d3:2d:a5:24:a4:59:b2:7e:de:
ae:a7:d2:bb:01:1c:fc:04:b3:7a:01:aa:39:d2:d2:
ae:87:a0:4c:f3:a6:59:cc:67

premier	**P Q**
produit	**N**
Euler	**T**
Exp public	**E**
Exp privé	**D**

Modulus:
00:bb:a8:aa:19:b2:00:24:06:ba:d9:38:c7:e3:f2:
22:45:45:13:36:d2:ff:0d:88:25:9f:5e:90:d8:6d:
51:7d:d4:4d:28:13:7d:36:cc:ce:6a:8c:08:60:14:
3f:71:8f:96:b1:9b:6c:dc:fc:7f:30:21:54:81:99:
a2:49:94:93:fa:26:65:74:9f:6f:45:0e:af:c8:52:
9c:c2:4a:78:e6:4a:91:f5:5a:cd:82:5b:ce:a6:7e:
4e:c5:0b:13:35:9e:b6:c5:d6:70:d4:0d:78:cf:3e:
0b:d1:87:32:26:30:dc:a6:90:3a:f4:49:b6:56:48:
08:f8:8e:9b:2e:b5:9c:ce:c4:1a:0f:fc:79:1d:69:
f2:d2:ae:4f:a0:01:a1:a8:3c:4c:cf:1d:0d:74:ac:
d6:5d:84:37:2f:3c:cb:64:5b:28:da:bf:83:75:23:
57:74:28:ef:9a:af:f2:af:27:7f:ec:b4:cc:58:f8:
d5:fb:5c:99:80:69:72:6d:a3:85:56:c6:75:21:9c:
4c:be:3b:5f:66:28:4f:39:45:d7:21:8c:c2:ad:ea:
7a:8c:18:35:83:44:52:56:6b:69:88:e5:5c:7a:3d:
37:7f:a3:9d:a8:89:8c:3a:a1:69:ca:cc:62:43:be:
72:4b:e3:8f:da:73:57:50:6c:8f:89:3e:de:6a:6f:
24:49