RSA

Le chiffrement RSA est un algorithme de cryptographie asymétrique.Il est notamment utilisé pour échanger des données sur internet.

Fonctionnement général

Asymétrique : il utilise une paire de clé (nombres entiers) composée d'une clé publique – pour chiffrer – et d'une clé privée – pour déchiffrer – des données confidentielles.

Ainsi, Alice rend sa clé publique accessible pour recevoir des données d'autres personnes, telles que Bob, afin de chiffrer les données. Sa clé privée est, quant à elle, gardée secrète et sera utilisée pour déchiffrer les données. Cette même clé privée peut être utilisée pour signer les données qu'elle envoie (et dans ce cas, la clé publique permettra à n'importe qui de vérifier la signature). Il doit être « calculatoirement impossible » de déchiffrer à partir de la clé publique, ie, de reconstituer la clé privée à partir de la clé publique.

Le chiffrement est basé sur la congruence des entiers : on s'intéresse au module (le reste des divisions euclidiennes) ; et sur le petit théorème de Fermat (soient p un nombre premier, a un nombre entier non divisible par p ; alors a^(p-1) – 1 est un multiple de p ou a^(p-1) – 1 est équivalent à 0 mod p) afin d'obtenir des fonctions à sens unique (qui sont faciles à calculer mais très difficiles à inverser).
La clé secrète est la décomposition d'un grand nombre entier, souvent de plusieurs centaines de chiffres. On peut utiliser la factorisation d'un entier *par exemple* : Soient p et q deux entiers connus, il est facile de calculer x = pq, mais bien plus compliqué de trouver p et q en ne connaissant que x.

De manière générale, tous les calculs de clé se font modulo un nombre entier n qui est le produit de deux nombres premiers. Les messages clairs et chiffrés sont des entiers inférieurs à l'entier n. Les opérations de chiffrement et de déchiffrement, quant à elles, consistent à élever le message à une certaine puissance modulo n (soient b une base, e un exposant et m un entier, on peut calculer c tel quel : c est équivalent à b^e (mod m). Si b, e et m sont positifs ou nuls et b < m, il existe une unique solution c telle que 0 ≤ c ≤ m).

En plus des principes mathématiques, il faut prendre en compte les éléments primordiaux de la sécurité : il ne faut ABSOLUMENT pas pouvoir récupérer une clé privée à partir d'une clé publique → procédé aléatoire. De plus, les données chiffrées ne doivent pas être trop courtes, afin d'avoir un déchiffrement avec un calcul modulaire. On peut compléter avec l'Optimal Asymmetric Encryption Padding (https://fr.wikipedia.org/wiki/Optimal_Asymmetric_Encryption_Padding).

Création des clés

Les clés seront générées par un serveur et non par Alice ici. Mais il subsiste un problème : il faut que Bob soit sûr que la clé est bien celle d'Alice (n'est pas réglé par le chiffrement). Le renouvellement des clés n'intervient que si la clé

privée est compromise OU après un certain temps (plusieurs années), par précaution.

Algorithme :
1. Soient p et q deux nombres premiers que l'on choisit, distincts.
2. Soit n un nombre tel que n = p*q (appelé module de chiffrement).
3. Calcul de l'indicatrice d'Euler : IE(n) = (p-1)*(q-1). L'indicatrice d'Euler est le nombre de nombres premiers de 1 à n. Par exemple, IE(8) = 4 (1,3,5,7).
4. On choisit un entier naturel e, premier avec EI(n) et avec e < EI(n) (appelé exposant de chiffrement).
5. Calculer l'entier naturel d, inverse de e modulo EI(n), avec d < EI(n) (appelé exposant de déchiffrement). On peut utiliser l'algorithme d'Euclide étendu (https://fr.wikipedia.org/wiki/Algorithme_d%27Euclide_%C3%A9tendu) : il existe deux entiers d et k tels que e*d + k*EI(n) = 1, donc e*d est équivalent à 1 mod EI(n), et donc que e est bien inversible modulo EI(n).

Le couple (n,e) est la clé publique du chiffrement, alors que le nombre d est sa clé privée. Le déchiffrement ne demande que la clé privée d et l'entier n, connu par la clé publique.

AUTRE EXPLICATION :

1. On choisit deux nombres premiers p et q (par exemple p = 53, q = 97, dans la vraie vie, on prendrait des nombres premiers avec des centaines de chiffres).
2. Soit N tel que N = p * q, donc ici N = 53 * 97 = 5141
3. On pose l'indicatrice d'Euler M = (p-1) * (q-1). Ici M = (53-1)*(97-1) = 4992
4. Pour créer la clé publique, on choisit un nombre C qui est premier avec M. Ici, on prend C = 7 (on pourrait prendre autre chose) et on a bien PGCD(4992,7)=1.

Notre clé publique est donc composée de (N,C). Donc ici N = 5141 et C = 7

5. On va calculer un nombre U à partir de C et M. Bézout montre que deux nombres a et b sont premiers entre eux ssi il existe des solutions u et v telles que a*u + b*v = 1. On va donc chercher U (la valeur de V ne servira pas).
On commence par installer la bibliothèque GMP, Puis il faut écrire le code que j'ai trouvé en Annexe. Il installe Bézout en plus.
Pour utiliser le programme, il faut taper : ./bezout -rsa C M (donc ici ./bezout -rsa 7 4992, ce qui donne : 4279).

Donc en résumé, clé publique : (5141,7) et clé privée : (4279, 5141).

Le chiffrage

Déjà, il nous faut des nombres à la place des caractères. On a deux méthodes possibles, on choisira celle qui remplace chaque caractère par son équivalent en code ASCII afin de couvrir toutes les possibilités.
Pour chiffrer, on va utiliser des puissances et modulo de grands nombres, pour nous aider, on peut installer le programme donné en annexe B.

Etape 1 : Bob veut envoyer un message à Alice, il va donc chercher, dans l'annuaire, la clé publique d'Alice. (par exemple ici : clé d'Alice = (5141,7)
Etape 2 : On remplace les caractères du message par leur équivalent en code ASCII.
Etape 3 : On enlève chaque nombre ASCII (caractère) à la puissance C de la clé publique (ici 7 donc). (exemple : je veux envoyer « B », B=> 66 (ASCII) => 66^7)
Le logiciel fera le calcul, donc pas besoin de le faire, c'est juste à titre d'information !
Etape 4 : Le modulo. On calcule le modulo de chaque nombre avec le N de la clé publique. Dans notre exemple : B=>66=>66^7=> 66^7 mod(5141) = 386. Pour obtenir la valeur avec le programme on écrirait : ./calculs_RSA 66 7 5141.

Le déchiffrage

On a donc Alice qui vient de recevoir le message 386 de Bob. Le déchiffrage se fait de la même manière : deux calculs et un remplacement.
Etape 1 : On élève à la puissance U le nombre. Donc ici 386 => 386^4279.
Etape 2 : On applique le modulo N sur le nouveau nombre obtenu : 386^4279 => 386^4276 mod(5141) = 66.
Etape 3 : On remplace le nombre obtenu par son équivalent dans la table ASCII. 66 => B.

L'attaque de l'homme du milieu

En plus d'Alice et Bob, on ajoute un nouvel arrivant : l'espion. Il va se placer entre Bob et Alice. Il récupère la clé publique d'Alice dans l'annuaire puis s'arrange pour que Bob obtienne sa clé publique au lieu de celle d'Alice. Bob va donc chiffrer le message avec la clé publique de l'espion et l'envoyer. L'espion intercepte alors le message et le déchiffre grâce à sa propre clé privée. Une fois le message lu, il peut le chiffrer de nouveau avec la clé publique d'Alice et l'envoyer à Alice, il passe alors inaperçu.

Compléments

- La clé publique est visible par tout le monde dans un annuaire qui associe la personne avec sa clé
- Procédure d'envoi d'un message de Bob vers Alice : Bob va dans l'annuaire pour récupérer la clé publique d'Alice. Une fois possédée, il chiffre le message puis l'envoi.

SOURCES

Wikipédia : https://fr.wikipedia.org/wiki/Chiffrement_RSA#Fonctionnement_g%C3%A9n%C3%A9ral
Informations sur les attaques :
https://fr.wikipedia.org/wiki/Chiffrement_RSA#Attaques
Openclassroom : https://openclassrooms.com/courses/la-cryptographie-asymetrique-rsa/rsa-qu-est-ce-donc
Bibliothèque GMP : https://gmplib.org/
Table ASCII : http://www.asciitable.com/

ANNEXE A

```
cd ~
mkdir RSA
cd RSA
wget http://ftp.sunet.se/pub/gnu/gmp/gmp-4.2.2.tar.gz
tar xvf gmp-4.2.2.tar.gz                    //Il manque peut être le trait à xvf
cd gmp-4.2.2
./configure –-enable-cxx
make
make check
sudo make install
cd ..
rm gmp-4.2.2 gmp-4.2.2.tar.gz               //Manque le -r pour récursif peut être
cd /usr/lib/
sudo ln -s /usr/local/lib/libgmpxx.so.4 libgmpxx.so.4
cd ~/RSA
wget http://tuxweb79.free.fr/Bezout_linux.zip
unzip Bezout_linux.zip
cd Bezout
make
chmod +x bezout
```

ANNEXE B

```
cd ~
mkdir RSA
cd RSA
wget http://tuxweb79.free.fr/Calculs_RSA_linux.zip
unzip Calculs_RSA_linux.zip
cd Calculs_RSA
make
chmod +x calculs_RSA
```