Projet Cryptographie

Ilhame MAHROUCH

Résumé

Ce projet de programmation en **langage** JAVA est constitué de trois parties. Dans la première il s'agit d'implémenter le procédé de chiffrement DES. La seconde concerne l'implantation du cryptosystème à clef publique RSA. Enfin, dans la troisième partie, il s'agit de programmer le protocole PGP.

1. Introduction

- Ce projet est à effectuer en **binôme**.

- Tous les programmes seront codés e**n JAVA** .

- Ce projet est constitué de trois parties (les sections 2, 3 et 4) qui ne sont pas indépendantes : la dernière partie utilise les algorithmes développés pour les deux premières parties.

- Lors de la soutenance il vous sera demandé d'expliquer les algorithmes en détail.

Courrier électronique : ilhame.mahrouch@outlook.fr

- Veiller à commenter très **soigneusement** votre code.

2. Programmation du cryptosystème DES

L'algorithme de **chiffrement**, l'algorithme de **déchiffrement** devront ainsi être implantés.

Le message clair M (qui est un bloc de 64 bits) et la clef principale, constituée de 64 bits, appelée KEY , seront des arguments d'entrée de votre algorithme de chiffrement DESencrypt, alors que le message chiffré (qui est un bloc de 64 bits) et la clef principale seront les arguments d'entrée de l'algorithme de déchiffrement DESdecrypt.

En bref, il faut coder les fonctions :

- KeySchedule(KEY, i) qui prend en entrée un bloc de 64 bits KEY, ainsi qu'un entier entre 1 et 16 (représentant la ronde courante), et renvoie la sous-clef du $i$ème tour K_i.
- DESencrypt(M, KEY) qui prend en entrée un bloc de 64 bits M et un bloc de 64 bits KEY, et renvoie un bloc de 64 bits C représentant le message M chiffré avec la clef secrète KEY par DES. Cet algorithme fait appel à l'algorithme KeySchedule.
- DESdecrypt(C, KEY) qui prend en entrée un bloc de 64 bits C et un bloc de 64 bits KEY, et renvoie un bloc de 64 bits M représentant le message C déchiffré avec la clef KEY par DES. Cet algorithme fait également appel à l'algorithme KeySchedule.

3. Programmation du cryptosystème RSA

Le procédé de chiffrement RSA est un algorithme de chiffrement à **clef publique**. Celle-ci est constituée de la paire (n, e) où n est un entier de la forme $n = pq$, où p et q sont deux entiers **premiers** (par exemple, $n = 15$ car

$15 = 3 \times 5$) et $e < (p-1)(q-1)$ est un entier **premier avec**[1] $(p-1)(q-1)$ (par exemple, si $n = 15$, alors $(p-1)(q-1) = 2 \times 4 = 8$ et on peut prendre $e = 3$ qui est premier avec 8). La **clef privée** est (p, q, d) où d est l'**inverse** de e modulo $(p-1)(q-1)$: cela signifie que $ed = k(p-1)(q-1)+1$ pour un certain entier k.

Un message clair ou chiffré est un entier $m \in \mathbb{Z}_n$, c'est-à-dire que $0 \leq m \leq n-1$. On **chiffre** m avec le clef publique $K_{\mathsf{pub}} = (n, e)$ en calculant l'entier $c \in \mathbb{Z}_n$ par

$$c = m^e \mod n$$

autrement dit on calcule m^e puis on calcule le reste de m^e dans la division entière par n (ces calculs sont possibles car on connaît m, e et n).
Le **déchiffrement** s'effectue de la façon suivante : soit $c = m^e \mod n$ un message chiffré. Alors on retrouve le message clair d'origine m à l'aide de la clef privée $K_{\mathsf{priv}} = (p, q, d)$ en calculant

$$c^d \mod n \ .$$

En effet on peut montrer – et vous l'admettrez – que $c^d \mod n = (m^e \mod n)^d \mod n = m$.

Il s'agit donc d'implanter un algorithme de chiffrement et un algorithme de déchiffrement. L'entrée de l'algorithme de chiffrement RSAencrypt est la clef publique (n, e) ainsi qu'un entier $m \in \mathbb{Z}_n$ et sa sortie est l'entier $c = m^e \mod n$. L'entrée de l'algorithme de déchiffrement RSAdecrypt est la clef privée (p, q, d) ainsi qu'un entier $c \in \mathbb{Z}_n$ et sa sortie est l'entier $c^d \mod n$. En particulier, si on prend comme entrée $c = m^e \mod n$, alors on retrouve en sortie l'entier m à l'aide de l'algorithme de déchiffrement.

En bref, il faut coder les fonctions :

- RSAencrypt(n, e, m) qui prend en entrée un entier n de la forme $n = pq$, p, q premiers, un entier e tel que $0 \leq e < (p-1)(q-1)$ qui est premier avec $(p-1)(q-1)$, et enfin un entier m tel que $0 \leq m \leq n-1$. Cet algorithme renvoie l'entier $c = m^e \mod n$.

[1] On dit que deux entiers a et b sont premiers entre eux si le seul diviseur commun de a et de b est 1.

- $\mathsf{RSAdecrypt}(p, q, d, c)$ qui prend en entrée deux entiers premiers p et q, un entier d tel que $0 \leq d < (p-1)(q-1)$ qui est premier avec $(p-1)(q-1)$, et enfin un entier c tel que $0 \leq c \leq n-1$. Cet algorithme renvoie l'entier $m = c^d \mod n$. Si on choisit p, q tels que $n = pq$, que d est l'inverse de e modulo $(p-1)(q-1)$ et enfin que $c = m^e \mod n$, alors $\mathsf{RSAdecrypt}(p, q, d, c)$ doit renvoyer m (d'après la propriété de déchiffrement).

Remarquez que le calcul des entiers e, d inverses l'un de l'autre modulo $(p-1)(q-1)$ ne fait pas partie du projet. Autrement dit, vous pouvez les calculer "à la main" (pas d'algorithme pour les générer).

4. Programmation de PGP

PGP (Pretty Good Privacy) utilise le cryptosystème DES conjointement avec le cryptosystème RSA. Supposons qu'Alice souhaite envoyer un message M de 64 bits à Bob en utilisant PGP, sans rencontrer Bob pour le choix de la clef secrète. La transmission est effectuée en plusieurs étapes[2] :

1. Alice choisit une clef KEY de 64 bits pour DES.
2. Alice découpe sa clef en 8 blocs KEY_i $(i = 1, \cdots, 8)$ de 8 bits consécutifs chacun, ainsi $\mathsf{KEY} = \mathsf{KEY}_1\ \mathsf{KEY}_2 \cdots \mathsf{KEY}_8$.
3. Alice transforme chaque bloc KEY_i de 8 bits, pour i variant de 1 à 8, en un entier k_i compris entre 0 et $2^8 - 1 = 255$ en considérant le block KEY_i comme l'écriture dans la base deux de l'entier k_i.
4. Alice chiffre l'entier k_i avec RSA en utilisant le clef publique $K_{\mathsf{pub}}^{Bob} = (n, e)$ de Bob, autrement dit, elle calcule $c_i = k_i^e \mod n$ pour chaque $i = 1, \cdots, 8$ (elle calcule $\mathsf{RSAencrypt}(n, e, k_i)$ pour $i = 1, \cdots, 8$). Elle obtient donc huit entiers c_i.
5. Alice chiffre le message clair M avec DES en utilisant la clef secrète KEY (elle calcule donc $\mathsf{DESencrypt}(M, \mathsf{KEY})$). Elle obtient le message chiffré C (de 64 bits).
6. Alice envoie $C, c_1, c_2, \cdots, c_8$ à Bob.
7. Bob déchiffre c_i à l'aide de sa clef privée (p, q, d) (de RSA), autrement dit il calcule $c_i^d \mod n$ et retrouve donc k_i (puisque $c_i = k_i^e \mod n$) pour chaque $i = 1, \cdots, 8$ (il calcule donc $\mathsf{RSAdecrypt}(p, q, d, c_i)$ pour $i = 1, \cdots, 8$).

[2]Il s'agit ici d'une adaptation du protocole PGP que vous ne trouverez pas dans la littérature.

8. Bob calcule l'écriture en base deux de l'entier k_i : il retrouve donc KEY_i pour chaque $i = 1, \cdots, 8$.
9. Bob reconstitue la clef de 64 bits $\mathsf{KEY} = \mathsf{KEY}_1\ \mathsf{KEY}_2 \cdots \mathsf{KEY}_8$.
10. Bob récupère enfin le message M en déchiffrant à l'aide de DES le message C avec la clef secrète qu'il a retrouvée KEY (il calcule donc $\mathsf{DESdecrypt}(C, \mathsf{KEY})$).

Afin de chiffrer avec RSA des entiers entre 0 et 255 (comme dans l'étape 4), l'entier $n = pq$ doit être supérieur ou égal à 256.

L'objectif de cette troisième et dernière partie est d'écrire deux programmes :

- Le premier programme $\mathsf{PGPencrypt}(M, \mathsf{KEY}, n, e)$ prend en argument une clef KEY de 64 bits, un bloc M de 64 bits, ainsi qu'une clef publique RSA (n, e). La sortie est un bloc C de 64 bits, et 8 entiers $c_1, \cdots, c_8$ calculés comme dans les étapes 1 à 5.

- Le second programme $\mathsf{PGPdecrypt}(C, c_1, c_2, \cdots, c_8, p, q, d)$ prend comme argument un bloc de 64 bits C, huit entiers $c_1, \cdots, c_8$ tous entre 0 et 255, et une clef privée RSA (p, q, d), sa sortie est un message de 64 bits M comme calculé lors des étapes 7 à 10. En particulier, si l'entrée du second programme est en fait la sortie du premier programme alors le message M obtenu en sortie du second programme est le même que celui donné en entrée du premier programme. **Veuillez bien noter qu'il vous faudra utiliser les algorithmes implantés lors des questions précédentes.**

Bonus (+3 points) : Créer une interface graphique en Java qui prend en param un fichier sélectionné depuis un emplacement et permet de crypter et décrypter avec PGP (La capture d'écran ci-dessous est donée à titre