Université de Limoges

Développement Logiciel Cryptographique

TP n° 2

1 Fonctions du cryptosystème RSA en mode standard

1.1 Génération de clé (mode standard)

Écrivez un programme qui prend en entrée deux entiers k et e, et qui génère une clé RSA (en mode standard) de k bits avec e comme exposant public.

Vous devrez veillez à ce que le module n soit le produit de deux premiers différents p et q de tailles sensiblement égales, et qu'il soit lui-même de taille exactement k bits.

Pour permettre le stockage de la clé dans un fichier par redirection de la sortie standard, votre programme devra afficher la clé en hexadécimal sur plusieurs lignes sous la forme suivante :

```
e = 0x......
n = 0x......
d = 0x......
```

Remarque 1 *Pour que l'inverse de e modulo $\phi(n)$ existe, il faut que e et $\phi(n)$ soient premiers entre eux.*

1.2 Chiffrement et déchiffrement (mode standard)

Écrivez une fonction de chiffrement `encrypt_rsa`(c, m, n, e) qui prend en entrée les deux éléments n et e d'une clé publique RSA ainsi qu'un message $m \in \mathbb{Z}_n$, et qui calcule la valeur du chiffré de m, soit $c = m^e \bmod n$.

Écrivez une fonction de déchiffrement `decrypt_rsa`(m, c, n, d) qui prend en entrée le module public n, l'exposant privé d ainsi qu'un chiffré $c \in \mathbb{Z}_n$, et qui calcule la valeur du déchiffré de c, soit $m = c^d \bmod n$.

Écrivez un programme qui prend en entrée le nom d'un fichier à chiffrer et le nom d'un fichier de clé RSA, et qui chiffre le contenu du fichier plutôt qu'un entier n'ayant pas de signification évidente. Pour réaliser ce chiffrement, le contenu du fichier devra être traité par blocs d'octets, et chacun de ces blocs sera encodé en un entier $m_i \in \mathbb{Z}_n$ en considérant sa séquence d'octets comme la représentation de m_i en base 256.

Écrivez le programme qui réalise l'opération inverse, c'est à dire qui déchiffre un fichier chiffré.

2 Fonctions du cryptosystème RSA en mode CRT

Répondre aux mêmes consignes que celles de toute la section 1 mais en générant et utilisant une clé RSA en mode CRT.

3 Applications

3.1 Échange d'information confidentielle

Cet exercice se fait en binômes.

Après s'être généré une clé RSA de 1024 bits, Alice en communique la partie publique à Bob. Bob choisit alors une information confidentielle de taille adéquate, la chiffre pour Alice grâce à la fonction `encrypt_rsa`(), puis la lui envoie. Alice reçoit l'information chiffrée et la déchiffre à l'aide de `decrypt_rsa`().

Permutez les rôles d'Alice et Bob.

3.2 L'attaque dite *des poubelles*

Cet exercice se fait en trinômes.

Alice, qui possède la clé publique (e_B, n_B) de Bob, lui envoie une information confidentielle m de manière chiffrée. Charly intercepte le chiffré c. Possédant lui aussi la clé publique de Bob, il génère un nombre aléatoire r, fabrique le chiffré modifié $c' = c \cdot r^{e_B} \bmod n_B$, et l'envoie à Bob. Bob ne reçoit que c' et tente de le déchiffrer. Le déchiffré m' de c' ne correspondant à rien

de sensé, il jette m' à la poubelle. Bob récupère m' et retrouve l'information confidentielle $m = m'/r \bmod n_B$.

Permutez les rôles d'Alice, Bob et Charly.

Remarque 2 *Vous pourrez simuler l'interception de Charly (et la non délivrance de c à Bob) par le fait que Alice envoie directement c à Charly plutôt qu'à Bob.*

Remarque 3 *Vous pourrez simuler la récupération par Charly de m′ dans la poubelle de Bob par le fait que Bob envoie m′ à Charly.*