Université Bretagne Sud
ENSIBS
Mathématiques pour la cryptographie

2023-2024

TP : Implémentation de RSA

Objectif :

Ce TP a pour objectif la mise en pratique des notions abordées dans le cours. A travers l'implémentation de l'algorithme RSA, nous aborderons les différentes opérations arithmétiques étudiées telles que le calcul du PGCD, le calcul modulaire, algorithme d'Euclide etc.

Les étudiants sont tenus de déposer le code source et un rapport à la fin de la séance du TP.

Les étudiants sont libres de choisir le langage de programmation.

Rappel :

RSA est un système de chiffrement à clé publique. Afin de générer les clés privée et publique, nous suivons les étapes suivantes :

1. Choisir deux grands nombres premiers p et q.
2. Calculer $n = pq$ et $\phi(n) = (p-1)(q-1)$
3. Choisir un entier e premier avec $\phi(n)$
4. Calculer d, l'inverse de e modulo $\phi(n)$.
5. La clé privée et la clé publique sont $K_d(d, n)$ et $K_e(e, n)$ respectivement.

Pour chiffrer le message m ($0 \leq m < n$) : $c = m^e[n]$ (c : message chiffré)

Pour déchiffrer : $m = c^d[n]$ (m: message en clair)

Notes :

RSA fonctionne avec des grands nombres, il convient alors d'utiliser des types de données adéquats.

Vous devez implémenter toutes les méthodes nécessaires pour la génération des clés et pour le chiffrement, test de primalité, le calcul modulaire, l'algorithme d'Euclide …

L'objectif est de maitriser toutes les étapes, par conséquent il n'est pas recommandé (n'est pas autorisé) d'utiliser le code source que vous pourrez trouver sur internet car ce n'est pas le but.

Énoncé :

1. Il est demandé de développer les méthodes nécessaires utilisées dans RSA.

- Test de primalité.
- Calcul du pgcd
- l'Algorithme d'Euclide étendu (calcul de l'inverse modulaire).
- Exponentiation rapide (exponentiation modulaire rapide).

Tester chaque méthode avec des petites valeurs.

2. Implémentation :

- Ecrire un programme permettant de générer la paire de clés $K_d(d, n)$ et $K_e(e, n)$, en fonction de la taille de n exprimée en nombre de bits.

- Ecrire les méthodes permettant de chiffrement/déchiffrement basé sur RSA.

3. Codage par bloc de caractères :

Soit l'alphabet suivant : **a** = 0, **b** = 1,…., **z** = 25, **0** = 26, **1** = 27, …, **9** = 35, **[espace]** = 36.

- Générer les clés privée/publique RSA, tel que n est suffisamment grand et représenté sur **32 bits (ou 16 bits).**
- Calculer la taille du bloc (nombre de caractères) à **chiffrer / déchiffrer** en fonction de n et la cardinalité de l'ensemble des caractères de l'alphabet.
- Utiliser le codage adéquat pour transformer la phrase (**lundi 18 septembre 2023**) en chiffre, puis chiffrer avec la clé publique générée ci-dessus.
- Déchiffrer le résultat obtenu (attention à la taille du bloc du message chiffré).

Note : Pour la question 3, on utilise :

Taille du bloc de caractères : $k <= log_L n$

L : nombre de caractère de l'alphabet

$k = log_L n \Leftrightarrow n = L^k$

Chiffrement / Déchiffrement : $C(x_0 x_1 x_2 \dots) = \sum_{0}^{k-1} (L^i x_i)$

4. Factorisation de n :

Les clés RSA sont habituellement de longueur comprises entre 1024 et 2048 bits. Jusqu'à maintenant, RSA reste sûr. On ne peut éventuellement pas casser une clé de cette taille, notamment avec la méthode *force brut*.

Prenons la valeur suivante :

n= 484001516976824291824081505569967425918027658822251613166283 7

- Evaluer la complexité de la factorisation de n.
- Ecrire un programme qui permet de trouver p et q (p et q sont des valeurs proches)
- Trouver d sachant que e = 65537