Ce TT a pour objectif La mise en place du chiffrement RSA et de quelques techniques d'attaques contre ce chiffrement. On fera attention à la complexité de nos fonctions afin de ne pas effectuer trop de calculs.

Vous prendrez soin de tester vos fonctions et de commenter votre code pour qu'il soit compréhensible. Les fonctions pouvant parfois s'appeler mutuellement, vous pouvez disposer des fonctions des questions précédentes pour répondre à une question même si la question précédente n'a pas été résolu.

Le type utilisé dans ce TT pour manipuler les entiers sera long long int car nous manipulerons de grands entiers. ils sont afficher par %ld au lieu de %d.

1 Introduction

Le chiffrement RSA est un algorithme de chiffrement a clé public asymétrique. Il repose sur la difficulté de factoriser un nombre en facteurs premiers. Le but de ce TT est de mettre en place ces algorithmes pour chiffrer et déchiffre.

Cet algorithme fonctionne comme suit : Étant donné deux nombres premiers p et q, on considère $n = pq$ et $\varphi(n) = (p-1)(q-1)$ son indicatrice d'Euler.

On considère $1 < e < \varphi(n)$ un nombre premier avec $\varphi(n)$. Et d l'inverse de e modulo $\varphi(n)$, c'est-à-dire $ed \equiv 1[\varphi(n)]$.

La clé publique est le couple (n, e) et la clé secrète est le couple (n, d).

Le chiffré d'un message m est alors $m^e[n]$, et si c est un message chiffré, on peut le déchiffrer en calculant $c^d[n]$.

2 Mise en place du chiffrement

Pour ce TT, on utilisera $p = 597689$, $q = 4523$, $e = 17$.

Question 1
Définir les entiers p, q, n, e, d et $phi = \varphi(n)$ en variable globale. Calculer n et φ dans le main.

Question 2
Écrire une fonction exp_rapide de type long long int qui prend en entrée trois entiers a et b et N et qui renvoie $a^N[b]$.

Question 3
Écrire une fonction inverse_modulaire de type long long int qui prend en entrée deux entiers a et b et qui renvoie l'inverse modulaire de a dans $\mathbb{Z}/b\mathbb{Z}$ si a et b sont premiers entre eux (i.e. le plus petit entier k tel que $a * k \equiv 1[b]$.). On renverra 0 si a et b ne sont pas premiers entre eux. On rappelle que l'algorithme d'Euclide étendue permet de faire ces calculs.

3 Attaques contre RSA

Le calcul d'inverse modulaire étant une opération facile, attaquer RSA consiste à factoriser n. Nous allons donc nous intéresser aux méthodes de factorisation

3.1 attaque exhaustive

Question 4
Écrire une fonction recherche_exhaustive de type long long int qui prend en entrée un entier n et qui renvoie un diviseur premier de n en essayant tous les nombres compris entre 2 et $\sqrt{n}$.

3.2 Méthode de Fermat

La méthode de Fermat consiste à chercher des "gros" facteurs. Si l'on souhaite factoriser $N = pq$, on cherche à écrire $N = a^2 - b^2$ dans le but d'avoir $N = (a-b) * (a+b)$. Pour ce faire on calcule différentes valeurs de a en partant de $a = \left\lceil \sqrt{N} \right\rceil$ et en incrémentant petit à petit la valeur de $a^2 - N$ et on observe si cette valeur est un carré ou pas.

Question 5 Écrire une fonction est_carre de type int qui prend en entrée un entier N et renvoie 1 si N est un carré et 0 sinon. On pourra importer <math.h> et utiliser la fonction sqrt.

Question 6
Écrire une fonction fermat de type long long int qui prend en entrée un entier n et renvoie un diviseur premier de n en appliquant la méthode de Fermat.

3.3 La méthode $p-1$

La méthode $p-1$ de Pollard fonctionne comme suit : si n est un entier non premier, et si p est un diviseur premier de n, par le petit théorème de Fermat, on a $a^{k(p-1)} \equiv 1[p]$ pour tout k et tout a premier avec p. Donc $a^{k(p-1)} - 1 \equiv 0[p]$. Cela signifie donc que p divise $a^{k(p-1)}$. Dès lors, comme p est aussi un facteur de n, on aura donc que p divise $pgcd(a^{k(p-1)}, n)$. Le but de la méthode est de calculer pour un certain entier a premier avec n une puissance qui sera un multiple de $(p-1)$. Si l'on tombe sur un diviseur de n on a gagné, sinon la méthode a échoué.

L'algorithme consiste donc à choisir un entier a premier avec n (en général $a = 2$ et de calculer a^M pour une bonne puissance M, généralement prendre M comme un produit de nombre premier convient.

Question 7

On prend $M = \prod_{p \in \mathbb{P}, p < B} p^{\lfloor \log_p(B) \rfloor}$. où $\mathbb{P}$ désigne l'ensemble des nombres premiers et B une borne choisie. On pourra écrire une fonction auxiliaire qui calcule M.

Écrire une fonction Pollard de type long long int qui prend en entrée un entier n et une borne B et qui essaie de calculer un diviseur de n, et le renvoie s'il réussit. On renverra 0 si l'algorithme échoue.

On pourra tester cette fonction avec $B = 20$.