MA-203: TP Noté - Résiduosité quadratique
Mise en œuvre du schéma de Blum-Goldwasser

18 mars 2021

Modalités. Il vous est demandé d'envoyer à l'adresse `thomas.debris@inria.fr` vos programmes avant le 20/03/2021 à minuit. En cas de retard, une pénalité de -2 points par jour vous sera infligée.

Le but de ce TP *noté* est de programmer le schéma de chiffrement de Blum et Goldwasser. Le langage de programmation est C. Étant donné que vous aurez à manipuler des grands entiers, **vous utiliserez la bibliothèque multi-précision GMP** déjà utilisée dans le TP6 :
http ://gmplib.org/
http ://gmplib.org/manual/
http ://gmplib.org/manual/Integer-Functions.html

L'algorithme de chiffrement de Blum et Goldwasser est défini comme suit :

— *Phase de génération de clefs* :

Alice génère p et q, deux grands nombres premiers, vérifiant $p \equiv q \equiv 3 \bmod 4$.

calcule et publie $n = pq$; garde (p, q) secret.

— *Chiffrement* : génération de la suite chiffrante (algorithme de Blum Blum Shub)

Bob choisit $s \in_R [1, n-1]$ (le germe aléatoire)

calcule $x_0 = s^2 \bmod n$

pour $i \in \mathbb{N}^*$, calcule $x_i = x_{i-1}^2 \bmod n$, et $z_i = x_i \bmod 2$ (la séquence $z_1, z_2, z_3, \ldots$ est la suite chiffrante).

— *Chiffrement* : construction et envoi du chiffré

Bob représente le message à chiffrer comme une chaîne binaire de longueur t : $m = m_1 \ldots m_t$

calcule $c_i = z_i \oplus m_i,\ 1 \leq i \leq t$

calcule $x_{t+1} = x_t^2 \bmod n$

envoie le chiffré $(c_1, \ldots, c_t, x_{t+1})$ à Alice.

— *Déchiffrement* : à réception du chiffré, Alice calcule :

$d_1 = ((p+1)/4)^{t+1} \bmod (p-1)$ et $d_2 = ((q+1)/4)^{t+1} \bmod (q-1)$

$u = x_{t+1}^{d_1} \bmod p$ et $v = x_{t+1}^{d_2} \bmod q$

$a = p^{-1} \bmod q$ et $b = q^{-1} \bmod p$

$x_0 = vap + ubq \bmod n$

$x_i = x_{i-1}^2 \bmod n,\ z_i = x_i \bmod 2,\ 1 \leq i \leq t.$

et retrouve le clair m en calculant, pour $1 \leq i \leq t$: $m_i = c_i \oplus z_i$.

Pour générer un nombre premier congru à 3 modulo 4 d'une taille donnée en argument, vous utiliserez la fonction "GenPremier" implantée dans le TP6 et qui vous est redonné ici.

Etapes de programmation :

1. Écrire un programme qui génère une clé privée et la clé publique qui lui est associée (voir `genkey.c`).
2. Compléter la fonction `BBS_step` qui prend en entrée l'état courant du générateur de Blum Blum Shub (i.e. x_i) et le module n, met à jour l'état courant (i.e. calcule x_{i+1}) et retourne z_{i+1} (voir `Blum.c`).
3. Écrire un programme de chiffrement qui prend en argument un ficher texte clair et une clé publique et qui calcule le chiffré correspondant avec l'algorithme de Blum-Goldwasser (voir `encrypt.c`).
4. Écrire un programme de déchiffrement qui prend en entrée un ficher texte chiffré et une clé privée et qui calcule le clair correspondant avec l'algorithme de Blum-Goldwasser (voir `decrypt.c`).

Le fichier Blum.c contient les fonctions Fermat et Genpremier ainsi que le prototype de la fonction BBS_step. Il contient également les fonctions d'entrées-sorties vous permettant de :

- lire une clef publique se trouvant dans un fichier
- écrire une clef publique dans un fichier
- idem pour une clef privée
- lire un message clair (fichier texte) ou écrire un message clair dans un fichier
- idem pour un texte chiffré.

Il est demandé de faire trois fichiers principaux : un pour la génération de clefs (modèle : genkey.c), un pour le chiffrement (modèle : encrypt.c), et un pour le déchiffrement (modèle : decrypt.c).