MA 203: TD7 - Résiduosité quadratique
Mise en œuvre du schéma de Blum-Goldwasser

7 avril 2017

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

Rappelons l'algorithme de Blum et Goldwasser :

— 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, nous utiliserons la fonction "GenPremier" implantée dans le TP6.
Etapes de programmation :

1. Écrire une 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}.
2. Écrire une fonction de chiffrement qui prend en entrée un ficher texte clair et calcule le chiffré de ce fichier par l'algorithme de Blum-Goldwasser (retourné en deuxième argument de la fonction, voir le modèle encrypt.c).
3. Écrire une fonction de déchiffrement qui prend en entrée un ficher texte chiffré et calcule le clair correspondant par l'algorithme de Blum-Goldwasser (retourné en deuxième argument de la fonction, voir le modèle 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).

Application : déchiffrer le message contenu dans le fichier chiffre.txt, avec la clef publique contenue dans le fichier key.pub et la clef secrète contenue dans le fichier key.