"AES" & "El Gamal"

GS15 - A15 - Projet Informatique

Sujet présenté en cours le 20/10

Rapport à rendre avant le 04/01 - Soutenance entre le 04/01 et le 08/01

---

1 Description du projet à réaliser

Le but de ce projet informatique est de vous faire créer un outil offrant la possibilité d'utiliser deux algorithmes différents de chiffrement : l'un à symétrique, l'autre à clé publique et un algorithme de signature. Le choix de l'algorithme utilisé est laissé à l'utilisateur, qui pourra par exemple l'indiquer en entrant un nombre spécifique au clavier avec une interface du type :

```
Selectionner votre fonction de chiffrement
->1<- Chiffrement AES
->2<- Chiffrement El Gamal
->3<- Signature El Gamal
```

L'utilisateur entre son choix (1, 2, ou 3) ...

Les trois algorithmes qui vous sont demandés sont décrits ci-dessous, respectivement dans les sections 2, 3 et 4.
Il est conseillé de ré-utilisé les fonctions données en TP pour la lecture et l'écriture des fichiers ainsi que les fonctions que vous avez pu écrire durant les séances de TP.
Enfin, le choix du langage de programmation vous appartient, néanmoins votre enseignant n'étant pas omniscient, un soutient n'est assuré que pour les langages C/GMP (et Matlab). La seule contrainte **obligatoire** est seulement de respecter les consignes données dans le Section 5 du présent document.

---

2 AES

Le chiffrement symétrique "AES" que vous devez coder doit être en tout point identique à AES, utilisant des états (State) et des éléments (Octets). Trois implémentation différentes sont demandées utilisant respectivement des clés de 128, 192 et 256 bits, pour chacun de ces cas, les états seront mis sous la forme de matrices de 4 lignes et de N_b colonnes avec respectivement $N_b \in \{4, 6, 8\}$ suivant la taille de la clé. On rappellera (voir les détails du cours) que le nombre de tournées dépend également de la taille de la clé, respectivement, 10, 12 ou 14 tournées suivant la taille de la clé.

Rappels des fonctions élémentaires de l'AES :

1. La fonction SubBytes introduit de la non-linéarité en opérant une inversion de chaque élément de la matrice dans un corps $G\mathbb{F}_{2^8}$. Vous pourrez implémenter cette fonction en refaisant les calculs d'inversion dans ce corps particulier ou en tabulant les opération une fois pour toutes (si votre implémentation utilise le calcul vous pourrez aussi vous amuser à changer la matrice $\mathbf{A}$ et le vecteur $\mathbf{c}$ de l'application affine.
   Idem pour la fonction InvSubHéxa, l'implémentation est laissé libre.
   NB : le choix de la matrice identité pour $\mathbf{A}$ ou du vecteur nul pour $\mathbf{c}$ sont naturellement proscrit !

2. La fonction ShiftRows opérant un décalage des lignes des états (là encore attention suivant la taille de la clé). La fonction InvShiftRows est assez évidente aussi.

3. La fonction MixColumns qui effectue une substitution des colonnes par multiplication matricielle. Là encore, vous pourrez si vous le souhaitez tabuler les opérations (de multiplication) ou bien effectuer les opérations sur le corps particulier utilisé dans AES. Vous pourrez aussi redéfinir la matrice de mélange $\mathbf{X}$ et son inverse $\mathbf{X}'$ (pour la fonction InvMixColumns).

4. La fonction AddRoundKey combinant par $\oplus$ ("xor") l'état State avec une sous-clés, cette fonction demeurera identique, les sous-clés utilisées étant de 320 bits dans votre algorithme.

5. La fonction KeyExpansion est laissé à votre choix, vous pourrez utiliser la fonction SubBytes par exemple ou reprendre la fonction d'expansion de la clé d'AES.

---

3 Chiffrement El Gamal

Le but de cet algorithme "El Gamal" est de vous faire implémenter cet algorithme de chiffrement à clé publique.
Pour cet algorithme, la seule contrainte est d'utiliser des entier grands (d'au moins quelques centaines de bits) et donc l'utilisation de GMP est nécessaire.
Vous pouvez, si cela vous simplifie les choses, effectuer les opérations de chiffrement et de déchiffrement successivement ou bien écrire le chiffré dans un fichier qui sera déchiffré ultérieurement.
Dans tous les cas, il est demandé d'implémenter :

1. Le test de primalité de Rabin-Miller et de générer pseudo-aléatoirement des entier d'une taille exactement $n+1$ bits (pour chiffrer des blocs de n bits). L'entier p premier ainsi trouvé définira le corps $\mathbb{Z}_p^\star$ sur lequel on travaillera.

2. Trouver ensuite un élément générateur (pas facile).
   Une petite aide : il est possible d'utiliser le théorème de Lagrange sur les groupes qui prouve que l'ordre d'un élément de $\mathbb{Z}_p^\star$ est toujours un diviseur de $p-1$. Si on choisit "*avec intelligence*" un entier premier p seule un nombre très réduit de puissances doivent être testées.

Une fois un élément générateur g trouvé, il ne vous reste plus qu'à choisir pseudo-aléatoirement un élément $x \in \mathbb{Z}_p^\star$ et à calculer $h = g^x$. La clé publique à diffuser est le triplet (P, g, h) et la clé privée à

garder secrète est x Le chiffrement se fait identiquement à ce que nous avons vu en cours !

En option : vous pourrez aussi vous amuser à "casser" El Gamal, en résolvant le problème du logarithme discret, c'est à dire retrouver x en connaissance $g^x \equiv h \bmod p$ à l'aide de l'algorithme BSGS. Vous pourrez faire ce test pour un entier premier p de 8, 12, 16, 20, ... bits et mesurer l'évolution du temps de calcul nécessaire.

---

4 Signature El Gamal

Dans la version de la signature que l'on propose d'implémenter dans ce projet on suppose que Alice choisit au hasard un entier y (avec $y \in \mathbf{Z}_p^\star$) et calcule $r = g^y$ ainsi que $s = (H(M) - x \times r)y^{-1}$ (ici $H(M)$ représente une fonction du message à signer, par exemple les n premiers bits du message a signer, vous pourrez inventer une autre fonction de hashage "plus évoluée").
La façon de calculer l'inverse y^{-1} vous appartient.
La signature a envoyer est le couple (r, s).
On acceptera la signature comme authentique si on pourra vérifier, à la réception du message M et de la clé (r, s) que : $g^{H(M)} = y^r \times r^s$

---

5 Documents à fournir et autres détails

Il est impératif que ce projet soit réalisé en binôme. Tout trinôme obtiendra une note divisé en conséquence (par $3/2$, soit une note maximale de $13,5$).
Encore une fois votre enseignant n'étant pas omniscient et ne connaissant pas tous les langages informatique du monde, l'aide pour la programmation ne sera assuré que pour le C/GMP (et le matlab). Par ailleurs votre code devra être commenté (succinctement, de façon à comprendre les étapes de calculs, pas plus).
Ce code doit prendre en entré un texte (vous pouvez aussi vous amuser à assurer la prise en charge d'image pgm comme en TP, de fichiers binaires, etc .... mais la prise en charge des textes est le minimum souhaité).
Un court rapport est également attendu; ce dernier devra argumenter les choix que vous fait notamment en ce qui concerne les implémentations que vous avez fait (par exemple, comment générer une clé pour chaque tournée, comment faire les calculs, etc. ...).
Vous avez le droit de chercher des solutions sur le net (ou bien où vous voulez), par contre, essayez autant que possible 1) de comprendre les éléments techniques trouvés (voire les présenter dans votre rapport s'ils sont intéressants, par exemple comment trouver un entier premier sécurisé pour El Gamal et comment vérifier qu'un entier a est générateur ?) et 2) d'utiliser au minimum les fonctions incluses dans GMP, car le but est de ré-implémenter les choses vues en cours afin que vous les maîtrisiez au mieux.
Un autre exemple de choses que vous pouvez faire est de proposer un protocole de chiffrement symétrique avec votre AES mais en utilisant un protocole d'échange de clé basé sur le chiffrement asymétrique de votre El Gamal ....
Je réponds volontiers aux questions (surtout en cours / TD) mais ne ferais pas le projet à votre place ... bon courage !