“Super AES” & “Super El Gamal”

GS15 - A14 - Projet Informatique

Sujet présenté en cours le mardi 21 octobre 2012
Projet à rendre et à présenter au plus tard le vendredi 19 décembre

---

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. Le choix de l'agoriothme utilisé est laissé à l'utilisateur, qui pourra par exemple l'indiqué en entrant un nombre spécifique au clavier avec les commandes :

```
>> fprintf('->1<- \t mon Super AES \n ->2<- \t mon Super El Gamal \n');
>> choix_user = input('Choix de l algorithme :\n ')
```

retournent les résultats suivants :

```
->1<-   mon Super AES
 ->2<-   mon Super El Gamal
Choix de l algorithme :
 1
choix_user =
     1
```

Comme dans l'exemple au dessus les deux algorithmes de Chiffrements qui vous sont demandés sont :

1. le chiffrement “super AES”,
2. le chiffrement “super El Gamal”,

La description de chacun de ces algorithmes de chiffrement est données ci-dessous, respecivement dans les Sections 2 et 3.
Il est conseillé de ré-utilisé les fonctions données en TP pour la lecture et l'écriture des fichiers avec *Matlab* ou un équivalent gratuit (*Octave* et *Scilab* pas ex.).
Enfin, le choix du language de programmation vous appartient, néanmoins votre enseignant n'étant pas omniscient, un soutient n'est assuré que pour le language Matlab. La seule contrainte **obligatoire** est seulement de respecter les consignes données dans le partie 4 du présent document.

2 Super AES

Le chiffrement symétrique "super AES" utilisera des états (State) et des éléments (Octets de l'AES) de taille supérieure. Trois implémentation différentes sont demandées utilisant respectivement des clés de 320, 480 et 640 bits.
L'algorithme que vous devrez construire est très similaire à AES. Les principales différences sont que les états représentant les flux de données, entrées/sorties des fonctions, seront mis sous la forme de matrices de 5 lignes et de N_b colonnes. En plus, chaque éléments de cette matrice sera désormais un héxa, c'est à dire un mot binaire de 16 bits. Les mots de la matrice seront donc désormais des blocs de 5 héxa soit 80 bits.
La figure ci-dessous illustre un état State pour des blocs de message qui ont désormais une longueur de 320 bits.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$

Représentation matricielle des blocs du message à chiffrer.
Chaque élément $a_{i,j}$ représente un héxa de 16 bits.
Chaque colonne représente un Mot de 5 héxa ou 80 bits.

Rappels des fonctions élémentaires de l'AES et détails des changements nécessaires/demandés par rapport à l'algorithme AES original :

1. La fonction SubBytes introduit de la non-linéarité en opérant une inversion de chaque élément de la matrice dans un corps $G\mathbb{F}_{2^8}$. Puisque dans le projet informatique le éléments ne sont plus des Octets mais des héxa, cette fonctions devra être modifiée.
   En particulier il est demandé de définir le Corps $G\mathbb{F}_{2^{16}}$ avec lequel vous souhaitez travailler, de trouver comment inverser les éléments de ce corps et éventuellement de tabuler cette inversion.
   Enfin, l'application affine $\mathbf{A} \times a_{i,j}^{-1} + \mathbf{c}$ étant utilisée dans la fonction SubBytes la matrice $\mathbf{A}$ et le vecteur $\mathbf{c}$ devront être adapté comme vous le souhaitez. Il faudra également définir $\mathbf{A}'$ qui est l'inverse de $\mathbf{A}$ dans le corps $\mathbb{Z}/2\mathbb{Z}$ et le vecteur $\mathbf{c}' = \mathbf{A} \times c$ pour pouvoir définir la fonction InvSubHéxa.
   NB : le choix de la matrice identité pour $\mathbf{A}$ ou du vecteur nul pour $\mathbf{c}$ sont naturellement proscrit !

2. La fonction ShiftRows pourra être modifié selon vos envies, mais cela ne semble pas nécessaire et pas très intéressant.

3. La fonction MixColumns devra également subir de profonde modification, car on travaillera désormais avec des mots de 5 héxa. Il faudra donc prendre soin à définir une matrice $\mathbf{X}$ d'une dimension adaptée, mais surtout définir son inverse $\mathbf{X}'$ dans $G\mathbb{F}_{2^{16}}$ pour inverser la fonction MixColumns (et ainsi définir la fonction InvMixColumns). Là encore, c'est à vous de définir la matrice $\mathbf{X}$ comme vous le souhaitez.
   NB : Il est attendu une matrice $\mathbf{X}$ "pas trop facile quand même", disons 3 éléments sur les 5 au moins doivent être distincts (si vous choisissez une matrice circulante).
   Enfin, là encore, selon la matrice $\mathbf{X}$ choisie et son inverse, il peut être judicieux de tabuler les calculs dans $G\mathbb{F}_{2^{16}}$.

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.

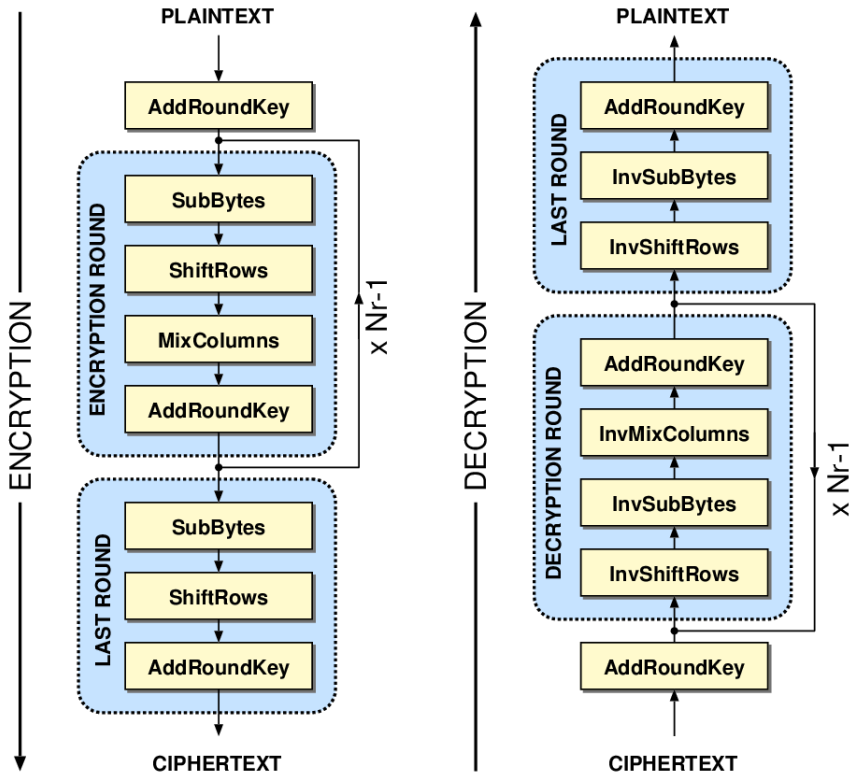
5. La fonction KeyExpansion est laissé à votre choix, vous pourrez utiliser la fonction SubBytes par exemple.

Par ailleurs, le nombre de tournées utilisées sera le même que celui utilisé dans l'AES.
Bien que aucune contrainte sur l'architecture de votre code ne soient imposée, il est fortement recommandé (dans votre intérêt) de coder chaque fonction élémentaire de "Super AES" comme une fonction ou une sous-fonctions de *Matlab* / *Octave* / *Scilab* (voir schéma page suivante).

Quelques aides pratiques vous sont données ci-dessous :

- Les ordinateurs de la plupart des salles informatiques sont équipés de *Matlab*, vous pouvez donc les utiliser librement. Une version d'évaluation de 30 jours de *Matlab* est également disponible à l'adresse suivante http://www.mathworks.fr/programs/trials/trial_request.html.
  Puisque vous devez vous mettre en binôme, cela vous fera chacun 30 jours soit 60 jours en tout (sans compter les salles informatiques) ce qui vous laissera largement le temps de faire votre projet.

- Il vous est interdit d'utiliser les fonctions de *Matlab* / *Octave* / *Scilab* adaptées aux calculs dans les corps de Galois. Vous devrez donc écrire un script pour tester si un polynôme est généraeteur et si un polynôme est irréductible.



Principe de chiffrement/déchiffrement de l'AES.
Il est fortement recommandé de coder les cinq différentes fonctions
qui interviennent comme des fonctions/sous-fonctions Matlab / Octave / Scilab.

2.1 Example d'implémentation possible

Tout d'abord, vous pouvez tout à fait utiliser les codes fournies et développés en TD pour la cryptographie symétrique et DES afin de faire les conversion de texte en entiers, binaires, etc ... Il vous faudra seulement prendre soin à utiliser 2 caractères (de 8 bits) pour réprésenter les héxa en entrée et en sortie de votre algorithme.
Voici un exemple des paramètres qu'il est possible d'utiliser dans votre algorithme.
Le corps $G\mathbb{F}_{2^{16}}$ choisi est défini par $\mathbb{Z}/2\mathbb{Z}[X]/ < M >$ avec $M = X^{16} + X^5 + X^3 + X^2 + 1$. On pourra vérifier son irréductibilité à l'aide d'un script ad-hoc.

La matrice $\mathbf{A}$ qui vous est donnée en exemple est la suivante :

$$\begin{pmatrix}
1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Cette matrice peut être générée aisément à l'aide de la commande suivante (le premier vecteur correspond à la première colonne, le second à la première ligne) :

```
A = toeplitz([1 0 0 0 1 1 1 0 0 1 1 1 1 0 0 1] , [ 1 1 0 0 1 1 1 0 0 0
1 1 1 0 0 0])
```

La matrice $\mathbf{A}'$ correspondante est (à vous de trouver comment l'obtenir sous *Matlab*) :

$$\begin{pmatrix}
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1
\end{pmatrix}$$

Enfin les vecteurs $\mathbf{c}$ et $\mathbf{c}'$ proposés les suivants :

$$\mathbf{c} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

et

$$\mathbf{c}' = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

On verifiera que :

```
Aprim = A^(-1) (ou bien mod(Aprim*A,2)=0)
cprim = Aprim*c (ou bien mod(Aprim*c+cprim,2)=eye(5) matrice identité)
```

Enfin les matrices de mixages et démixages des colonnes choisies sont les suivantes. À vous de trouver comment calculer $\mathbf{X}'$ à partir de $\mathbf{X}$ avec *Matlab* / *Octave* / *Scilab* (Aide: utiliser dans $G\mathbb{F}_{2^p}$ la méthode d'inversion d'une matrice de Gauss-Jordan ou la "formule" de Laplace $X^{-1} = \frac{1}{\det(X)}\mathrm{com}(X)^{\mathrm{T}}$ avec $\det$ le déterminant et $\mathrm{com}(X)$ la comatrice ou matrice des cofacteurs de X) :

$$\mathbf{X} = \begin{pmatrix} \{2\} & \{4\} & \{1\} & \{2\} & \{2\} \\ \{2\} & \{2\} & \{4\} & \{1\} & \{2\} \\ \{2\} & \{2\} & \{2\} & \{4\} & \{1\} \\ \{1\} & \{2\} & \{2\} & \{2\} & \{4\} \\ \{4\} & \{1\} & \{2\} & \{2\} & \{2\} \end{pmatrix} \text{ et } \mathbf{X}' = \begin{pmatrix} \{51EB\} & \{3ABE\} & \{8F02\} & \{D5DC\} & \{8741\} \\ \{8741\} & \{51EB\} & \{3ABE\} & \{8F02\} & \{D5DC\} \\ \{D5DC\} & \{8741\} & \{51EB\} & \{3ABE\} & \{8F02\} \\ \{8F02\} & \{D5DC\} & \{8741\} & \{51EB\} & \{3ABE\} \\ \{3ABE\} & \{8F02\} & \{D5DC\} & \{8741\} & \{51EB\} \end{pmatrix}$$

Cet exemple de paramètres choisis, ainsi que celui de l'AES, pourront vous être très utiles pour vérifier que vous avez bien compris comment faire les calculs nécessaires avec *Matlab* / *Octave* / *Scilab*.
Évidemment, l'implémentation de l'algorithme ne doit pas se faire avec ce jeu de paramètres donnés à titre d'exemple ...
Vous pourrez tabuler l'ensemble de vos calculs pour accélerer le chiffrement / déchiffrement pour peu que vous code offre la possibilité de recalculer ces tables (si je change d'élements générateur ou de polynôme irréductible par exemple)

---

3 "Super El Gamal"

Le but de cet algorithme "Super El Gamal" est de vous faire ré-utiliser les codes utilisés dans le "Super AES" pour faire du chiffrement et de la signature de El-Gamal dans le corps de Galois $G\mathbb{F}_{2^{16}}$ (vous pourrez comparer si vous le souhaitez avec une implémentation de El Gamal dans $\mathbb{Z}_p$.
Dans les deux algorithmes de El-Gamal, chiffrement et signature, on commence par l'étape de génération des clés suivante :

1. Alice (=vous) choisit un entier premier P, un élément générateur g.
2. On choisit ensuite la clé secrète x (avec $0 < x < P$) aléatoirement et on calcule $h = g^x \mod P$.
3. La clé publique à diffuser est le triplet (P, g, h) et la clé privée à garder secrète est x.

Dans votre algorithme "Super El Gamal" la différence majeure réside dans la première étape :

1. Alice (=vous) choisit un polynôme irréductible P de degré $\deg(P) = p$, un élément générateur g de $G\mathbb{F}_{2^p}$.

2. On choisit ensuite la clé secrète x (avec $0 < x < 2^p$) aléatoirement et on calcule $h = g^x \mod P$.

3. La clé publique à diffuser et la clé secrète seront identiques.

Pour le chiffrement d'un bloc M représenté sous la forme d'un entier $0 \leq M < 2^p$ le chiffré C de M est donné par $C = (g^y, M \times h^y)$ avec y (avec $0 < y < P$) un entier aléatoirement choisi.
Pour le déchiffrement (que vous devez aussi implémenter) on calcule M à partir de $C = (g^y, M \times h^y)$ de la façon suivante. On commence par calculer $(g^y)^x$ puis son inverse : $c' = ((g^y)^x)^{-1}$ pour enfin retrouver le message clair ainsi : $M = (M \times h^y) \times c'$.

Dans la version simplifié de signature que l'on propose d'implémenter dans ce projet on suppose que Alice choisit au hasard un entier y (avec $0 < y < 2^p$) 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 2^p premiers bits du message a signer, vous pourrez inventer une autre fonction de hashage "plus évoluée").
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 : $g^{H(M)} = y^r \times r^s$

---

4 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, seulement si vous utilisez *Matlab* / *Octave* / *Scilab* une aide vous sera garantie, dans tous les cas il est impératif de fournir l'ensemble des codes sources développés nécessaires au fonctionnement de votre projet.
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, de son, 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. ...).
Je réponds volontier aux questions (de préférence en cours / TD) mais ne ferais pas le projet à votre place ... bon courage !