Projet de cryptologie
Encadré par M. Michaut et M. David

Cryptage-décryptage de l'AES 128 bits en langage C

Par Quentin Comte-Gaz et Adrien Lillo

ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

Sommaire

1 Présentation sommaire de l'AES 128 bits

1.1 Origine

L'AES ou Advanced Encryption Standard est devenu le nouveau standard de cryptage en 2000 après avoir remporter le concours du même nom. En effet, le DES qui était jusque là le standard devenait de moins en moins sur.

1.2 Fonctionnement

Ce système de cryptage est de type symétrique. Il faut noter que ce type de cryptage est rapide et utilise des clés relativements courtes. Mais en contre partie L'échange de clé est difficile. On utilise donc un système complémentaire (le RSA) lors de l'échange de clé.

Le fonctionnement de l'AES 128 est basé sur le découpage du message à crypter en bloc de 128 bits (16 octets). En fonction du niveau de cryptage voulu, on choisi une clé de 128, 192 ou 256 bits. Les 16 octets en entrée sont permutés selon une table définie au préalable. Ces octets sont ensuite placés dans une matrice de 4x4 éléments et ses lignes subissent une rotation vers la droite. L'incrément pour la rotation varie selon le numéro de la ligne. Une transformation linéaire est ensuite appliquée sur la matrice.

Finalement, un XOR entre la matrice et une autre matrice permet d'obtenir une matrice intermédiaire. Ces différentes opérations sont répétées plusieurs fois et définissent un "tour".

Le nombre de tours change en fonction de la taille de la clé et donc du niveau de sécurité attendu. Le table ci-après regroupe l'ensemble des caractéristiques pour l'ensemble des AES.
Nb : Taille des blocs (en mots de 32 bits)
Nk : Taille de la clé (en mots de 32 bits)
Nr : Nombre de tours

	N_k	N_b	N_r
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

FIGURE 1.1 – Configuration de l'AES

1.3 Attaques

Cet algorithme a été conçu pour résister à tous les types d'attaques connus. Il rend le cassage linéaire et différentiel impossible. A ce jour la force brute est l'une des seules solutions possibles. Néanmoins, si un jour l'AES venait à être cassable "facilement", il suffirait d'augmenter la taille de la clé (cf : Tableau précedent). Notons qu'il n'exite pas de clés faibles pour ce système (pour le moment).

2 Le cryptage-décryptage

2.1 Création des clefs de ronde KeyExpansion

Nous avons une fonction générant une clé aléatoire sur 128 bits. Cette clé sera le point de départ des clés de ronde. La fonction KeyExpansion va former Nb*(Nr+1) mots. Elle va commencer par faire une permutation circulaire puis on va appliquer la S-box pour finir par créer les clés de ronde.

2.2 Les differentes étapes lors d'un tour

2.2.1 Ajout des clefs de ronde au message AddRoundKey

Cette fonction permet de faire un XOR entre le message d'entrée et la clé de ronde.

2.2.2 Permutation du message (Inv)SubBytes

Permute chaque élément du message par un élément de la S-Box. On notera que cette permutation est non-linéaire.

2.2.3 Décalage : (Inv)ShiftRow

Décalage cyclique des lignes des blocs messages de 0 puis 1 puis 2 puis 3 vers la gauche pour les 1er puis 2ème puis 3ème puis 4ème ligne.

2.2.4 Transformation linéaire : (Inv)MixColumns

Cette fonction mélange les colonnes du bloc message. Les colonnes sont considérées comme des polynômes de GF(2^8) et multipliées modulo $x^4 + 1$ avec un polynome $a(x)$ fixé tel que : $a(x) = 3 * x^3 + x^2 + x + 2$

2.3 La fonction de cryptage-décryptage d'un message de 16 octets

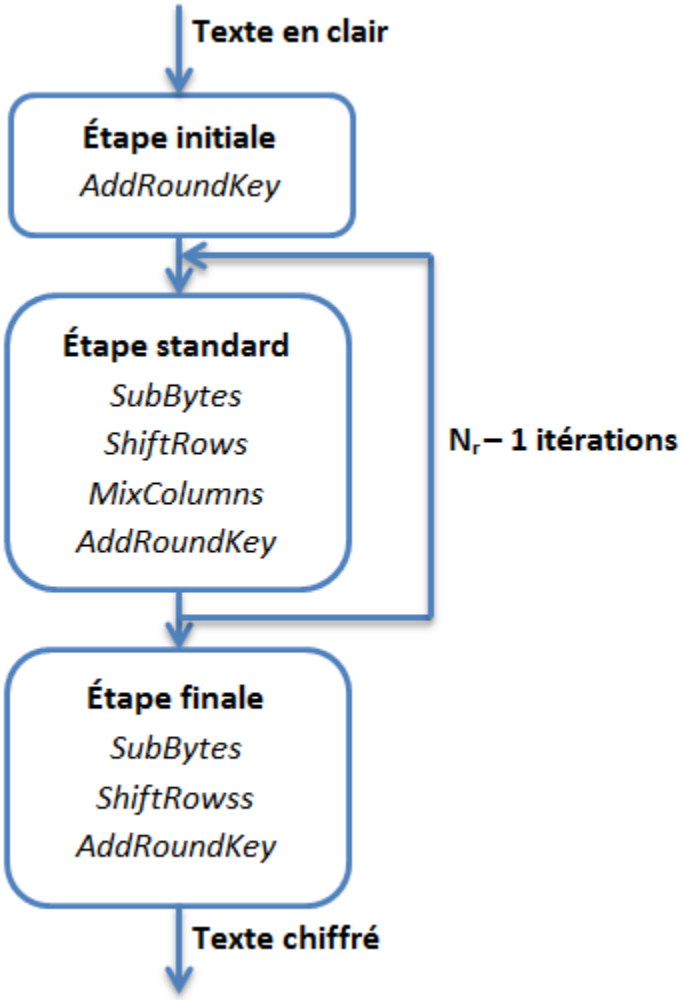

FIGURE 2.1 – Foncionnement général de l'AES

Maintenant que chaque fonction a été décrite, il est plus commode de comprendre le fonctionnement du cryptage grâce au schéma ci-dessus. On aperçoit très clairement que la clé de ça réussite réside dans le nombre d'itérations des fonctions précedentes.

Notons que pour le décryptage, le schéma est quasi-identique. Il suffit d'utiliser les fonctions inverses.

2.4 La fonction de cryptage-décryptage d'un message entier

Un message complet est en général constitué de plus de 16 octets, il faut donc réitérer le schéma précédent à l'ensemble des blocs du message. Notons que si un message n'est pas divisible par 16, il faudra rajouter des éléments NULL au message afin de le crypter.

3 Les tests

3.1 Test de fonctionnement

Ce test a pour but de crypter-décrypter avec le même programme (que nous avons créé). Nous avons pour cela crypté-décrypté un message de 16 octets puis un message de 32 octets.

Voici le cryptage-décryptage du message de 16 octets et celui de 32 octets :

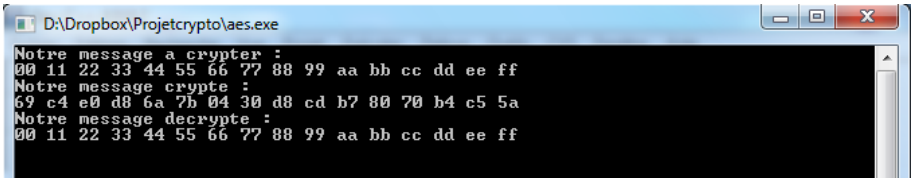

FIGURE 3.1 – Resultat du cryptage-décryptage d'un message de 16 octets.

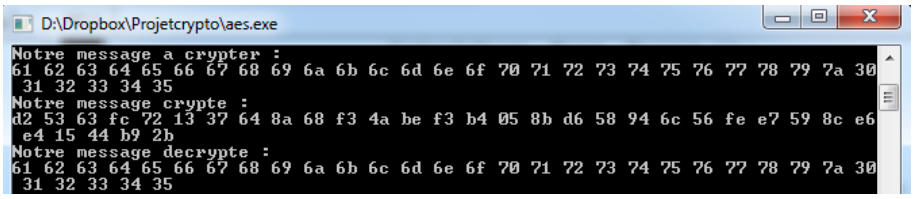

FIGURE 3.2 – Resultat du cryptage-décryptage d'un message de 32 octets.

3.2 Test de norme

Pour savoir si notre message crypté peut être décrypté (avec la clef) par n'importe quel programme, il faut suivre à la lettre le standard "Advanced Encryption Standard (AES)".
Pour verifier que cette norme est bien respectée, il faut verifier à chaque étape de notre cryptage que nous avons le même resultat que celui de la norme (cf annexe B).
Nous avons pour cela créé une nouvelle fonction permettant d'afficher en héxadécimal notre message de 16 octets dont voici le code source et le résultat obtenu :

```
1 void affichage(unsigned char *in){
2   int j;
3   for(j=0;j<TAILLEMESSAGE;j++){
4       printf("%02.2x",in[j]);
5   }
6   printf("\n");
7 }
```

Listing 3.1 – Fonction permettant l'affichage de TAILLEMESSAGE octets en héxadécimal.

```
1 printf("round[ %d].s_box ",round);
2 affichage(MESSAGE);
```

Listing 3.2 – Ce qu'il faut rajouter dans la fonction de cryptage-décryptage.

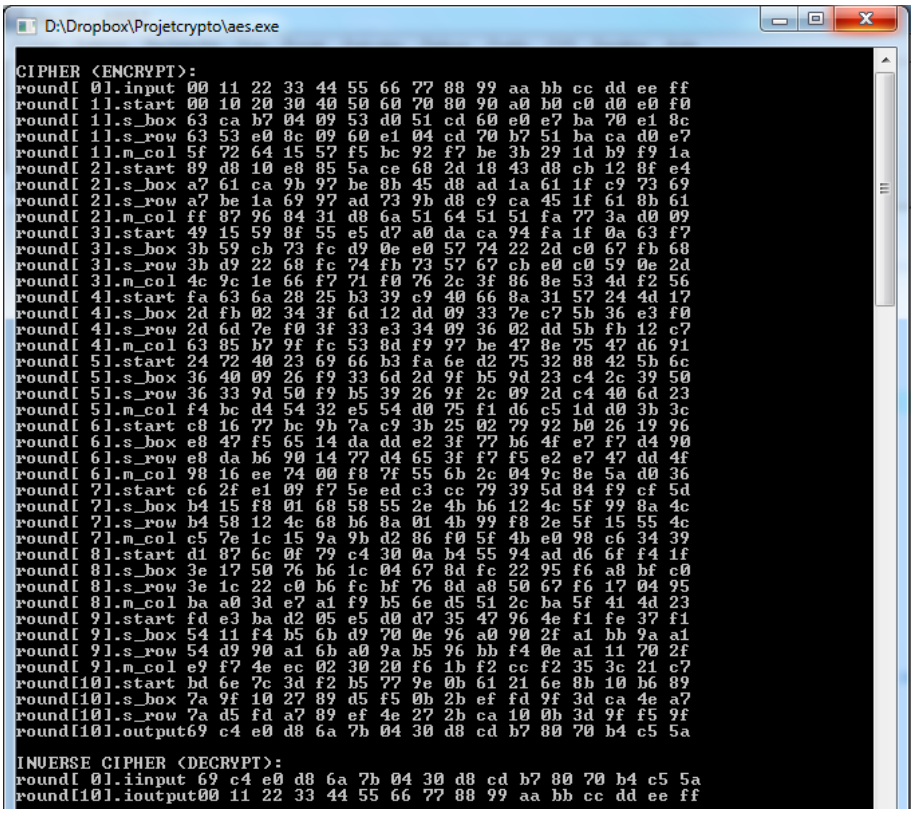

FIGURE 3.3 – Resultat du test (qui est bien en accord avec la norme en annexe B).

4 Mode d'emploi

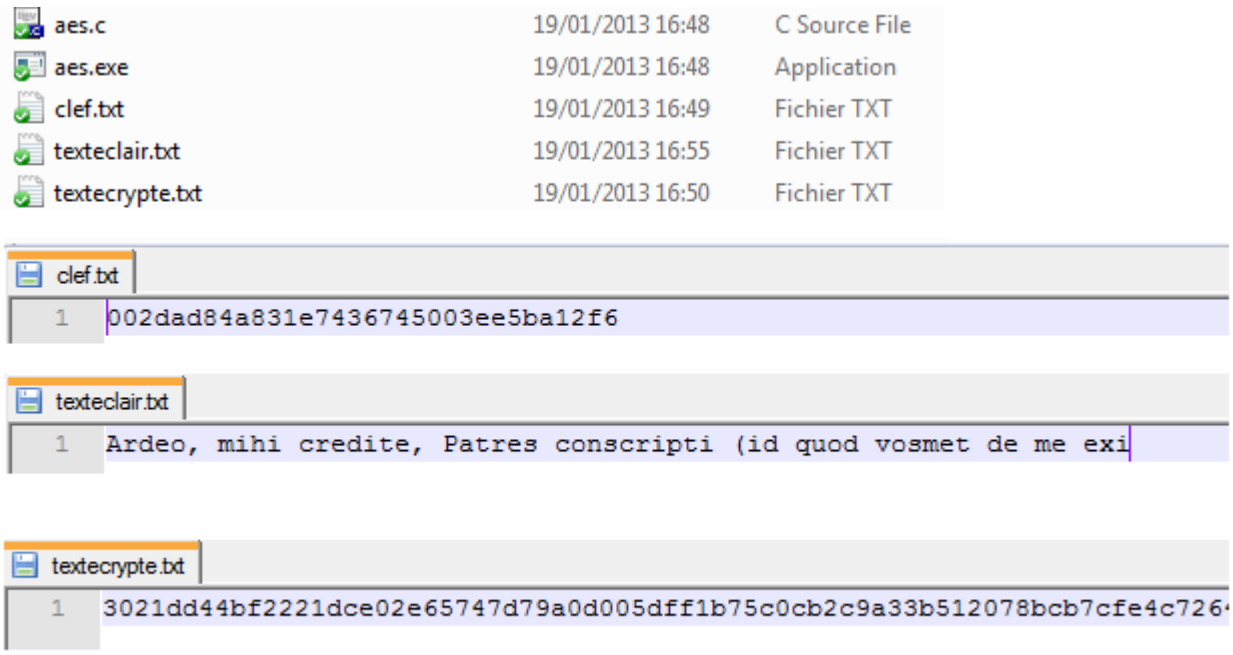

FIGURE 4.1 – Les différents fichiers du projet

Le projet est composé de 5 fichiers. Trois fichiers texte comportant respectivement la clé de cryptage, le texte clair (à crypter ou après décryptage) ainsi que le texte crypté.

On notera que le texte crypté ainsi que la clé sont enregistrés en hexadécimal. Cela permet d'éviter l'ambiguïté de certains caractères ASCII comme le saut de ligne.

Pour compiler ce projet, il suffit de complier le fichier aes.c. Il est important que les trois fichiers textes soient présents car sinon le programme de fonctionnera pas. Tout texte devant être crypté doit être copié au préalable dans texteclair.txt

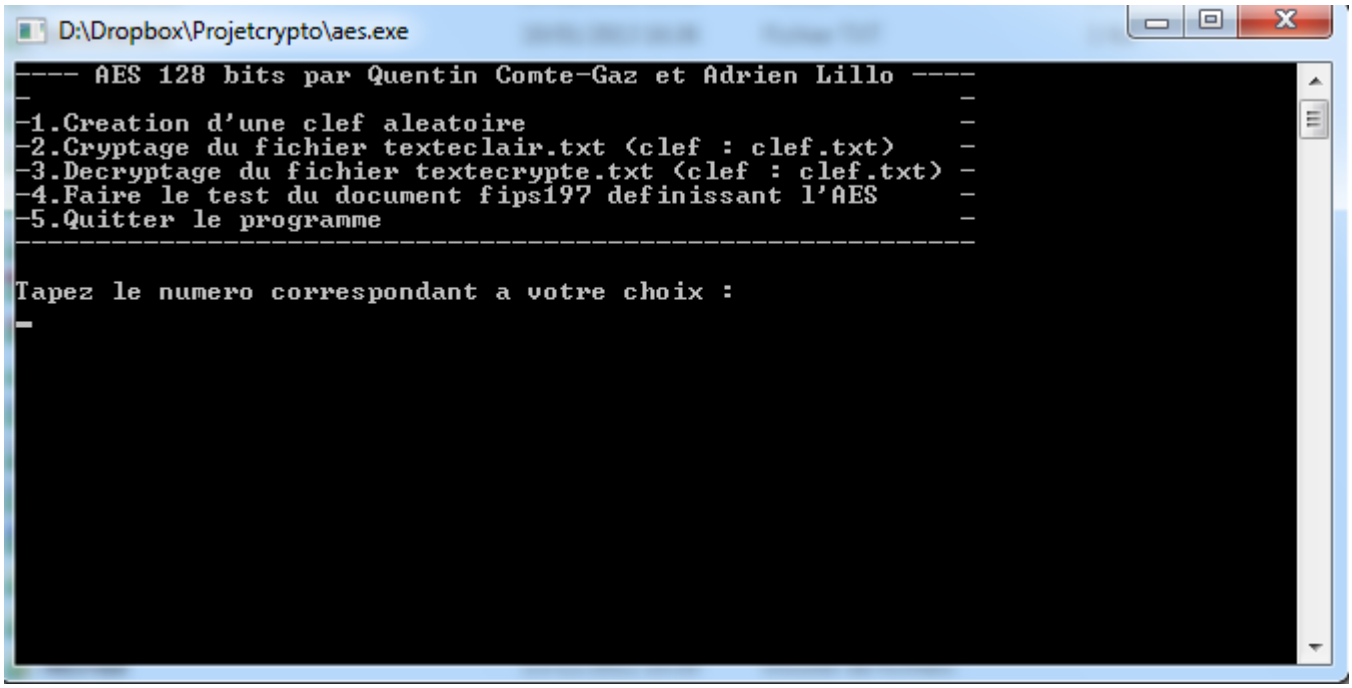

FIGURE 4.2 – Le menu du programme

1. Création d'une clef aléatoire : Génère une clé aléatoirement puis l'enregistre dans clef.txt

2. Cryptage du fichier texteclair.txt (clef : clef.txt) : Crypte le fichier texteclair.txt avec la clé enregistrée dans clef.txt. Le message crypté est enregistré dans textecrypte.txt

3. Decryptage du fichier textecrypte.txt (clef : clef.txt) : Décrypte le fichier textecrypte.txt avec la clé enregistrée dans clef.txt. Le message décrypté est enregistré dans texteclair.txt

4. Faire le test du document fips197 definissant l'AES : Déroule l'ensemble des tests de la norme AES sous vos yeux afin de vérifier le bon fonctionnement de l'agorithme de cryptage-décryptage.

A Les tables à utiliser : (inv)S-Box, Rcon et MulGF256

```
int sbox[256] =   {
    0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,
    0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,
    0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,
    0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
    0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,
    0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,
    0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
    0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
    0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
    0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,
    0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
    0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,
    0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,
    0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,
    0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
    0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16 };

int sboxinv[256] ={
    0x52 ,0x09 ,0x6a ,0xd5 ,0x30 ,0x36 ,0xa5 ,0x38 ,0xbf ,0x40 ,0xa3 ,0x9e ,0x81 ,0xf3 ,0xd7 ,0xfb
    ,0x7c ,0xe3 ,0x39 ,0x82 ,0x9b ,0x2f ,0xff ,0x87 ,0x34 ,0x8e ,0x43 ,0x44 ,0xc4 ,0xde ,0xe9 ,0xcb
    ,0x54 ,0x7b ,0x94 ,0x32 ,0xa6 ,0xc2 ,0x23 ,0x3d ,0xee ,0x4c ,0x95 ,0x0b ,0x42 ,0xfa ,0xc3 ,0x4e
    ,0x08 ,0x2e ,0xa1 ,0x66 ,0x28 ,0xd9 ,0x24 ,0xb2 ,0x76 ,0x5b ,0xa2 ,0x49 ,0x6d ,0x8b ,0xd1 ,0x25
    ,0x72 ,0xf8 ,0xf6 ,0x64 ,0x86 ,0x68 ,0x98 ,0x16 ,0xd4 ,0xa4 ,0x5c ,0xcc ,0x5d ,0x65 ,0xb6 ,0x92
    ,0x6c ,0x70 ,0x48 ,0x50 ,0xfd ,0xed ,0xb9 ,0xda ,0x5e ,0x15 ,0x46 ,0x57 ,0xa7 ,0x8d ,0x9d ,0x84
    ,0x90 ,0xd8 ,0xab ,0x00 ,0x8c ,0xbc ,0xd3 ,0x0a ,0xf7 ,0xe4 ,0x58 ,0x05 ,0xb8 ,0xb3 ,0x45 ,0x06
    ,0xd0 ,0x2c ,0x1e ,0x8f ,0xca ,0x3f ,0x0f ,0x02 ,0xc1 ,0xaf ,0xbd ,0x03 ,0x01 ,0x13 ,0x8a ,0x6b
    ,0x3a ,0x91 ,0x11 ,0x41 ,0x4f ,0x67 ,0xdc ,0xea ,0x97 ,0xf2 ,0xcf ,0xce ,0xf0 ,0xb4 ,0xe6 ,0x73
    ,0x96 ,0xac ,0x74 ,0x22 ,0xe7 ,0xad ,0x35 ,0x85 ,0xe2 ,0xf9 ,0x37 ,0xe8 ,0x1c ,0x75 ,0xdf ,0x6e
    ,0x47 ,0xf1 ,0x1a ,0x71 ,0x1d ,0x29 ,0xc5 ,0x89 ,0x6f ,0xb7 ,0x62 ,0x0e ,0xaa ,0x18 ,0xbe ,0x1b
    ,0xfc ,0x56 ,0x3e ,0x4b ,0xc6 ,0xd2 ,0x79 ,0x20 ,0x9a ,0xdb ,0xc0 ,0xfe ,0x78 ,0xcd ,0x5a ,0xf4
    ,0x1f ,0xdd ,0xa8 ,0x33 ,0x88 ,0x07 ,0xc7 ,0x31 ,0xb1 ,0x12 ,0x10 ,0x59 ,0x27 ,0x80 ,0xec ,0x5f
    ,0x60 ,0x51 ,0x7f ,0xa9 ,0x19 ,0xb5 ,0x4a ,0x0d ,0x2d ,0xe5 ,0x7a ,0x9f ,0x93 ,0xc9 ,0x9c ,0xef
    ,0xa0 ,0xe0 ,0x3b ,0x4d ,0xae ,0x2a ,0xf5 ,0xb0 ,0xc8 ,0xeb ,0xbb ,0x3c ,0x83 ,0x53 ,0x99 ,0x61
    ,0x17 ,0x2b ,0x04 ,0x7e ,0xba ,0x77 ,0xd6 ,0x26 ,0xe1 ,0x69 ,0x14 ,0x63 ,0x55 ,0x21 ,0x0c ,0x7d};
```

FIGURE A.1 – S-Box et S-Box inv

```
// Table de multiplication à utiliser dans InvMixColumns (tables de Galois(2^8)
//http://en.wikipedia.org/wiki/Rijndael_mix_columns

int Mul2[256] = {
    0x00,0x02,0x04,0x06,0x08,0x0a,0x0c,0x0e,0x10,0x12,0x14,0x16,0x18,0x1a,0x1c,0x1e,
    0x20,0x22,0x24,0x26,0x28,0x2a,0x2c,0x2e,0x30,0x32,0x34,0x36,0x38,0x3a,0x3c,0x3e,
    0x40,0x42,0x44,0x46,0x48,0x4a,0x4c,0x4e,0x50,0x52,0x54,0x56,0x58,0x5a,0x5c,0x5e,
    0x60,0x62,0x64,0x66,0x68,0x6a,0x6c,0x6e,0x70,0x72,0x74,0x76,0x78,0x7a,0x7c,0x7e,
    0x80,0x82,0x84,0x86,0x88,0x8a,0x8c,0x8e,0x90,0x92,0x94,0x96,0x98,0x9a,0x9c,0x9e,
    0xa0,0xa2,0xa4,0xa6,0xa8,0xaa,0xac,0xae,0xb0,0xb2,0xb4,0xb6,0xb8,0xba,0xbc,0xbe,
    0xc0,0xc2,0xc4,0xc6,0xc8,0xca,0xcc,0xce,0xd0,0xd2,0xd4,0xd6,0xd8,0xda,0xdc,0xde,
    0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xee,0xf0,0xf2,0xf4,0xf6,0xf8,0xfa,0xfc,0xfe,
    0x1b,0x19,0x1f,0x1d,0x13,0x11,0x17,0x15,0x0b,0x09,0x0f,0x0d,0x03,0x01,0x07,0x05,
    0x3b,0x39,0x3f,0x3d,0x33,0x31,0x37,0x35,0x2b,0x29,0x2f,0x2d,0x23,0x21,0x27,0x25,
    0x5b,0x59,0x5f,0x5d,0x53,0x51,0x57,0x55,0x4b,0x49,0x4f,0x4d,0x43,0x41,0x47,0x45,
    0x7b,0x79,0x7f,0x7d,0x73,0x71,0x77,0x75,0x6b,0x69,0x6f,0x6d,0x63,0x61,0x67,0x65,
    0x9b,0x99,0x9f,0x9d,0x93,0x91,0x97,0x95,0x8b,0x89,0x8f,0x8d,0x83,0x81,0x87,0x85,
    0xbb,0xb9,0xbf,0xbd,0xb3,0xb1,0xb7,0xb5,0xab,0xa9,0xaf,0xad,0xa3,0xa1,0xa7,0xa5,
    0xdb,0xd9,0xdf,0xdd,0xd3,0xd1,0xd7,0xd5,0xcb,0xc9,0xcf,0xcd,0xc3,0xc1,0xc7,0xc5,
    0xfb,0xf9,0xff,0xfd,0xf3,0xf1,0xf7,0xf5,0xeb,0xe9,0xef,0xed,0xe3,0xe1,0xe7,0xe5};

int Mul3[256] = {
    0x00,0x03,0x06,0x05,0x0c,0x0f,0x0a,0x09,0x18,0x1b,0x1e,0x1d,0x14,0x17,0x12,0x11,
    0x30,0x33,0x36,0x35,0x3c,0x3f,0x3a,0x39,0x28,0x2b,0x2e,0x2d,0x24,0x27,0x22,0x21,
    0x60,0x63,0x66,0x65,0x6c,0x6f,0x6a,0x69,0x78,0x7b,0x7e,0x7d,0x74,0x77,0x72,0x71,
    0x50,0x53,0x56,0x55,0x5c,0x5f,0x5a,0x59,0x48,0x4b,0x4e,0x4d,0x44,0x47,0x42,0x41,
    0xc0,0xc3,0xc6,0xc5,0xcc,0xcf,0xca,0xc9,0xd8,0xdb,0xde,0xdd,0xd4,0xd7,0xd2,0xd1,
    0xf0,0xf3,0xf6,0xf5,0xfc,0xff,0xfa,0xf9,0xe8,0xeb,0xee,0xed,0xe4,0xe7,0xe2,0xe1,
    0xa0,0xa3,0xa6,0xa5,0xac,0xaf,0xaa,0xa9,0xb8,0xbb,0xbe,0xbd,0xb4,0xb7,0xb2,0xb1,
    0x90,0x93,0x96,0x95,0x9c,0x9f,0x9a,0x99,0x88,0x8b,0x8e,0x8d,0x84,0x87,0x82,0x81,
    0x9b,0x98,0x9d,0x9e,0x97,0x94,0x91,0x92,0x83,0x80,0x85,0x86,0x8f,0x8c,0x89,0x8a,
    0xab,0xa8,0xad,0xae,0xa7,0xa4,0xa1,0xa2,0xb3,0xb0,0xb5,0xb6,0xbf,0xbc,0xb9,0xba,
    0xfb,0xf8,0xfd,0xfe,0xf7,0xf4,0xf1,0xf2,0xe3,0xe0,0xe5,0xe6,0xef,0xec,0xe9,0xea,
    0xcb,0xc8,0xcd,0xce,0xc7,0xc4,0xc1,0xc2,0xd3,0xd0,0xd5,0xd6,0xdf,0xdc,0xd9,0xda,
    0x5b,0x58,0x5d,0x5e,0x57,0x54,0x51,0x52,0x43,0x40,0x45,0x46,0x4f,0x4c,0x49,0x4a,
    0x6b,0x68,0x6d,0x6e,0x67,0x64,0x61,0x62,0x73,0x70,0x75,0x76,0x7f,0x7c,0x79,0x7a,
    0x3b,0x38,0x3d,0x3e,0x37,0x34,0x31,0x32,0x23,0x20,0x25,0x26,0x2f,0x2c,0x29,0x2a,
    0x0b,0x08,0x0d,0x0e,0x07,0x04,0x01,0x02,0x13,0x10,0x15,0x16,0x1f,0x1c,0x19,0x1a};

int Mul9[256] = {
    0x00,0x09,0x12,0x1b,0x24,0x2d,0x36,0x3f,0x48,0x41,0x5a,0x53,0x6c,0x65,0x7e,0x77,
    0x90,0x99,0x82,0x8b,0xb4,0xbd,0xa6,0xaf,0xd8,0xd1,0xca,0xc3,0xfc,0xf5,0xee,0xe7,
    0x3b,0x32,0x29,0x20,0x1f,0x16,0x0d,0x04,0x73,0x7a,0x61,0x68,0x57,0x5e,0x45,0x4c,
    0xab,0xa2,0xb9,0xb0,0x8f,0x86,0x9d,0x94,0xe3,0xea,0xf1,0xf8,0xc7,0xce,0xd5,0xdc,
    0x76,0x7f,0x64,0x6d,0x52,0x5b,0x40,0x49,0x3e,0x37,0x2c,0x25,0x1a,0x13,0x08,0x01,
    0xe6,0xef,0xf4,0xfd,0xc2,0xcb,0xd0,0xd9,0xae,0xa7,0xbc,0xb5,0x8a,0x83,0x98,0x91,
    0x4d,0x44,0x5f,0x56,0x69,0x60,0x7b,0x72,0x05,0x0c,0x17,0x1e,0x21,0x28,0x33,0x3a,
    0xdd,0xd4,0xcf,0xc6,0xf9,0xf0,0xeb,0xe2,0x95,0x9c,0x87,0x8e,0xb1,0xb8,0xa3,0xaa,
    0xec,0xe5,0xfe,0xf7,0xc8,0xc1,0xda,0xd3,0xa4,0xad,0xb6,0xbf,0x80,0x89,0x92,0x9b,
    0x7c,0x75,0x6e,0x67,0x58,0x51,0x4a,0x43,0x34,0x3d,0x26,0x2f,0x10,0x19,0x02,0x0b,
    0xd7,0xde,0xc5,0xcc,0xf3,0xfa,0xe1,0xe8,0x9f,0x96,0x8d,0x84,0xbb,0xb2,0xa9,0xa0,
    0x47,0x4e,0x55,0x5c,0x63,0x6a,0x71,0x78,0x0f,0x06,0x1d,0x14,0x2b,0x22,0x39,0x30,
    0x9a,0x93,0x88,0x81,0xbe,0xb7,0xac,0xa5,0xd2,0xdb,0xc0,0xc9,0xf6,0xff,0xe4,0xed,
    0x0a,0x03,0x18,0x11,0x2e,0x27,0x3c,0x35,0x42,0x4b,0x50,0x59,0x66,0x6f,0x74,0x7d,
    0xa1,0xa8,0xb3,0xba,0x85,0x8c,0x97,0x9e,0xe9,0xe0,0xfb,0xf2,0xcd,0xc4,0xdf,0xd6,
    0x31,0x38,0x23,0x2a,0x15,0x1c,0x07,0x0e,0x79,0x70,0x6b,0x62,0x5d,0x54,0x4f,0x46};
```

FIGURE A.2 – Tables de galois 1 3 9

```
int MulB[256] = {
    0x00,0x0b,0x16,0x1d,0x2c,0x27,0x3a,0x31,0x58,0x53,0x4e,0x45,0x74,0x7f,0x62,0x69,
    0xb0,0xbb,0xa6,0xad,0x9c,0x97,0x8a,0x81,0xe8,0xe3,0xfe,0xf5,0xc4,0xcf,0xd2,0xd9,
    0x7b,0x70,0x6d,0x66,0x57,0x5c,0x41,0x4a,0x23,0x28,0x35,0x3e,0x0f,0x04,0x19,0x12,
    0xcb,0xc0,0xdd,0xd6,0xe7,0xec,0xf1,0xfa,0x93,0x98,0x85,0x8e,0xbf,0xb4,0xa9,0xa2,
    0xf6,0xfd,0xe0,0xeb,0xda,0xd1,0xcc,0xc7,0xae,0xa5,0xb8,0xb3,0x82,0x89,0x94,0x9f,
    0x46,0x4d,0x50,0x5b,0x6a,0x61,0x7c,0x77,0x1e,0x15,0x08,0x03,0x32,0x39,0x24,0x2f,
    0x8d,0x86,0x9b,0x90,0xa1,0xaa,0xb7,0xbc,0xd5,0xde,0xc3,0xc8,0xf9,0xf2,0xef,0xe4,
    0x3d,0x36,0x2b,0x20,0x11,0x1a,0x07,0x0c,0x65,0x6e,0x73,0x78,0x49,0x42,0x5f,0x54,
    0xf7,0xfc,0xe1,0xea,0xdb,0xd0,0xcd,0xc6,0xaf,0xa4,0xb9,0xb2,0x83,0x88,0x95,0x9e,
    0x47,0x4c,0x51,0x5a,0x6b,0x60,0x7d,0x76,0x1f,0x14,0x09,0x02,0x33,0x38,0x25,0x2e,
    0x8c,0x87,0x9a,0x91,0xa0,0xab,0xb6,0xbd,0xd4,0xdf,0xc2,0xc9,0xf8,0xf3,0xee,0xe5,
    0x3c,0x37,0x2a,0x21,0x10,0x1b,0x06,0x0d,0x64,0x6f,0x72,0x79,0x48,0x43,0x5e,0x55,
    0x01,0x0a,0x17,0x1c,0x2d,0x26,0x3b,0x30,0x59,0x52,0x4f,0x44,0x75,0x7e,0x63,0x68,
    0xb1,0xba,0xa7,0xac,0x9d,0x96,0x8b,0x80,0xe9,0xe2,0xff,0xf4,0xc5,0xce,0xd3,0xd8,
    0x7a,0x71,0x6c,0x67,0x56,0x5d,0x40,0x4b,0x22,0x29,0x34,0x3f,0x0e,0x05,0x18,0x13,
    0xca,0xc1,0xdc,0xd7,0xe6,0xed,0xf0,0xfb,0x92,0x99,0x84,0x8f,0xbe,0xb5,0xa8,0xa3};

int MulD[256] = {
    0x00,0x0d,0x1a,0x17,0x34,0x39,0x2e,0x23,0x68,0x65,0x72,0x7f,0x5c,0x51,0x46,0x4b,
    0xd0,0xdd,0xca,0xc7,0xe4,0xe9,0xfe,0xf3,0xb8,0xb5,0xa2,0xaf,0x8c,0x81,0x96,0x9b,
    0xbb,0xb6,0xa1,0xac,0x8f,0x82,0x95,0x98,0xd3,0xde,0xc9,0xc4,0xe7,0xea,0xfd,0xf0,
    0x6b,0x66,0x71,0x7c,0x5f,0x52,0x45,0x48,0x03,0x0e,0x19,0x14,0x37,0x3a,0x2d,0x20,
    0x6d,0x60,0x77,0x7a,0x59,0x54,0x43,0x4e,0x05,0x08,0x1f,0x12,0x31,0x3c,0x2b,0x26,
    0xbd,0xb0,0xa7,0xaa,0x89,0x84,0x93,0x9e,0xd5,0xd8,0xcf,0xc2,0xe1,0xec,0xfb,0xf6,
    0xd6,0xdb,0xcc,0xc1,0xe2,0xef,0xf8,0xf5,0xbe,0xb3,0xa4,0xa9,0x8a,0x87,0x90,0x9d,
    0x06,0x0b,0x1c,0x11,0x32,0x3f,0x28,0x25,0x6e,0x63,0x74,0x79,0x5a,0x57,0x40,0x4d,
    0xda,0xd7,0xc0,0xcd,0xee,0xe3,0xf4,0xf9,0xb2,0xbf,0xa8,0xa5,0x86,0x8b,0x9c,0x91,
    0x0a,0x07,0x10,0x1d,0x3e,0x33,0x24,0x29,0x62,0x6f,0x78,0x75,0x56,0x5b,0x4c,0x41,
    0x61,0x6c,0x7b,0x76,0x55,0x58,0x4f,0x42,0x09,0x04,0x13,0x1e,0x3d,0x30,0x27,0x2a,
    0xb1,0xbc,0xab,0xa6,0x85,0x88,0x9f,0x92,0xd9,0xd4,0xc3,0xce,0xed,0xe0,0xf7,0xfa,
    0xb7,0xba,0xad,0xa0,0x83,0x8e,0x99,0x94,0xdf,0xd2,0xc5,0xc8,0xeb,0xe6,0xf1,0xfc,
    0x67,0x6a,0x7d,0x70,0x53,0x5e,0x49,0x44,0x0f,0x02,0x15,0x18,0x3b,0x36,0x21,0x2c,
    0x0c,0x01,0x16,0x1b,0x38,0x35,0x22,0x2f,0x64,0x69,0x7e,0x73,0x50,0x5d,0x4a,0x47,
    0xdc,0xd1,0xc6,0xcb,0xe8,0xe5,0xf2,0xff,0xb4,0xb9,0xae,0xa3,0x80,0x8d,0x9a,0x97};

int MulE[256] = {
    0x00,0x0e,0x1c,0x12,0x38,0x36,0x24,0x2a,0x70,0x7e,0x6c,0x62,0x48,0x46,0x54,0x5a,
    0xe0,0xee,0xfc,0xf2,0xd8,0xd6,0xc4,0xca,0x90,0x9e,0x8c,0x82,0xa8,0xa6,0xb4,0xba,
    0xdb,0xd5,0xc7,0xc9,0xe3,0xed,0xff,0xf1,0xab,0xa5,0xb7,0xb9,0x93,0x9d,0x8f,0x81,
    0x3b,0x35,0x27,0x29,0x03,0x0d,0x1f,0x11,0x4b,0x45,0x57,0x59,0x73,0x7d,0x6f,0x61,
    0xad,0xa3,0xb1,0xbf,0x95,0x9b,0x89,0x87,0xdd,0xd3,0xc1,0xcf,0xe5,0xeb,0xf9,0xf7,
    0x4d,0x43,0x51,0x5f,0x75,0x7b,0x69,0x67,0x3d,0x33,0x21,0x2f,0x05,0x0b,0x19,0x17,
    0x76,0x78,0x6a,0x64,0x4e,0x40,0x52,0x5c,0x06,0x08,0x1a,0x14,0x3e,0x30,0x22,0x2c,
    0x96,0x98,0x8a,0x84,0xae,0xa0,0xb2,0xbc,0xe6,0xe8,0xfa,0xf4,0xde,0xd0,0xc2,0xcc,
    0x41,0x4f,0x5d,0x53,0x79,0x77,0x65,0x6b,0x31,0x3f,0x2d,0x23,0x09,0x07,0x15,0x1b,
    0xa1,0xaf,0xbd,0xb3,0x99,0x97,0x85,0x8b,0xd1,0xdf,0xcd,0xc3,0xe9,0xe7,0xf5,0xfb,
    0x9a,0x94,0x86,0x88,0xa2,0xac,0xbe,0xb0,0xea,0xe4,0xf6,0xf8,0xd2,0xdc,0xce,0xc0,
    0x7a,0x74,0x66,0x68,0x42,0x4c,0x5e,0x50,0x0a,0x04,0x16,0x18,0x32,0x3c,0x2e,0x20,
    0xec,0xe2,0xf0,0xfe,0xd4,0xda,0xc8,0xc6,0x9c,0x92,0x80,0x8e,0xa4,0xaa,0xb8,0xb6,
    0x0c,0x02,0x10,0x1e,0x34,0x3a,0x28,0x26,0x7c,0x72,0x60,0x6e,0x44,0x4a,0x58,0x56,
    0x37,0x39,0x2b,0x25,0x0f,0x01,0x13,0x1d,0x47,0x49,0x5b,0x55,0x7f,0x71,0x63,0x6d,
    0xd7,0xd9,0xcb,0xc5,0xef,0xe1,0xf3,0xfd,0xa7,0xa9,0xbb,0xb5,0x9f,0x91,0x83,0x8d};
```

FIGURE A.3 – Tables de galois B D E

```
int Rcon[255] = {
    0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a,
    0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39,
    0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a,
    0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8,
    0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef,
    0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc,
    0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,
    0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3,
    0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94,
    0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
    0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35,
    0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f,
    0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04,
    0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63,
    0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd,
    0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb  };
```

FIGURE A.4 – Rcon

B Les vecteurs de test

AES-128 (Nk=4, Nr=10)

PLAINTEXT : 00112233445566778899aabbccddeeff
KEY : 000102030405060708090a0b0c0d0e0f

CIPHER (ENCRYPT) :
round[0].input 00112233445566778899aabbccddeeff
round[1].start 000102030405060708090a0b0c0d0e0f0
round[1].s_box 63cab7040953d051cd60e0e7ba70e18c
round[1].s_row 6353e08c0960e104cd70b751bacad0e7
round[1].m_col 5f72641557f5bc92f7be3b291db9f91a
round[2].start 89d810e8855ace682d1843d8cb128fe4
round[2].s_box a761ca9b97be8b45d8ad1a611fc97369
round[2].s_row a7be1a6997ad739bd8c9ca451f618b61
round[2].m_col ff87968431d86a51645151fa773ad009
round[3].start 4915598f55e5d7a0daca94fa1f0a63f7
round[3].s_box 3b59cb73fcd90ee05774222dc067fb68
round[3].s_row 3bd92268fc74fb735767cbe0c0590e2d
round[3].m_col 4c9c1e66f771f0762c3f868e534df256
round[4].start fa636a2825b339c940668a3157244d17
round[4].s_box 2dfb02343f6d12dd09337ec75b36e3f0
round[4].s_row 2d6d7ef03f33e334093602dd5bfb12c7
round[4].m_col 6385b79ffc538df997be478e7547d691
round[5].start 247240236966b3fa6ed2753288425b6c
round[5].s_box 36400926f9336d2d9fb59d23c42c3950
round[5].s_row 36339d50f9b539269f2c092dc4406d23
round[5].m_col f4bcd45432e554d075f1d6c51dd03b3c
round[6].start c81677bc9b7ac93b25027992b0261996
round[6].s_box e847f56514dadde23f77b64fe7f7d490
round[6].s_row e8dab6901477d4653ff7f5e2e747dd4f
round[6].m_col 9816ee7400f87f556b2c049c8e5ad036
round[7].start c62fe109f75eedc3cc79395d84f9cf5d
round[7].s_box b415f8016858552e4bb6124c5f998a4c
round[7].s_row b458124c68b68a014b99f82e5f15554c
round[7].m_col c57e1c159a9bd286f05f4be098c63439
round[8].start d1876c0f79c4300ab45594add66ff41f
round[8].s_box 3e175076b61c04678dfc2295f6a8bfc0
round[8].s_row 3e1c22c0b6fcbf768da85067f6170495
round[8].m_col baa03de7a1f9b56ed5512cba5f414d23
round[9].start fde3bad205e5d0d73547964ef1fe37f1
round[9].s_box 5411f4b56bd9700e96a0902fa1bb9aa1
round[9].s_row 54d990a16ba09ab596bbf40ea111702f
round[9].m_col e9f74eec023020f61bf2ccf2353c21c7
round[10].start bd6e7c3df2b5779e0b61216e8b10b689
round[10].s_box 7a9f102789d5f50b2beffd9f3dca4ea7
round[10].s_row 7ad5fda789ef4e272bca100b3d9ff59f
round[10].output 69c4e0d86a7b0430d8cdb78070b4c55a

INVERSE CIPHER (DECRYPT) :
round[0].iinput 69c4e0d86a7b0430d8cdb78070b4c55a
...
round[10].ioutput 00112233445566778899aabbccddeeff