

### **Fiche de synthèse n° 1**

Développement d'un logiciel de sécurisation des données confidentielles, chiffrement avec plusieurs algorithmes.

### **Objectifs**

- Implémenter l'algorithme AES
- Implémenter l'algorithme XOR
- Créer une interface graphique avec GTK +

### **Conditions de réalisation**

Outils :

- C ansi
- Eclipse
- GTK2.0 +

Date de début : septembre 2009

Durée : 80h

### **Compétences concernées**

- ✓ C31 : Gérer un projet de développement de logiciels.
- ✓ C32 : Développer à l'aide d'un langage de programmation procédural.
- ✓ C37 : Mettre au point et maintenir une application.
- ✓ C39 : Maîtriser le poste de développement et son environnement.

## Présentation

Une organisation militaire, un état, ou une entreprise a très souvent des informations sensibles quelle souhaite garder en son unique possession : prototypes, données des clients, comptabilité, dossiers classés secrets.

Il est possible de les chiffrer de façon à les rendre illisibles par les personnes externes, c'est-à-dire les rendre illisibles sans le mot de passe associé au chiffrement.

Le but de de l'application est de permettre simplement à l'utilisateur de chiffrer et déchiffrer n'importe quel fichier.

## Choix des algorithmes

XOR est un des algorithmes de chiffrement les plus simples. Comme son nom l'indique il consiste à exécuter un ou logique entre la chaîne à chiffrer et la clé, la même opération permet le déchiffrement. Il est rapide, mais facile à casser par une cryptanalyse, donc non fiable.

Advanced Encryption Standard ( AES ) est un standard mis en place en 2000 pour le gouvernement américain pour remplacer DES ( datant de 1970 ) à la suite d'un concours entre 15 algorithmes. L'algorithme belge Rijndael a été choisi parmi les cinq finalistes ( MARS, RC6, Rijndael, Serpent, et Twofish ), car il offre une rapidité supérieure aux autres pour une sécurité équivalente.

Cet algorithme symétrique de chiffrement par blocs est toujours utilisé pour les dossiers secrets et top secrets du gouvernement américain, il est considéré comme fiable, il est pour l'instant extrêmement dur à casser.

Le choix du langage s'est porté sur le C, car nous avons besoin d'un langage bas niveau pour travailler sur les fichiers binaires, et les langages objets sont plus gourmands en ressources. De plus, le C ANSI est multiplateforme.

## Algorithme XOR

« XOR » signifie « ou exclusif », le tableau à droite représente l'algèbre de Boole correspondant.

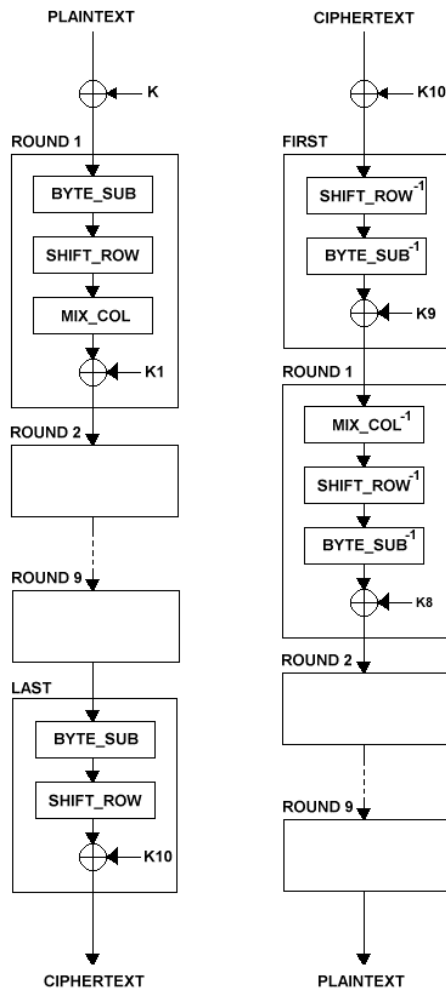
Pour chiffrer une chaîne, il suffit d'appliquer un ou exclusif entre chaque caractère de la chaîne et le caractère de la clé correspondant. La clé est préalablement entendue pour faire la même longueur que la chaîne.

Pour déchiffrer, il suffit d'appliquer le même traitement à la chaîne chiffrée, avec la même clé.

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

## Implémentation de AES

Le schéma page suivante présente le fonctionnement général d'AES, il contient quatre fonctions et leurs inverses respectifs, qui travaillent toutes sur des blocs de 128 bits et une clé de même taille. Un bloc de 128 bits est représenté par une matrice 4x4 d'octets dans la description officielle de l'algorithme. 10 Passages sont effectués, ils utilisent à chaque fois une clé différente. Les 10 clés sont générées à partir du bloc K grâce à la procédure KeyExpansion().



L'algorithme est dans un premier temps implémenté grâce à la documentation<sup>1</sup> du NIST ( National Institute of Standards and Technology ) sans interface graphique, en parallèle de XOR.

Pour clarifier le code, un type appelé « byte » est créé ( octet en français ) il correspond à 8 bits. En C, « unsigned char », est un entier codé sur 8 bits et destiné originellement à stocker un code ASCII.

```
typedef unsigned char byte ;
```

Un bloc de 128 bits est donc formé par un tableau de 16 octets.

La procédure ShiftRows permet d'exécuter une rotation de chaque ligne sur un bloc, la première n'est pas décalée, la deuxième subit une rotation d'une case vers la gauche, la troisième de deux cases, et la dernière de trois cases :

avant					après			
0	4	8	12	→	0	4	8	12
1	5	9	13		5	9	13	1
2	6	10	14		10	14	2	6
3	7	11	15		15	3	7	11

```
void ShiftRows( byte* state )
{
    /* Copie les éléments un à un */
    for( i = 0 ; i < 16 ; i++ )
        temp[i] = *( state + i ) ;
    /* La première ligne ne change pas */
    /* Rotation de la deuxième ligne */
    *( state + 1 ) = temp[5] ;
    *( state + 5 ) = temp[9] ;
    *( state + 9 ) = temp[13] ;
    *( state + 13 ) = temp[1] ;
    /* Rotation de la troisième ligne */
    *( state + 2 ) = temp[10] ;
    *( state + 6 ) = temp[14] ;
    *( state + 10 ) = temp[2] ;
    *( state + 14 ) = temp[6] ;
    /* Rotation de la quatrième ligne */
    *( state + 3 ) = temp[15] ;
    *( state + 7 ) = temp[3] ;
    *( state + 11 ) = temp[7] ;
    *( state + 15 ) = temp[11] ;
}
```

Une fois l'implémentation des algorithmes et les tests unitaires validés, une interface graphique est créée avec la librairie GTK.

<sup>1</sup><http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

## Bourrage

---

En anglais « Padding ».

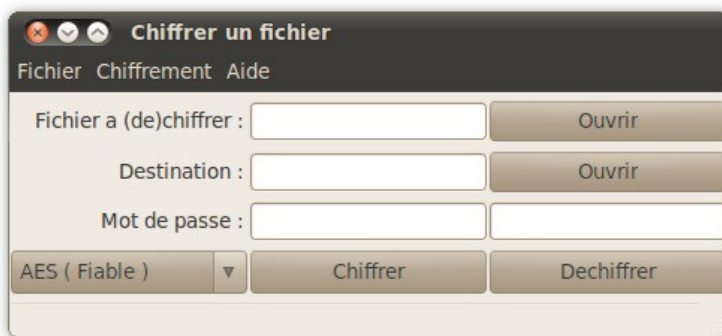
L'algorithme AES travaille sur des blocs de 128 bits, or la taille du fichier fourni n'est pas forcément un multiple de 128, il faut donc ajouter un « padding » à la fin et mémoriser le nombre d'octets ajoutés pour le déchiffrement. On complète le fichier en écrivant dans chaque octet le nombre d'octets ajoutés. Ce « padding » doit être supprimé lors du déchiffrement

La clé de chiffrement doit aussi faire 128 bits avant son extension, elle donc complétée par des valeurs fixées arbitrairement.

La documentation d'AES ne suffit donc pas, il faut réaliser des algorithmes de découpage et de bourrage des données.

## Aspect fonctionnel

---



Une interface simple permet d'utiliser le programme, le mot de passe doit être renseigné deux fois lors du chiffrement pour vérification, une liste déroulante permet de choisir l'algorithme, et la barre d'état affiche les messages de retour.

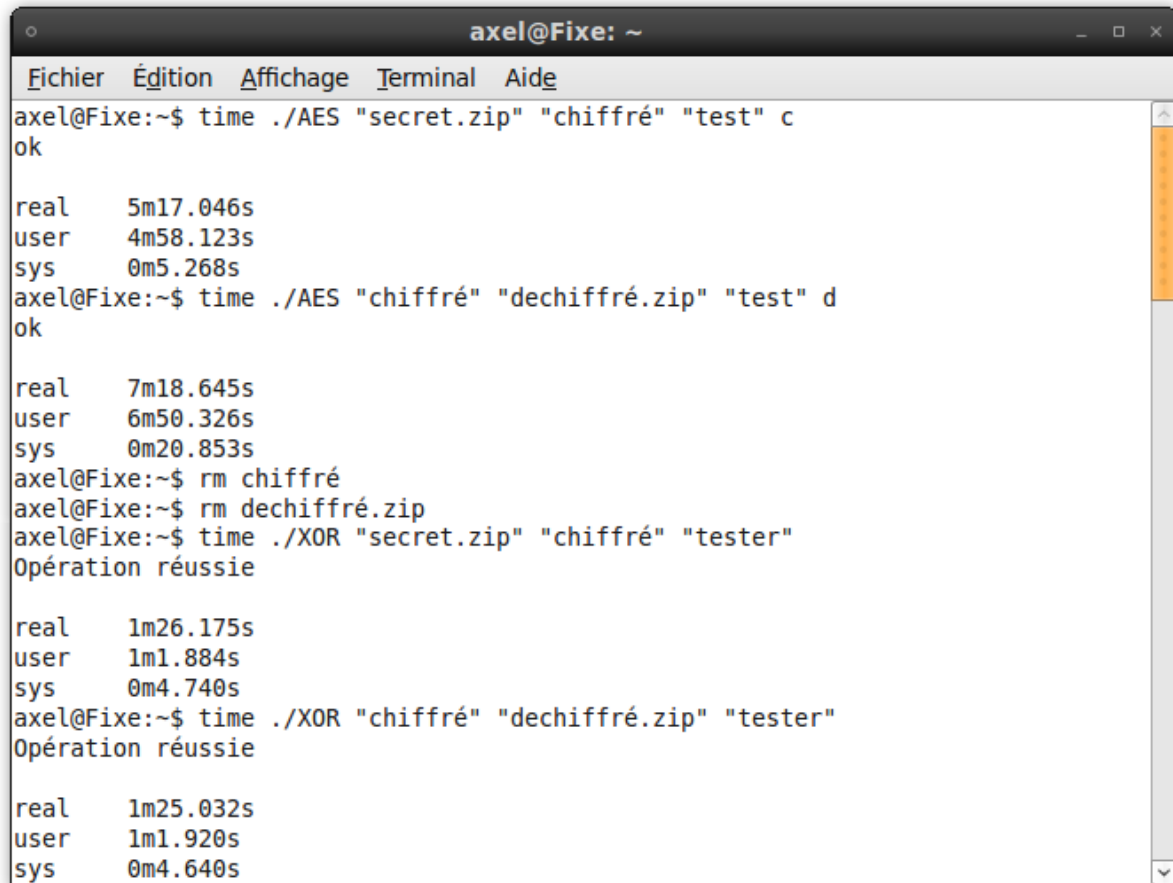
## Tests & Performances

---

Un test unitaire est réalisé sur le code de l'algorithme AES en ligne de commande pour vérifier la corrélation entre l'algorithme et l'application. Il permet de visualiser étape par étape le chiffrement d'un bloc et de vérifier grâce à la documentation qui propose le même test.

Pour comparer les deux algorithmes, des tests de performances sont effectués ( sur un ordinateur de bureau d'une puissance « standard » ) avec la commande « time » de Linux, permettant d'afficher le temps d'exécution. Pour être représentatif, les tests sont effectués sur une archive de taille importante ( 937Mio )

Les premiers tests étaient effectués avec un affichage des blocs chiffrés dans l'invite de commande, mais celui-ci ralentissait considérablement l'application ( environ 30x plus lent ), il a donc été supprimé.



```
axel@Fixe:~$ time ./AES "secret.zip" "chiffré" "test" c
ok

real    5m17.046s
user    4m58.123s
sys     0m5.268s
axel@Fixe:~$ time ./AES "chiffré" "dechiffré.zip" "test" d
ok

real    7m18.645s
user    6m50.326s
sys     0m20.853s
axel@Fixe:~$ rm chiffré
axel@Fixe:~$ rm dechiffré.zip
axel@Fixe:~$ time ./XOR "secret.zip" "chiffré" "tester"
Opération réussie

real    1m26.175s
user    1m1.884s
sys     0m4.740s
axel@Fixe:~$ time ./XOR "chiffré" "dechiffré.zip" "tester"
Opération réussie

real    1m25.032s
user    1m1.920s
sys     0m4.640s
```

On note que l'algorithme XOR est nettement plus rapide que AES, c'est grâce à sa simplicité, mais rappelons-le, XOR n'est pas fiable, et peut facilement être décrypté par une personne malveillante.

On remarque aussi que AES est plus lent au déchiffrement. Cela est dû au padding, car chaque blocs est vérifié pour savoir si ce n'est pas le dernier.

## Conclusion

---

Cette activité fut très intéressante à développer, car elle m'a permis d'appliquer et de comprendre les concepts du chiffrement. Le programme est fonctionnel, mais des améliorations peuvent être apportées : portage sous Windows, et compression des données avant le chiffrement, il est effet prouvé qu'une donnée compressée avant d'être chiffrée est moins vulnérable aux attaques.