UE IN608N (CRYPTO): Attaque intégrale sur une version réduite d'un AES-128

Rapport IN608

Docherty Ronan, Ejjed Zakaria, Gago Jérémy, Guerin Raphaël, Thienard Rodolphe

10 Mai 2022

INTRODUCTION

Le but de ce Projet était de d'implémenter une version réduite d'un AES-128 avec un mode de chiffrement OFB et/ou CTR en C. Dans notre cas nous avons choisit le chiffrement CTR. Nous avons choisit le mode CTR pour des questions d'optimisations, car il est parallélisable et plus rapide que le mode OFB.
Mais aussi et surtout réalisé l'attaque intégrale sur 4 tours de l'AES-128 en mode de chiffrement CTR.

IMPLÉMENTATION DE L'AES-128

Gestion des commandes

Nous avons créer le fichier command.c et command.h pour faire la gestion des commandes. En fonction de la demande...

Un tour d'AES-128 est composé de 4 fonctions :

1. SubOctet : Substitue les octets de la matrice d'état.
2. DecaleLignes : Décale les lignes de la matrice d'état.
3. MelangeColonnes : Mélanges les colonnes de la matrice d'état.
4. AjouteCleTour : Ajoute la clé de tour à la matrice d'état.

Nous commencerons donc par la fonction SubOctet puisque c'est la première fonction du tour.

SubOctet

La fonction `void suboctet(int **matrix)` nous permet de substituer l'octet contenu dans une case de notre matrice, à l'aide de la table de substitution de l'AES, qui nous donnera un octet différent du précédent.

DecaleLignes

La fonction `decale_lignes(int **matrix);` permet de décaler les lignes de la matrice d'état en fonction de la ligne. Pour faire cela on créer une matrices (`int **`) temporaire qui stockera le décalage des lignes. Ces modifications serons recopié dans la matrices d'état.

MelangeColonnes

La fonction `void melange_colonnes(int **matrix)` permet, à l'aide de deux matrices(`operation_double` et `operation_triple`), d'effectuer certaines opérations pour donner de nouvelles valeurs aux lignes de chaque colonne de notre matrice. Les opérations sont les suivantes :

1. $$B'_{0,1} = (2 \star B_{0,1}) \oplus (3 \star B_{1,1}) \oplus B_{2,1} \oplus B_{3,1}$$
   La première ligne prendra la valeur de : l'octet renvoyé par la case correspondant à l'octet de la ligne 0 de cette colonne dans `operation_double` XOR l'octet renvoyé par la case correspondant à l'octet de la ligne 1 dans `operation_triple` XOR l'octet de la ligne 2 XOR l'octet de la ligne 3.

2. $$B'_{1,1} = B_{0,1} \oplus (2 \star B_{1,1}) \oplus (3 \star B_{2,1}) \oplus B_{3,1}$$
   La seconde prendra la valeur de : l'octet de la ligne 0 XOR l'octet renvoyé par la case correspondant à l'octet de la ligne 1 dans `operation_double` XOR l'octet renvoyé par la case correspondant à l'octet de la ligne 2 dans `operation_triple` XOR l'octet de la ligne 3

3. $$B'_{2,1} = B_{0,1} \oplus B_{1,1}(2 \star B_{2,1}) \oplus (3 \star B_{3,1})$$
   La troisième prendra la valeur de : l'octet de la ligne 0 XOR l'octet de la ligne 1 XOR l'octet renvoyé par la case correspondant à l'octet de la ligne 2 dans `operation_double` XOR l'octet renvoyé par la case correspondant à l'octet de la ligne 3 dans `operation_triple`

4. $$B'_{3,1} = (3 \star B_{0,1}) \oplus B_{1,1} \oplus B_{2,1}(2 \star B_{3,1})$$
   La dernière prendra la valeur de : l'octet renvoyé par la case correspondant à l'octet de la ligne 0 dans `operation_triple` XOR l'octet de la ligne 1 XOR l'octet de la ligne 2 XOR l'octet renvoyé par la case correspondant à l'octet de la ligne 3 dans `operation_double`

AjouteCleTour

La fonction `addroundkey(int **matrix, int **key);` fait un XOR entre la matrice d'état courante et la matrice de la clé de tour :
`matrix[i][j] = matrix[i][j] ^ key[i][j];`
Comme les deux matrices sont en hexadécimales il existe le XOR en C qui est le : " ^ ".

Les Tours