## RAPPORT PROJET CRYPTOGRAPHIE IN603 CIESLA Julie LDD BI

## I. DESCRIPTION ATTAQUE PAR LE MILIEU

Pour réaliser l'attaque par le milieu, j'ai tout d'abord utilisé la fonction de chiffrement pour générer tous les chiffrés c' possibles à partir de mon message clair m1 et de toutes les clés possibles, c'est-à-dire, pour les clés allant de 0 à ffffff en hexadécimal. Ensuite j'ai fait de même avec la fonction de déchiffrement pour générer tous les clairs m' possibles en utilisant cette fois si c1.

Dans un second temps, j'ai trié mes listes Lc et Lm via la fonction intégrée dans Python qui permet de faire un tri optimal en O(nlog(n)). Ensuite j'ai comparé mes deux listes Lm et Lc contenant tous les m' et c'. Pour comparer j'ai écrit une fonction qui parcourt tous les éléments de Lm et puis qui parcours les éléments de Lc entre j\_prec et la fin de la liste tant que m' > c'. La valeur de j\_prec est incrémentée à chaque fois qu'on parcourt un nouvel élément dans Lc à moins que m' = c', dans quel cas la valeur de j\_prec est égal à la position de c' dans la liste (si plusieurs même valeur de c' dans la liste alors prend la plus petite position). Dès qu'un élément commun est trouvé on stocke dans une liste l\_communs la clé correspondant à m' et la clé correspondant à c'. Ainsi on obtient une comparaison d'éléments plus petite que O(n²).

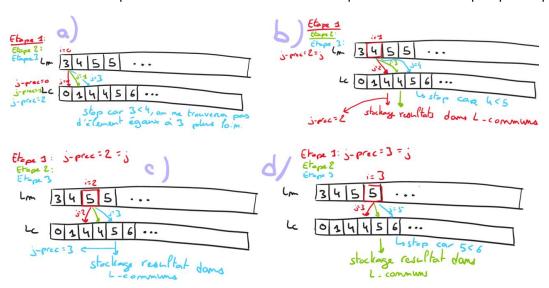


Figure 1: Schéma expliquant la comparaison d'éléments dans Lm et Lc. a) Cas où l'élément de Lm n'est égale à aucun autre dans Lc. b) Cas où un élément dans Lm est égal à plusieurs autres dans Lc. d) Cas où l'élément dans Lm est égal à un autre élément dans Lc. d) Cas où l'élément dans Lm est égal au précédent de la liste et où il est aussi égal à un élément dans Lc.

En faisant l'attaque par le milieu ainsi j'ai trouvé 16.777.216 couples de clés potentielles.

Par la suite, pour trouver quel couple de clé (k1, k2) était le bon, j'ai utilisé la fonction de chiffrement avec le message clair m2 et la clé potentiel k1 ce qui m'a donné un chiffré intermédiaire  $c_i$ , puis j'ai rechiffré avec comme message clair  $c_i$  et la clé potentiel k2 ce qui m'a donné un chiffré  $c_i$ '. Si  $c_i$ ' est égale à c2, alors c'est que (k1,k2) est le couple de clés qui a servi à chiffrer c1 et c2 à partir de m1 et m2. (Couples de bi-clés trouvés écrit dans annexe.txt)

## II. CHOIX IMPLEMENTATION

Au départ j'avais implémenté mon programme en utilisant pas mal de listes et beaucoup de fonctions pour faire le chiffrement et déchiffrement. Le problème était que mes fonctions étaient trop longues, en effet, mon programme mettait 5h30 pour faire l'attaque par le milieu. J'ai alors décidé de supprimer la plupart des fonctions et d'inclure le code des fonctions directement dans ma partie de code principale et j'ai aussi utilisé des string à la place des listes à certains endroits. De plus, j'ai utilisé du multiprocessing, au lieu d'exécuter mon programme sur 1 seul cpu, certaines parties s'exécutent sur 4 ou 8 cpu pour accélérer les calculs. Ainsi mon programme ne met plus que 20 minutes pour s'exécuter. Le problème c'est que toutes ces optimisations ont rendu mon code assez illisible, c'est pour cela que j'ai joint dans l'archive mon code non optimisé mais beaucoup plus lisible.

Il reste un certain nombre d'éléments à optimiser dans mon travail, il faudrait en effet, réduire encore davantage l'utilisation des listes qui est couteuse, il faudrait que j'évite d'écrire et de lire dans des fichiers .txt ce qui me ferait gagner environ 5 minutes, il faudrait aussi que je fasse des tests pour savoir le nombre de cpu le plus optimal pour exécuter mon programme (problème de saturation de la mémoire à certains moments ce qui ralentit le code).