EXPLICATION DE L'ATTAQUE ET L'IMPLÉMENTATION

pour l'implémentation de l'attaque, on a utilisé la structure de données main_keys qui permet de stocker la clé maitre et est composée d'un tableau 20 cages de type uint4_t. Ce dernier est une structure donnée composée d'une variable de 4 bits qui peut contenir des valeurs comprises entre [0..9,a..f].Le tableau permet par la suite de représenter les 80 bits de la clé maître.

les inverses de la substitution et de la permutation, on les obtient comme suit :
par exemple si perm[1] = 6 alors l'inverse de la permutation de 6 est inv_perm[6] = 1, de même pour la substitution.

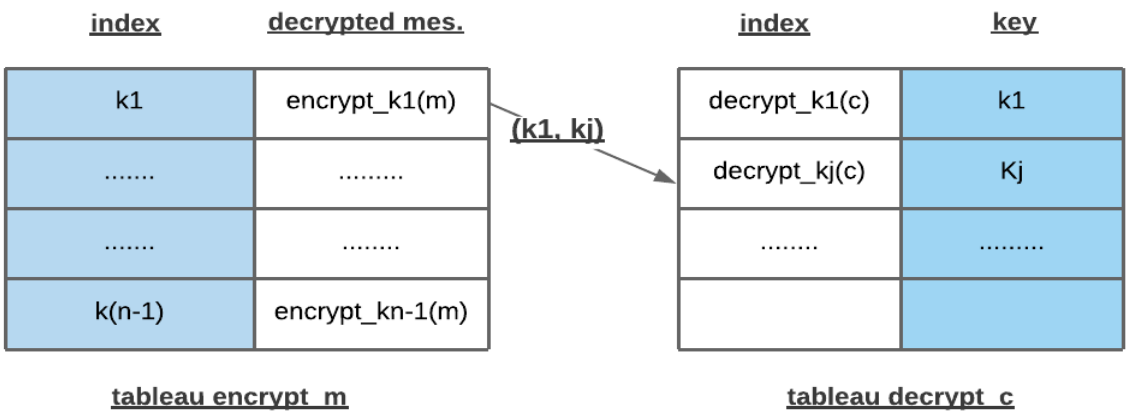
Pour stocker l'ensemble des clés générées on a utilisé la structure de données data_attack composée d'une variable m qui est le message et c qui est 2PRESENTk1,k2(m), un tableau encrypt_m qui contient tout les chiffrés de m en utilisant les clés k comprises [0, 0xffffff] et un tableau decrypt_c qui contient tous les clés qui ont permit de déchiffrés c et les messages déchiffrés sont les index du tableau et les valeurs du tableau sont les clés. Donc le tableau decrypt_c devient une table de hachage dans laquelle chaque entrée correspond à un couple de (déchiffré, clé).

Pour chercher un couple de clés (k1, k2) il suffit de prendre le chiffré c' de m par k1 puis récupérer k2 = decrypt_c[c'] et on incrémente de 1 le nombre de clés trouvées, si 2PRESENTk1,k2(m) = c on arrête la recherche de clés et le couple (k1,k2) est retourné. Cette méthode permet de générer les chiffrés de m et le déchiffrés de c tout en cherchant le bon couple de clé (k1, k2) d'éviter de trier les deux tableaux et d'effectuer une nouvelle boucle pour chercher les couples de clés.

Sur le schéma ci-dessous, la recherche de clés se résume comme suite :
Au cours de la génération des chiffrés et des déchiffrés, on remplit le tableau de decrypt_c par decrypt_c[decrypt_ki(c1)] = ki et le tableau encrypt_m[ki] = encrypt_ki(m) . On cherche ensuite dans decrypt_c la deuxième clé via cette instruction :
kj = decrypt_c[encrypt_c(ki)] où :

- i et j appartiennent [0, 0xffffff – 1].
- ki : est la clé qui a permit de chiffré le message m1
- encrypt_c[ki] : est le chiffré du message m par la clé ki
- kj : est la clé qui a permit de déchiffré c1

Et si kj est différente de -1 et on vérifie si 2PRESENTki,kj(m1) est égale à c1 on renvoi le couple (ki, kj) qui est stocké dans la structure de donnée **key_pairs** sinon on continue à boucler sur i.



rechercher de couple de clé(k1, k2)