DM CRYPTOGRAPHIE

Implémentation du chiffrement :

Afin de chiffrer le message donné par l’utilisateur avec la clé choisie, j’ai choisi de suivre l’algorithme présenté dans le sujet du DM. Ainsi, commencer par convertir en binaire le message et la clé. Ajouter des 0 en bits de poids faible pour atteindre les 80 bits de la clé maitre. Ensuite faire le XOR entre la sous clé et le message et stocker le résultat dans message. Puis on applique la fonction de substitution sur le message suivit par celui de permutation permis à l’aide de tables répertoriant les informations de substitutions et de permutations. Puis enfin s’exécute l’algorithme de cadencement de clé. Ces trois opérations seront alors répétées 10 fois. Une fois les 10 boucles effectuées, il nous suffit d’appliquer la fonction XOR au message avec la dernière sous-clé obtenue. Cette méthode nous permet d’obtenir le message chiffré.

Implémentation du déchiffrement :

Pour ce qui est du déchiffrage, on commence par stocker dans un tableau toutes les sous clés obtenues par 11 boucles de cadencement de clé. Ensuite on applique les étapes du chiffrement à l’envers en veillant à utiliser les sous-clés dans l’ordre inverse d’obtention (de K11 à K1).

Implémentation de l’attaque par le milieu :

Pour l’attaque par le milieu j’ai voulu commencer par créer un fichier qui nous permettrait de stocker l’ensemble des couples de clés (k1, k2) qui permettent de chiffrer m1 en c1.  
Dans un premier temps il faut stocker l’ensemble des combinaisons hexadécimales possibles de longueur 6 dans deux listes, une répertoriant les clés et les messages obtenus par le chiffrement et l’autre stockant les clés et les messages obtenues par le déchiffrement. Une fois l’ensemble des messages intermédiaires obtenus et les deux listes complétées, on parcourt les deux listes obtenues et compare les messages qui y sont stockés, si on trouve des messages intermédiaires identiques alors les clés associées seront inscrites dans le fichier texte ouvert en début de programme. Le fichier texte permet de stocker les résultats de la première attaque et de réduire les possibilités des couples de clés possible.   
Ensuite pour la deuxième attaque permettant de réduire encore le nombre de clés, je voulais récupérer les couples de clés stockées dans le fichier de résultat de la première attaque et ne tester que les clés retenues avec le nouveau couple clair chiffré. Pour ensuite stocker dans un nouveau fichier les quelques couples de clés retenues par la seconde attaque, qui permettent donc le double chiffrement par PRESENT24 des messages m1 et m2.   
Ayant rencontré beaucoup de problèmes dans l’implémentation de cette attaque et de durée de traitement, j’ai donc dû recommencer plusieurs fois cette dernière, et mon attaque n’est pas opérationnelle.