Rapport du DM de Cryptographie
PRESENT24

2.7 A vos machines

1) Le chiffrement fonctionne bien. Il a été testé avec l'ensemble des vecteurs de test.
2) Pour dechiffrer un message, j'ai besoin de considérer les fonctions inverses de S-Box et de la P-Box respectivement le tableau de substitution et le tableau de permutation. Il m'a suffit d'inverser les deux lignes et trier la première ligne.

Tableau de Substitution inversé :

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	5	E	F	8	C	1	2	D	B	4	6	3	0	7	9	A

Tableau de Permutation inversé :

i	0	1	2	3	4	5	6	7	8	9	10	11
P(i)	0	4	8	12	16	20	1	5	9	13	17	21
i	12	13	14	15	16	17	18	19	20	21	22	23
P(i)	2	6	10	14	18	22	3	7	11	15	19	23

3.1 A vos machines

Pour implementer l'attaque par le milieu, l'idée etait la suivante :
J'ai comme donnée deux messages claires, m1 et m2, ainsi que deux messages chiffrés, c1 et c2. La première étape est de remplir un tableau qui contiendra le resultat du chiffrement de m1 avec une clé et cette clé. La deuxième étape est de remplir un tableau qui contiendra le resultat du dechiffrement de c1 avec la même clé et cette même clé. Et je répète cette opération pour l'ensemble des clés possible (codé sur 24 bits donc 2^24 combinaisons possibles soit 16777216 valeurs). Donc j'aurais quelque chose de la sorte pour chaque tour i de boucle:

tab_chiffrement[i][0] ← res_chifrement	tab_dechiffrement[i][0] ← res_dechiffrement
tab_chiffrement[i][1] ← cle$_i$	tab_dechiffrement[i][1] ← cle$_i$

Si j'ai des éléments communs entre les deux tableaux, cela signifie qu'il y a une 'collision' et donc une paire de clé potentiellement valide.

Une fois cette étape faite, il me faut trier l'un des deux tableaux. Pour cela, j'ai choisi de faire un tri de type quicksort. Je l'ai choisi d'une part de sa vitesse d'éxécution, le quicksort étant l'un des tri les plus rapide et efficace, d'autre part par sa facilité d'implémentation.

Après cela, il me suffisait d'effectuer une recherche sur l'ensemble du tableau trié pour chaque case du tableau non trié. J'ai choisi une recherche dichotomique. Cela me permet de ne pas effectuer une recherche sur toute les valeurs une par une mais de faire une recherche sur le milieu du tableau trié. Puis en fonction si la valeur est inférieur ou supérieur a la valeur recherchée, je regarde dans la partie haute ou basse du tableau trié. Cela me fais gagner beaucoup de temps lors de la recherche en evitant de regarder inutilement certaines cases.

Enfin si le programme trouve une collision, je vérifie la validité de cette colision en chiffrant m2 avec la première clé puis avec la deuxieme clé. Je compare avec le message chiffre que je suis censé obtenir. Si c'est les mêmes messages, la clé est valide et je peux l'afficher. Par ailleurs vu que mon tableau est trié sur plusieurs clés sont valides, elle se trouve forcément autour de la collision trouvé dans mon tableau. Pour mes valeurs, j'obtiens trois clé possibles :

Message 1 : 10DC72 Cle 1 : D1D556	Clé trouvé :
Message 2 : 8F60F1 Cle 2 : 8D3B0B	(20507A, 3D4773) ; (61DD54, DF6DC6) ; (104FFF, 6EE8EA)