Pablo Rauzy <pr@up8.edu>
https://pablo.rauzy.name/teaching/is

Introduction à la sécurité

TP 2 : Chiffrement par bloc

Dans ce TP :
- Implémenter un cryptosystème jouet basé sur un réseau de substitution-permutation.
- Comprendre l'utilisation des modes d'opération.

Exercice 0.

Recommandations.

1. Ce TP est à faire en Python.
2. Faites les questions dans l'ordre et pensez à tester votre code.
3. N'hésitez jamais à rajouter de la sortie de debug pour comprendre tout ce qui se passe.

Exercice 1.

Implémentation d'un cryptosystème jouet.

1. Notre cryptosystème, que nous appellerons *ToyCipher*, est un réseau de substitution-permutation à quatre tours, avec une clef de 32 bits et une taille de bloc (message) de 8 bits (un octet).

   La clef (k) est simplement la concaténation des 4 sous-clefs de 8 bits servant de clef de tour dans l'ordre en partant de l'octet de poids faible vers celui de poids fort.

   Les trois premiers tours consistent en l'ajout de la sous-clef du tour à l'état (avec un `xor`), puis au passage à travers une même boîte S 4×4 des deux nibbles qui composent l'octet de l'état, puis au passage à travers une boîte P dont la permutation consiste en une rotation de 2 bits vers la droite (c'est-à-dire que les bits $abcdefgh$ se retrouvent après la permutation en $ghabcdef$).

   Le dernier tour se contente de l'ajout de la sous-clef sans faire les boîtes S et la boîte P.

   → Pourquoi le dernier tour est-il différent des trois premiers ?

2. Pour la boîte S, on peut par exemple réutiliser celle de PRESENT qu'on a vue en cours :

n	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(n)$	c	5	6	b	9	0	a	d	3	e	f	8	4	7	1	2

   → Implémentez le calcul d'un tour dans une fonction `round(state, subkey)` dont chacun des deux arguments fait un octet (l'état et la sous-clef de tour).

3. → Implémentez l'algorithme de chiffrement dans une fonction `enc`.
4. Pour déchiffrer, il faut faire les tours "à l'envers".

   → Implémentez la fonction inverse de `round` dans une fonction `back_round`.

5. → Implémentez l'algorithme de déchiffrement dans une fonction `dec`.
6. Pensez toujours à tester votre code ! Notamment `m == dec(enc(m, k), k)` doit être vrai quelque soit `m` un message et `k` une clef.

   → Vérifiez par exemple que le chiffrement de la valeur 42 avec la clef `0x13acab12` donne 1. Et que le déchiffrement de 1 donne bien 42 (avec la même clef, évidemment).

Exercice 2.

Chiffrer plus qu'un bloc.

1. Maintenant qu'on sait chiffrer un octet, on veut pouvoir chiffrer n'importe quel fichier.

   → Écrivez une fonction `enc_file` qui prend le nom d'un fichier et une clef en argument, qui ouvre ce fichier en mode binaire et le chiffre octet par octet en utilisant `enc` et écrit le résultat dans le fichier du même nom avec le suffixe `.enc` ajouté.

2. Évidemment, on veut pouvoir déchiffrer.

   → Écrivez la fonction `dec_file` (écrivez le résultat dans le fichier du même nom avec le suffixe `.dec` pour éviter d'écraser vos fichiers existants).

3. → Vérifiez que vos fonctions marchent correctement. Amusez-vous à envoyer un fichier chiffrer et la clef secrète à l'un·e de vos camarades pour qu'iel le déchiffre avec son code, en vice versa, pour vérifier que vos implémentations sont compatibles.

Exercice 3.

Jouer avec les modes d'opération.

1. Créez un fichier "test.txt" avec dedans le texte "salut salut salut". Chiffrez le avec la clef `0xabc00def`, et observez le résultat.

   → Que remarquez-vous ?
2. → Proposez un type d'attaque contre ce chiffrement.
3. Pour éviter ce genre de problème, on utilise les *modes d'opération*.

   → En vous basant sur les schémas vus cours[1], implémentez les paires de fonctions suivantes (le vecteur d'initialisation peut être généré aléatoirement mais il doit être conservé pour le déchiffrement) :
   - `enc_file_cbc` et `dec_file_cbc`
   - `enc_file_cfb` et `dec_file_cfb`
   - `enc_file_ofb` et `dec_file_ofb`
4. Testez votre code, et vérifiez que le soucis de la première question de cet exercice est corrigé.

   → Quel est l'avantage de la génération aléatoire du vecteur d'initialisation ?
5. → Que remarquez vous comme avantage additionnel des modes CFB et OFB ?

1. Ils sont d'ailleurs tirés directement des pages Wikipédia correspondantes, que vous êtes également encouragé·es à consulter.