Faculté des Sciences & Techniques — Université de Limoges

Licence 3ème année

Initiation Sécurité

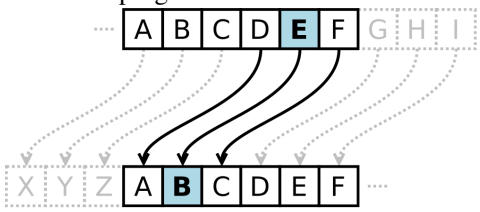
TP n°1

Chiffrement/Déchiffrement et Cryptanalyse

Chiffrement & Cryptanalyse

1 – Écrire un programme qui compte le nombre d'occurences de chaque lettre de l'alphabet dans un texte et tri les caractères suivant l'ordre décroissant de ce nombre d'occurences.

2 – Écrire un programme réalisant le chiffrement de César après saisie du décalage :




```
Plain:    ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher:   XYZABCDEFGHIJKLMNOPQRSTUVW

Plaintext:  the quick brown fox jumps over the lazy
dog
Ciphertext: QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV
ALD
```

3 – Écrire un programme d'**aide à la cryptanalyse** du chiffrement César à partir de l'analyse fréquentielle des caractères présents dans le chiffré et par comparaison aux fréquences les plus élevées des caractères de la langue française.
Vous déchiffrerez le message suivant :

```
BULML TTLWH ZZHKB ULTHP UMHZA BLBZL ZVBSL CHUAI HSHUJ HUASL MLZAV ULASV BYSLA
```

4 – Écrire un programme de **déchiffrement du code César** par la méthode « **brute-force** ».

5 – Écrire un programme réalisant le **chiffrement** par la méthode la « **Scytale** » grecque :

```
_____________________________
|   |   |   |   |   |   |   |
| H | E | L | P | M |   |   |
|_E | I | A | M | U |__ |__ |
| N | D | E | R | A |   |   |
| T | T | A | C | K |   |   |
|   |   |   |   |   |   |   |
_____________________________
```

Le texte :

```
Help me I am under attack
```

devient :

```
HENTEIDTLAEAPMRCMUAK
```

6 – Écrire un programme permettant de **déterminer si** un message est chiffré suivant la méthode de la Scytale **ou** par le chiffrement de César.

7 – Écrire un programme de chiffrement utilisant la technique de l'OTP, « *One Time Password* » :

- ⋆ les deux chaînes, texte en clair et texte secret, sont données de même longueur (le secret est appelé clé ou « *pad* ») ;
- ⋆ chaque caractère du texte chiffré est le caractère résultat de l'opérateur `xor` appliqué sur les deux caractères correspondant du texte en clair et du secret.

Vous afficherez le résultat en notation hexadécimale.

Exemple :

notation	**xor**	**résultat**
hexadécimale	`41⊕5A`	`1B`
binaire	`1000001⊕1011010`	`11011`
décimale	`65⊕90`	`27`

8 – Utilisation d'un GCL, « générateur à congruence linéaire », est un générateur de nombres pseudo-aléatoires basé sur des congruences et une fonction affine :

$$X_{n+1} = (aX_n + c) \bmod m, \text{ où le terme } X_0 \text{ est appelé « } \textit{seed} \text{ ».}$$

- ▷ *Pour chaque seed, on obtient une nouvelle suite de nombres.*
- ▷ *Les nombres de la suite ont l'apparence de l'aléas.*
- ▷ *Cette suite est plus ou moins grande : tout nouveau nombre étant basé sur le précédent, si un nombre apparait une deuxième fois dans la suite alors la suite se répète entièrement à partir de ce nombre.*
- ▷ *Le nombre de valeurs de la suite étant fini, il dépend de m, la suite se répètera forcément.*
- ▷ *En utilisant, un même seed, on obtient la même séquence de nombres (d'où le nom de « pseudo »-aléatoire).*

Certaines valeurs bien choisies pour m, a et c permettent d'obtenir des séquences assez longues.

On utilisera les valeurs trouvées par Donald Knuth :

m	a	c
2^{64}	6364136223846793005	1442695040888963407

Questions :

a. Quelle est la taille maximale des valeurs données par le générateur avec les paramètres de D. Knuth ?

b. L'opérateur binaire « xor » permet de combiner une séquence binaire S_a avec une séquence binaire S_b en inversant les bits de S_a de même rang que les bits à 1 de S_b, ce qui donne la séquence S_r.
Vérifiez que si on combine avec un xor S_r avec S_b, on obtient bien S_a.

c. Écrire un programme de « chiffrement » permettant de combiner un message M avec une séquence de valeurs obtenues à l'aide du générateur à congruence linéaire pour un seed donné.
Vous vérifierez que l'opération de déchiffrement est possible en utilisant le même seed.