Perdu Nicolas
Rottier Béatrice

# TP CRYPTOGRAPHIE

ISITV Télécom – 2003 / 2004

# INTRODUCTION

## A. Le sujet du TP

Mettre en place, en C, l'algorithme suivant :

1. saisie du texte en clair
2. saisie de la clé de chiffrage
3. opération de chiffrement puis affichage du cryptogramme
4. saisie de la clé de déchiffrage
5. opération de déchiffrement puis affichage du message décrypté

Pour ces trois cryptosystèmes

- CS de César
- CS de Vigenère
- CS "Che Gevara"

Remarques :

- Pour le message en clair, on ne travaillera qu'avec des minuscules, sans valeurs numériques, et on ne chiffrera pas les espaces.

- Avant de chiffrer, on ôtera le code ASCII de 'a' (pour que a⇔0, b⇔1…). Avant d'afficher les caractères, on le ré-ajoutera.

- Pour la saisie, le cryptogramme,… on utilisera des tableaux de 256 éléments (par exemple `char clair[256];`).

- Les clés seront toutes sous forme numérique

## B. Contenu du rapport

Pour chaque cryptosystème nous détaillerons :

- Ses caractéristiques
- Sa mise en œuvre (transcodage, clé, chiffrement, déchiffrement)
- Son implémentation

Pour tester les programmes, il suffit de lancer le « crypto.exe » qui propose d'utiliser l'un ou l'autre des algorithmes via un menu.

# Le cryptosystème de César

## 1. Caractéristiques

- Mono-alphabétique
- Les lettres gardent la même position
- Seules 26 clés sont possibles (puisque la clé est modulo 26)
- Le cryptogramme est une suite de caractères

N'empêche pas les attaques fréquentielles
Le nombre réduit de clés (26) permet de toutes les tester

## 2. Mise en oeuvre

Transcodage

'a' correspond à 0
'b' correspond à 1
:
'z' correspond à 26.

Clé

La clé K est comprise entre 0 et 26. On aura donc K = K modulo(26).

Chiffrement

Cryptogramme(caractère) = [ caractère + K ] modulo(26)

*Le modulo(26) permettant de conserver une valeur qui correspond à un caractère soit comprise entre 0 et 25.*

Déchiffrement

Décrypté(caractère) = [ caractère – K + 26] modulo(26)

*Le +26 permet de s'affranchir de problèmes posés par des nombres négatifs et ne modifie pas le résultat qui est modulo(26).*

Exemple

La clé est de 3 :
chiffrement('a') = chiffrement(0) = (0+3) modulo(26) = 3 ⇔ 'd'

Cryptage :

| **Clair** | **A** | **T** | **T** | **A** | **Q** | **U** | **E** | **Z** | | **A** | **S** | **T** | **E** | **R** | **I** | **X** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Transcodé | **0** | **19** | **19** | **0** | **16** | **20** | **4** | **25** | | **0** | **18** | **19** | **4** | **17** | **8** | **23** |
| Chiffré | 3 | 22 | 22 | 3 | 19 | 23 | 7 | 28 | | 3 | 21 | 22 | 7 | 20 | 11 | 26 |
| Modulo 26 | **3** | **22** | **22** | **3** | **19** | **23** | **7** | **2** | | **3** | **21** | **22** | **7** | **20** | **11** | **0** |
| **Chiffré** | **D** | **W** | **W** | **D** | **T** | **X** | **H** | **C** | | **D** | **V** | **W** | **H** | **U** | **L** | **A** |

Décryptage :

| **Chiffré** | **D** | **W** | **W** | **D** | **T** | **X** | **H** | **C** | | **D** | **V** | **W** | **H** | **U** | **L** | **A** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Transcodé | **3** | **22** | **22** | **3** | **19** | **23** | **7** | **2** | | **3** | **21** | **22** | **7** | **20** | **11** | **0** |
| Déchiffré | 0 | 19 | 19 | 0 | 16 | 20 | 4 | 25 | | 0 | 18 | 19 | 4 | 17 | 8 | -3 |
| +26, mod(26) | **0** | **19** | **19** | **0** | **16** | **20** | **4** | **25** | | **0** | **18** | **19** | **4** | **17** | **8** | **23** |
| **Déchiffré** | **A** | **T** | **T** | **A** | **Q** | **U** | **E** | **Z** | | **A** | **S** | **T** | **E** | **R** | **I** | **X** |

## 3. Implémentation

Nous avons mis en place l'algorithme décrit dans la partie théorique. Hormis le problème expliqué en conclusion (gestion du tampon de saisie), tout se passe comme attendu.

Pour voir l'exécution du programme, lancer crypto.exe puis saisir le message à crypter (en minuscules), et entrer 1, pour «cryptosystème de César ».

# Le cryptosystème de Vigenère

## 1. Caractéristiques

- Poly-alphabétique
- Les lettres gardent la même position
- Le cryptogramme est une suite de caractères
- Une même lettre peut être codée de différentes façons
- Un cryptogramme ne correspond pas toujours à la même lettre claire
- Il y a $26^L$ clés possibles (L = longueur du message)

Empêche les attaques fréquentielles

Il faudrait tester $\sum_{i=1}^{T} 26^i$ clés (avec T la longueur du texte)

Ce chiffrement est « parfait » si la longueur de la clé est aussi longue que le message à crypter (démontré par Shannon).

## 2. Mise en oeuvre

Le principe est identique à celui du cryptosystème de César, sauf que la clé est sur plusieurs caractères.

Exemple

La clé est 3 1 4 (longueur=3).

chiffrement('abac') = chiffrement(0,1,0,2)
= (0+3,1+1,0+4,2+3) modulo(26)
= 3,2,4,5 ⇔ 'dcef'

Cryptage :

| **Clair** | **A** | **T** | **T** | **A** | **Q** | **U** | **E** | **Z** | | **A** | **S** | **T** | **E** | **R** | **I** | **X** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Transcodé | 0 | 19 | 19 | 0 | 16 | 20 | 4 | 25 | | 0 | 18 | 19 | 4 | 17 | 8 | 23 |
| **Clé** | **3** | **1** | **4** | **3** | **1** | **4** | **3** | **1** | | **4** | **3** | **1** | **4** | **3** | **1** | **4** |
| Chiffré | 3 | 20 | 23 | 3 | 17 | 24 | 7 | 26 | | 4 | 21 | 20 | 8 | 20 | 9 | 27 |
| Modulo 26 | 3 | 20 | 23 | 3 | 17 | 24 | 7 | 0 | | 4 | 21 | 20 | 8 | 20 | 9 | 1 |
| **Chiffré** | **D** | **U** | **X** | **D** | **R** | **Y** | **H** | **A** | | **E** | **V** | **U** | **I** | **U** | **J** | **B** |

Décryptage :

| **Chiffré** | **D** | **U** | **X** | **D** | **R** | **Y** | **H** | **A** | | **E** | **V** | **U** | **I** | **U** | **J** | **B** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Transcodé | 3 | 20 | 23 | 3 | 17 | 24 | 7 | 0 | | 4 | 21 | 20 | 8 | 20 | 9 | 1 |
| **Clé** | **3** | **1** | **4** | **3** | **1** | **4** | **3** | **1** | | **4** | **3** | **1** | **4** | **3** | **1** | **4** |
| Déchiffré | 0 | 19 | 19 | 0 | 16 | 20 | 4 | -1 | | 0 | 18 | 19 | 4 | 17 | 8 | -3 |
| +26, mod(26) | 0 | 19 | 19 | 0 | 16 | 20 | 4 | 25 | | 0 | 18 | 19 | 4 | 17 | 8 | 23 |
| **Déchiffré** | **A** | **T** | **T** | **A** | **Q** | **U** | **E** | **Z** | | **A** | **S** | **T** | **E** | **R** | **I** | **X** |

## 3. Implémentation

Nous avons mis en place l'algorithme décrit dans la partie théorique. Hormis le problème expliqué en conclusion (gestion du tampon de saisie), tout se passe comme attendu.

Pour voir l'exécution du programme, lancer crypto.exe puis saisir le message à crypter (en minuscules), et entrer 2, pour «cryptosystème de Vigenère ».

# Le cryptosystème de Che Gevara

## 1. Caractéristiques

On utilise une table de transcodage.
Les caractéristiques sont identiques à celles du cryptosystème de Vigenère (solidité, impossibilité d'attaque fréquentielle…).
Le cryptogramme est une suite de nombres.

Ce cryptosystème a l'avantage d'être très simple coté cryptage, mais plus délicat côté décryptage. 5on a besoin de connaître la clé ET la table de transcodage).

## 2. Mise en application

Transcodage

On utilise une table de transcodage qui affecte à chaque lettre une valeur comprise entre 0 et 99, sans règle particulière. Cette table est connue de la personne qui crypte et du destinataire du message.

Clé

Comme pour Vigenère, elle est sur plusieurs caractères (la longueur idéale étant celle du message à chiffrer), mais chaque élément prend une valeur comprise entre 0 et 99 au lieu d'être compris entre 0 et 25.

Chiffrement (avec K l'élément de la clé qui correspond au caractère)

Cryptogramme(caractère) = [ caractère + K ] modulo(100)

Déchiffrement

Décrypté(caractère) = [ caractère – K ] modulo(100)

Exemple

Table de transcodage

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **A** | 6 | **B** | 38 | **C** | 32 | **D** | 4 | **E** | 8 | **F** | 30 | **G** | 36 |
| **H** | 34 | **I** | 39 | **J** | 31 | **K** | 78 | **L** | 72 | **M** | 70 | **N** | 76 |
| **O** | 9 | **P** | 79 | **Q** | 71 | **R** | 58 | **S** | 2 | **T** | 0 | **U** | 52 |
| **V** | 50 | **W** | 56 | **X** | 54 | **Y** | 1 | **Z** | 59 | | | | |

La clé est de 10 60 12 4.
chiffrement('aba') = chiffrement(0,1,0)
= (0+10,1+60,0+12) modulo(100)
= 10 61 12

Cryptage :

| **Clair** | **A** | **T** | **T** | **A** | **Q** | **U** | **E** | **Z** | | **A** | **S** | **T** | **E** | **R** | **I** | **X** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Transcodé | 6 | 0 | 0 | 6 | 71 | 52 | 8 | 59 | | 6 | 2 | 0 | 8 | 58 | 39 | 54 |
| **Clé** | **10** | **60** | **12** | **4** | **10** | **60** | **12** | **4** | | **10** | **60** | **12** | **4** | **10** | **60** | **12** |
| Chiffré | 16 | 60 | 12 | 10 | 81 | 112 | 20 | 63 | | 16 | 62 | 12 | 12 | 68 | 99 | 66 |
| **Chiffré mod(100)** | **16** | **60** | **12** | **10** | **81** | **12** | **20** | **63** | | **16** | **62** | **12** | **12** | **68** | **99** | **66** |

Décryptage :

| **Chiffré** | **16** | **60** | **12** | **10** | **81** | **12** | **20** | **63** | | **16** | **62** | **12** | **12** | **68** | **99** | **66** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Clé** | **10** | **60** | **12** | **4** | **10** | **60** | **12** | **4** | | **10** | **60** | **12** | **4** | **10** | **60** | **12** |
| Déchiffré | 6 | 0 | 0 | 6 | 71 | -48 | 8 | 59 | | 6 | 2 | 0 | 8 | 58 | 39 | 54 |
| +100 mod(100) | 6 | 0 | 0 | 6 | 71 | 52 | 8 | 59 | | 6 | 2 | 0 | 8 | 58 | 39 | 54 |
| **Transcodé (inverse)** | **A** | **T** | **T** | **A** | **Q** | **U** | **E** | **Z** | | **A** | **S** | **T** | **E** | **R** | **I** | **X** |

## 3. Implémentation

Nous avons mis en place l'algorithme décrit dans la partie théorique. Hormis le problème expliqué en conclusion (gestion du tampon de saisie), tout se passe comme attendu.

Pour voir l'exécution du programme, lancer crypto.exe puis saisir le message à crypter (en minuscules), et entrer 3, pour «cryptosystème de Che Gevara ».

# CONCLUSION

Le programme que nous avons mis en place fonctionne correctement : le message clair est bien crypté, le décryptage est bon si l'on rentre la même clé que pour crypter, et donne un résultat « incohérent » si la clé est différente.

Du côté du programme, il persiste un « bug » dû à l'utilisation du langage C. En effet, en C, il est particulièrement compliqué de gérer les tampons de saisie. Ainsi, si l'on saisit plus de valeurs que ce que l'on attend (par exemple 2 valeurs pour la longueur de la clé), les valeurs supplémentaires sont gardées en mémoire et prises en compte pour les saisies suivantes.

En fait, ce problème n'est pas gênant quand on utilise le programme en faisant attention à ne rentrer que le nombre attendu de valeurs. Il le devient si le programme est utilisé par quelqu'un d'étourdi…

Ce TP nous a permis de travailler sur plusieurs cryptosystèmes et de les mettre en place. Il serait intéressant maintenant de travailler sur des algorithmes permettant de cryptanalyser les cryptogrammes obtenus, et de comparer leur performances.