

# Cryptographie symétrique: César, Vigenere, réseaux de Feistel

Cours de sécurité: DUT S4

2013-2014

## 1 Chiffrement de César

### 1.1 Cryptographie

1. Observez puis compilez le programme `cesar.cpp`, qui implémente l'algorithme de chiffrement/déchiffrement de César vu en cours
  - Entrée: un fichier et le décalage (une lettre minuscule:  $'a' = 0 \dots 'z' = 25$ )
  - Sortie: le fichier chiffré
  - Notes:
    - On ne prendra en compte (jusqu'à la fin de ce tp) que les fichiers contenant uniquement des lettres minuscules (pas d'accents, pas d'espace, pas de ponctuation). Pourquoi ?
    - On pose  $'a' = 0 \dots 'z' = 25$
2. Testez-le sur des fichiers de votre crû, chiffrez puis déchiffrez.
3. Si un programme a été chiffré avec la clé 'e', quelle est la clé de déchiffrement ?
4. Que remarquez vous pour un chiffrement/déchiffrement avec pour clé 'n' (i.e un décalage de 13) ?

### 1.2 Cryptanalyse

On se propose maintenant de casser le chiffrement de césar via une attaque de type texte chiffré connu.

1. Comment faire ?
2. Créez un programme `freq.cpp` qui calcule la fréquence d'apparition des lettres d'un fichier texte (vous pourrez vous aider du squelette `lecture_fichier.cpp`)
3. Retrouvez le texte clair à partir du fichier chiffré `cesar1_chiffre.txt`
  - La fréquence des lettres de l'alphabet est donné en annexe

## 2 Vigenere

### 2.1 Cryptographie

1. Observez puis compilez le programme `vigenere.cpp` qui implémente le chiffrement de vigenere vu en cours.
  - Entrée: un fichier et la clé (un mot de lettres minuscules)
  - Sortie: le fichier chiffré
  - Notes:
    - On pose les même conditions que pour le chiffrement de Cesar
2. Testez-le sur des fichiers de votre cru. Chiffrez, puis déchiffrez.
3. A quel autre algorithme est-il équivalent pour une longueur de clé de 1 ?

### 2.2 Cryptanalyse

On se propose maintenant de casser le chiffrement de vigenere via une attaque de type texte chiffré connu.

1. Déterminez la longueur de la clé:
  - Via un programme `ic` qui va implémenter l'attaque par calcul de l'indice de coïncidence
2. Créez un programme `decoup` :
  - Entrées: un fichier `f`, un entier `len`
  - Sorties:  $len$  fichiers  $f_0 \dots f_{len-1}$  tels que  $f_i$  contient toutes les lettres de  $f$  dont la position dans le texte modulo  $len$  est égale à  $i$  (découpage en "*colonnes*" vu en cours).
3. Retrouvez le texte clair à partir du fichier chiffré `vigenere1_chiffre.txt`
4. Notes:
  - L'indice de coïncidence d'un texte français est d'environ 0.08
5. Cassez le chiffrement du fichier `vigenere2_chiffre.txt` et retrouvez le texte en clair
  - Attention, celui-là est en anglais
6. Essayez d'automatiser la découverte de la clé le plus possible

## 3 Réseaux de Feistel

1. Implémentez le réseau de Feistel suivant:
  - Longueur de bloc de 64bits
  - Longueur de la clé de 32 bits
  - 12 rondes
  - La fonction  $f$  est laissée à votre choix (elle doit être inversible !)

- Le générateur de clé  $g$  est également laissé à votre choix
  - Si vous n'avez pas d'idée, vous pouvez prendre  $k_i = k + i * x$  où  $x$  est un entier quelconque
- 2. Testez le chiffrement et le déchiffrement sur des fichiers de votre crû
- 3. Observez:
  - La confusion du chiffrement
  - La diffusion du chiffrement

## 4 Annexe

Lettre	Fréquence %	Lettre	Fréquence %
<b>A</b>	<b>9.42</b>	<b>N</b>	<b>7.15</b>
<b>B</b>	<b>1.02</b>	<b>O</b>	<b>5.14</b>
<b>C</b>	<b>2.64</b>	<b>P</b>	<b>2.86</b>
<b>D</b>	<b>3.39</b>	<b>Q</b>	<b>1.06</b>
<b>E</b>	<b>15.87</b>	<b>R</b>	<b>6.46</b>
<b>F</b>	<b>0.95</b>	<b>S</b>	<b>7.90</b>
<b>G</b>	<b>1.04</b>	<b>T</b>	<b>7.26</b>
<b>H</b>	<b>0.77</b>	<b>U</b>	<b>6.24</b>
<b>I</b>	<b>8.41</b>	<b>V</b>	<b>2.15</b>
<b>J</b>	<b>0.89</b>	<b>W</b>	<b>0.00</b>
<b>K</b>	<b>0.00</b>	<b>X</b>	<b>0.30</b>
<b>L</b>	<b>5.34</b>	<b>Y</b>	<b>0.24</b>
<b>M</b>	<b>3.24</b>	<b>Z</b>	<b>0.32</b>

Figure 1: Fréquence d'apparition des lettres de l'alphabet dans un texte français