Cryptographie et algorithmique

F.Gaudon

1er novembre 2010

Table des matières

1 Avant de commencer

Ce document contient une liste d'algorithmes en rapport avec la cryptographie pour le lycée et les programmes correspondants.

La cryptographie abordée ici porte essentiellement sur les méthodes de chiffrement de textes par substitution (codage de César, Vigenère) à l'exception notable de l'algorithme RSA.

Le langage de programation Python a été choisi car il est considéré par les développeurs professionels comme l'un des plus appropriés pour l'apprentissage de la programmation. Le langage de programmation du logiciel XCas a été choisi pour sa structure francisée et néanmoins très proche de celle des langages de programmation usuels (Python, C entre autres), ce qui en fait un logiciel particulièrement adapté à un public francophone.

Ce document suppose une connaissance des instructions fondamentales (entrées, sorties, boucles, conditions) du langage Python ainsi que de celles du logiciel XCas. Il n'est fait usage dans les programmes que de notions pouvant être abordées au niveau du lycée dans le cadre de l'apprentissage de l'algorithmique. En particulier, il n'est pas fait usage de la notion de fonction (informatique...). Pour le langage Python, afin d'utiliser des caractères accentués dans les programmes, on n'oubliera pas de préciser l'encodage de caractères utilisé en plaçant au début de chaque programme la ligne suivante sous Windows

```
#-*- coding:Latin-1 -*-
```

et sous Linux ubuntu

```
#-*- coding:Utf-8 -*-
```

2 Préformattage d'un texte pour analyse

2.1 Élimination de la ponctuation et des espaces dans un texte

Entrées :
texte : chaîne de caractère.
Début traitement
    nbcar prend la valeur de longueur(*texte*) ;
    alphabet="abcdefghijklmnopqrstuvwxyz" ;
    texteformate prend la valeur $\emptyset$;
    pour *k* ***allant de*** *1* ***à*** *nbcar* **faire**
        j prend la valeur 0 ;
        trouve prend la valeur 0 ;
        tant que *j<longeur(alphabet) et trouve* $= 0$ **faire**
            si $texte[k] = alphabet[j]$ **alors**
                $trouve = 1$;
            fin
            si $trouve = 0$ **alors**
                j prend la valeur de $j + 1$;
            fin
        fin
        si $trouve = 1$ **alors**
            $texteformate = texteformate + texte[k]$;
        fin
    fin
Fin
Sorties : *texteformate*

Programmes correspondants :

XCas :

```
texte:="Ceci n'est qu'un exemple.";
alphabet:="abcdefghijklmmnopqrstuvwxyz...
    ...ABCDEFGHIJKLMNOPQRSTUVWXYZ";
nbcar:=length(texte);
afficher("Texte : "+texte);
afficher("Nombre de caractères :"+nbcar);
texteformate:="";
pour k de 0 jusque nbcar-1 faire
  j:=0;
  trouve:=0;
  tantque (j<length(alphabet)) et ...
    ...(trouve==0) faire
    si texte[k]==alphabet[j] alors
      trouve:=1;
    fsi;
    si trouve==0 alors j:=j+1;fsi;
  ftantque;
  si trouve==1 alors
    texteformate:=texteformate+texte[k];
  fsi;
fpour;
afficher(texteformate);
```

Python :

```
texte="Ceci n'est qu'un exemple"
alphabet="abcdefghijklmnopqrstuvwxyz...
   ...ABCDEFGHIJKLMNOPQRSTUVWXYZ"
nbcar=len(texte)
print "Texte : ",texte
print "Nombre de caractères :",nbcar
texteformate=""
for k in range(nbcar):
    j=0
    trouve=0
    while (j<len(alphabet)) ...
  ...and (trouve==0):
if texte[k]==alphabet[j]:
    trouve=1
if trouve==0:
    j=j+1
    if trouve==1:
texteformate=texteformate+texte[k]
print texteformate
```

Remarques :

- Les ... apparaissant ci-dessus ne doivent pas apparaître dans les programmes, ils signalent juste que les deux lignes ne doivent en faire qu'une.
- Le texte à traiter est ici entré sous forme d'une chaîne de caractère. En cas de « long texte » à traiter. En python, on pourra le stocker dans un fichier à part et utiliser les instructions suivantes :
  `fichier=open('icivotrenomdefichier.txt','r')`
  `texte=fichier.read()`
  `fichier.close()`

2.2 Formatage du texte en lettres minuscules

L'algorithme suivant met un texte préalablement débarassé de tout signe de ponctuation en lettres minuscules.

Entrées :
texte : chaîne de caractère.
Début traitement
  nbcar prend la valeur de longueur(*texte*) ;
  texteformate prend la valeur $\emptyset$;
  pour *k* ***allant de*** *1* ***à*** *nbcar* **faire**
    si *ord(texte[k])<97* **alors**
      texteformate=texteformate+chr(ord(*texte*[*k*])+32) ;
    sinon
      texteformate prend la valeur de *texteformate* + *texte*[*k*] ;
    fin
  fin
Fin
Sorties : *texteformate*

Programmes correspondants :

XCas :

```
texte:="CecinestquUNexemple";
nbcar:=length(texte);
afficher("Texte : "+texte);
afficher("Nombre de caractères :"+nbcar);
texteformate:="";
pour k de 0 jusque nbcar-1 faire
  si ord(texte[k])<97 alors
    texteformate:=texteformate+...
    ...char(ord(texte[k])+32)
  else
    texteformate:=texteformate+texte[k];
  fsi;
fpour;
afficher(texteformate);
```

Python :

```
texte="CecinestquUNexemple"
nbcar=len(texte)
print "Texte : ",texte
print "Nombre de caractères :",nbcar
texteformate=""
for k in range(nbcar):
    if ord(texte[k])<97:
texteformate=texteformate+...
      ...chr(ord(texte[k])+32)
    else:
texteformate=texteformate+texte[k]
print texteformate
```

3 Analyse de fréquences

L'analyse de fréquences d'un texte permet, lorsque l'on a simplement remplacé une lettre d'un texte par une autre (méthode de substitution simple), de reconnaître les lettres les plus fréquentes du texte, le "e" en français par exemple.

3.1 Analyse de fréquences d'une lettre de l'alphabet

L'algorithme suivant affiche la fréquence d'apparition de la lettre demandée par l'utilisateur dans un texte préalablement formatté (caractères mis en minuscules, espaces et signes de ponctuation supprimés) donné :

Entrées :
lettre : caractère ;
texte : chaîne de caractère.
Début traitement
  nbcar prend la valeur de longueur(*texte*) ;
  compteur prend la valeur 0 ;
  pour *k* ***allant de*** *1* ***à*** *nbcar* **faire**
    si *texte*[*k*] = *lettre* **alors**
      compteur prend la valeur de *compteur*+1 ;
    fin
    freq prend la valeur $\frac{compteur}{nbcar}$
  fin
Fin
Sorties : *freq*

Programmes correspondants :

XCas :

```
texte:="cecinestquunexemple";
nbcar:=length(texte);
afficher("Texte : "+texte);
afficher("Nombre de caractères :"+nbcar);
saisir(lettre);
compteur:=0;
pour k de 0 jusque nbcar-1 faire
  si texte[k]==lettre alors
  compteur:=compteur+1;fsi;
fpour;
freq:=approx(compteur/nbcar);
afficher("Fréquence :"+freq);
```

Python :

```
texte="cecinestquunexemple"
nbcar=len(texte)
print "Texte : ",texte
print "Nombre de caractères :",nbcar
lettre=raw_input("Lettre : ")
compteur=0.0
for k in range(nbcar):
    if texte[k]==lettre:
compteur=compteur+1
freq=compteur/nbcar
print "Fréquence :",freq
```

Remarque :
Pour le programme sous XCAS, on entrera la lettre voulue entre parenthèses afin que le logiciel traite la lettre entrée comme un caractère.

3.2 Analyse de fréquence de l'ensemble des lettres de l'alphabet

Même problème que le précédent mais on ajoute une boucle pour calculer la fréquence de chacune des lettres de l'alphabet.

Entrées :
texte : chaîne de caractères.
Début traitement
  nbcar prend la valeur de longueur(*texte*) ;
  alpahbet prend la valeur "abcdefghijklmnopqrstuvwxyz"
  pour *j* ***allant de*** *1* ***à*** *26* **faire**
    compteur prend la valeur 0 ;
    pour *k* ***allant de*** *1* ***à*** *nbcar* **faire**
      si $texte[k] = alphabet[j]$ **alors**
        compteur prend la valeur de *compteur*+1 ;
      fin
      freq prend la valeur $\frac{compteur}{nbcar}$
      Afficher *freq*
    fin
  fin
Fin

Programmes correspondants :

XCas :

```
texte:="cecinestquunexemple";
nbcar:=length(texte);
afficher("Texte : "+texte);
afficher("Nombre de caractères :"+nbcar);
alphabet:="abcdefghijklmnopqrstuvwxyz";
pour j de 0 jusque 25 faire
  compteur:=0;
  pour k de 0 jusque nbcar-1 faire
    si texte[k]==alphabet[j] alors
    compteur:=compteur+1;fsi;
  fpour;
  freq:=approx(compteur/nbcar);
  afficher(alphabet[j]+" : "+freq);
fpour;
```

Python :

```
texte="cecinestquunexemple"
nbcar=len(texte)
print "Texte : ",texte
print "Nombre de caractères :",nbcar
alphabet="abcdefghijklmnopqrstuvwxyz"
for lettre in alphabet:
  compteur=0.0
  for k in range(nbcar):
      if texte[k]==lettre:
  compteur=compteur+1
  freq=compteur/nbcar
  print lettre," : ",freq
```

Remarque :

Dans le programme pour Xcas, j est l'indice de la lettre dans l'alphabet. On aurait pu procéder de même en langage Python mais celui-ci permet aussi d'utiliser la variable *lettre* comme un caractère de l'alphabet ce qui est plus pratique.

4 Substitution d'une lettre par une autre

L'algorithme suivant est à la base des méthodes de substitution simple : il substitue une lettre donnée par une autre lettre donnée dans un texte donné.

Entrées :
*lettre*1, *lettre*2 : caractères ;
texte : chaîne de caractère.
Début traitement
  nbcar prend la valeur de longueur(*texte*) ;
  pour *k* ***allant de*** *1* ***à*** *nbcar* **faire**
    si *texte*[*k*] = *lettre*1 **alors**
      texte[*k*] = *lettre*2 ;
    fin
  fin
Fin
Sorties : *texte*

Programmes correspondants :

XCas :

```
texte:="abcdefgh";
nbcar:=length(texte);
afficher(texte);
saisir("Lettre à substituer : ",lettre1);
saisir("Lettre de substitution:",lettre2);
pour k de 0 jusque nbcar-1 faire
  si texte[k]==lettre1
  alors texte[k]:=lettre2;fsi;
fpour;
afficher(texte);
```

Python :

```
texte="abcdefgh"
lettre1=raw_input("Lettre à substituer :")
lettre2=raw_input("Lettre de substitution:")
nbcar=len(texte)
print "Texte à coder : "+texte
textecode=""
for k in range(nbcar):
  if texte[k]==lettre1:
      textecode=textecode+lettre2
  else:
      textecode=textecode+texte[k]
print "Texte codé : "+textecode
```

Remarques :

- Dans le programme sous Python, on a utilisé une deuxième chaîne de caractères "textecode" car l'instruction `lettre[k]=lettre2` n'est pas acceptée et provoque une erreur. On aurait cependant pu utilisé l'instruction `texte=texte.replace(lettre1,lettre2)` par contre à la place de l'ensemble de la boucle `for`.
- Dans le programme XCas, entrer les lettres entre guillemets afin qu'elles soient reconnues comme du texte.

5 Codage affine

L'algorithme suivant est un exemple de chiffre de substitution : il chiffre un texte préformatté en utilisant un cryptage affine donné par la fonction $f \mapsto mx + p$ où m et p sont entrés par l'utilisateur. Attention, selon les valeurs choisis de m et p, on obtient un cryptage qui n'est pas nécessairement efficace : en effet, pour certaines valeurs de m et p, deux lettres différentes du texte non codé peuvent donner une même lettre dans le texte codé ce qui ne permet pas par la suite de retrouver le message original. On peut d'ailleurs montrer que cela arrive si et seulement si la fonction affine utilisée est telle que m n'est pas premier avec 26.

La programmation de cet algorithme nécessite l'utilisation de ce que l'on appelle les codes ASCII. En effet, en informatique, chaque caractère imprimable est représenté par un nombre entier. En particulier, les lettres minuscules sont représentées par les nombres allant de 97 (code ASCII de "a") à 122 (code ASCII de "b").

Entrées :
texte : chaîne de caractère ;
m, p : nombres.
Début traitement
  nbcar prend la valeur de longueur(*texte*) ;
  textecode prend la valeur "" ;
  pour k ***allant de*** *1* ***à*** *nbcar* **faire**
    u prend la valeur ASCII de *texte*$[k]$ otée de 97 ;
    v prend la valeur de $m \times u + p$;
    w prend la valeur du reste de la division euclidienne de v par 26 ;
    t prend la valeur du caractère de code ASCII $w + 97$;
    textecode$[k]$ prend la valeur de t.
  fin
Fin
Sorties : *textecode*

Programmes correspondants :

XCas :

```
texte:="abcdefgh";
saisir("m : ",m);
saisir("p : ",p);
afficher("Texte : "+texte);
f(x):=m*x+p;
textecode:="";
pour k de 0 jusque length(texte)-1 faire
  u:=ord(texte[k])-97;
  v:=f(u);
  w:=irem(v,26);
  t:=char(97+w);
  textecode:=textecode+t;
fpour;
afficher("Texte codé : "+textecode);
```

Python :

```
texte="cecinestquunexemple"
m=input("m : ")
p=input("p : ")
print "Texte : ",texte
textecode=""
nbcar=len(texte)
for k in range(nbcar):
  u=ord(texte[k])-97
  v=2*u-3
  w=v%26
  t=chr(97+w)
  textecode=textecode+t
print "Texte codé :",textecode
```

Remarque :
Le codage dit « de César » qui consiste à décaler les lettres d'un certain nombre de rangs est un cas particulier de codage affine : il consiste à prendre $m = 1$ et m égal au nombre de rangs de décalage.

6 Codage par la méthode de Vigenère

Les programmes suivants implémentent la méthode de codage de Vigenère qui utilise un mot clef indiquant le codage par la méthode de César à utiliser pour chacune des lettres du texte.

Programmes correspondants :

XCas :

```
texte:="cecinestquunexemple";
nbcar:=length(texte);
saisir("clef ",clef);
cleftexte:=string(clef);
longclef:=length(cleftexte);
afficher("Texte à coder : "+texte);
textecode:="";
pour k de 0 jusque nbcar-1 faire
  j:=irem(k,longclef);
  cark:=cleftexte[j];
  decalage:=ord(cark)-97;
  textecode:=textecode+...
...char(irem(ord(texte[k])...
      ...-97+decalage,26)+97);
fpour;
afficher("Texte codé : "+textecode);
```

Python :

```
texte="cecinestquunexemple"
clef=raw_input("Clef de codage : ")
longclef=len(clef)
nbcar=len(texte)
print "Texte à coder",texte
textecode=""
for k in range(nbcar):
  nbcar=len(texte)
  j=k%longclef
  cark=clef[j]
  decalage=ord(cark)-97
  textecode=textecode+...
 ...chr((ord(texte[k])-97+decalage)%26+97)
print "Texte codé : ",textecode
```

7 Algorithme RSA

Les programmes suivants, écrits pour XCas, implémentent l'algorithme de chiffrement RSA (Rivest, Shamir, Adleman). Pour des précisions théoriques sur l'algorithme, on pourra se reporter à la littérature abondante déjà existante, en particulier l'aide du logiciel XCas dans laquelle figure une autre implémentation de cet algorithme.

Rappelons que cet algorithme utilise une clé publique (u, n) nécessaire au chiffrement d'un message, et une clé privé (m, n) nécessaire au déchiffrement, n étant le produit de deux « grands » (voire « très grands ») nombres premiers p et q tenus secrets. La difficulté de la factoristation en nombres premiers garantie ainsi la robustesse de l'algorithme face aux tentatives de déchiffrement. Les caractères du message sont préalablement transformés en liste de codes ASCII correspondants puis chiffrés par blocs de plusieurs nombres (pour mettre en échec les tentatives de déchiffrement par analyse de fréquences). On a d'ailleurs fait figurer dans le programme suivant la possibilité de choisir le nombre j de caractères regroupés dans un même bloc.

7.1 Génération des clés publiques et privées

```
// Recherche de p et q premiers
saisir(p1);
p:=nextprime(p1);
saisir(q1);
q:=nextprime(q1);
n:=p*q;
// Détermination de m et u
m:=nextprime(max((p-1)/2,(q-1)/2));
tantque ((gcd(m,p)!=1) et (gcd(m,q)!=1)) faire
  m:=nextprime(m);
ftantque;
phi=(p-1)*(q-1);
u:=bezout_entiers(m,phi)[1];
tantque (u<0) faire u:=u+phi;ftantque;
afficher("Clé publique : u = "+u+", n = "+n);
afficher("Clé privée : m = "+m+", n = "+n);
// Recherche de la puissance j la plus grande telle que 26^j<min(p,q)
// pour les regroupements en blocs de j nombres.
j:=0;
tantque (26^j<min(p,q)) faire j:=j+1;ftantque;
j:=j-1;
afficher("Codage par blocs de "+j+" nombre(s) maximum");
```

7.2 Codage

Codage par l'algorithme RSA :

```
// Transformation du texte en liste de nombres.
texte:="cecinestquunexemple";
listenombres:=[];
nbcar:=length(texte);
pour k de 0 jusque nbcar-1 faire
  listenombres[k]:=ord(texte[k])-97;
fpour;
afficher(listenombres);
// Codage par bloc avec la clef publique
saisir(u);
saisir(n);
saisir(j);
nombreblocs:=intDiv(length(listenombres),j)+1;
k:=0;
listecode:=[];
tantque (k<nombreblocs) faire
    nombre:=0;
    l:=0;
    tantque (l<j) et (j*k+l<nbcar) faire
nombre:=nombre*100+listenombres[j*k+l];
l:=l+1;
ftantque;
    nombrecode:=irem(nombre^u,n);
    listecode[k]:=nombrecode;
    k:=k+1;
ftantque;
afficher(listecode);
```

Exemple :
On pourra essayer la procédure de codage avec $n = 152415790094497781$ et $u = 115769854373006801$ ainsi que des blocs de $j = 5$.
On obtient la liste $[5543549333326022 3, 70666835290124841, 102960777055406018, 5126508880992947]$ qu'il est inutile de transformer sous forme de caractères.

Remarque :
Dans l'exemple précédent, XCas trouve la factorisation de n en $p * q$ avec $p = 123456791$ et $q = 1234567891$ instantanément avec la commande `idivis(n)` mais en pratique on prend des valeurs de n bien plus grandes (de l'ordre de quelques centaines de chiffres...).

7.3 Décodage

Décodage en utilisant l'algorithme RSA :

```
saisir(m);
saisir(n);
saisir(listecode);
nombreblocs:=nops(listecode);
textedecode:="";
afficher(nombreblocs);
pour t de 0 jusque nombreblocs-1 faire
  nombredecode:=powmod(listecode[t],m,n);
  s:=0;
  textedecodebloc:="";
  taille:=length(string(nombredecode));
  si irem(taille,2)==0 alors smax:=intDiv(taille,2) sinon smax:=intDiv(taille,2)+1;fsi;
  tantque (s<smax) faire
      nombrecar:=irem(nombredecode,100);
      textedecodebloc:=textedecodebloc+char(nombrecar+97);
      nombredecode:=iquo(nombredecode,100);
      s:=s+1;
  ftantque;
  textedecodebloc2:="";
  pour y de 1 jusque length(textedecodebloc) faire
      textedecodebloc2:=textedecodebloc2+textedecodebloc[length(textedecodebloc)-y];
  fpour;
  textedecode:=textedecode+textedecodebloc2;
fpour;
afficher(textedecode);
```

Exemple :
On pourra décoder la liste de codes obtenue ci-dessus en utilisant la clef privée donnée par $n = 152415790094497781$ et $m = 12345701$ correspondante à la clef publique utilisée dans l'exemple précédent.