Programmes du TIPE de Juliette Debono
Session 2021
N°4897

Sommaire

1) MD5

```python
# -*- coding: utf8 -*-
# MD5
# Implémentation manuelle d'une fonctions de hachages très utilisée : MD5
# Plus long que la fonction de python
# Tous est fait sur des entiers sauf la normalisation qui travaille sur
les nombres en binaire
import math

def ROTL(x, n):
    """Opération de rotation binaire vers la gauche"""
    return (x << n) | x >> (32 - n)

def decomposition(l, n):
    """Décomposition de l en une liste d'entier de longueur n"""
    return [int(l[i:i+n], 2) for i in range(0, len(l), n)]

def normalisation(mot):
    """Normalise mot :
    - Le converti en bits
    - Lui ajoute 1
    - Ajoute le nombre minimal de 0 pour atteindre un multiple de 512-64
    - Retourne les bits pour être en little endian
    - Ajoute la longueur de self d'origine représenté sur 64 bits codé
en little endian"""
    # Conversion en bits
    x = ""
    for n in range(len(mot)):
        x += format(ord(mot[n]),'08b')
    lon = len(x)
    # Ajout 1
    x += "1"
    # Ajout nb 0
    while len(x) % 512 != 448:
        x += "0"
    x2 = ""
    a = 32
    # Retourner les bits en little endian
    for i in range(0, len(x), a):
        a, b, c, d, e = i, i+8, i+16, i+24, i+32
        x2 += x[d : e] + x[c : d] + x[b : c] + x[a : b]
    # Ajout longueur
    l = format(lon,'064b')
    x2 += l[32:] + l[:32]
    return x2

def toHex(value):
    """Converti un entier en hexadécimal en l'inversant"""
    value = hex(value)
    diff = len(value) - 8
    value = value[diff:]
    return ''.join([value[i*2:(i+1)*2] for i in
reversed(range(len(value)//2))])
```

1) MD5

```
64  def md5(mot):
65      """ Fonction MD5"""
66      r = [7, 12, 17, 22] * 4 + [5, 9, 14, 20] * 4 + [4, 11, 16, 23] * 4 +
67  [6, 10, 15, 21] * 4
68      K = [int(abs((2**32) * math.sin(i + 1))) for i in range(64)]
69      x = normalisation(mot)
70      x = decomposition(x, 32)
71      A, B, C, D = 0x67452301, 0xEFCDAB89, 0x98BADCFE, 0x10325476
72      for j in range(0, len(x), 16):
73          a, b, c, d = A, B, C, D
74          for t in range(64):
75              if t >= 0 and t < 16:
76                  f = (B & C) | ((~B) & D)
77                  g = t
78              elif t >= 16 and t < 32:
79                  f = (B & D) | (C & (~D))
80                  g = (5 * t + 1) % 16
81              elif t >= 32 and t < 48:
82                  f = B ^ C ^ D
83                  g = (3 * t + 5) % 16
84              elif t >= 48 and t < 64:
85                  f = C ^ (B | (~D))
86                  g = (7 * t) % 16
87              new_B = (A + f + x[j + g] + K[t]) % 2**32
88              new_B = (ROTL(new_B, r[t]) % 2**32) + B
89              A, B, C, D = D, new_B, B, C
90          A = (a + A) % 2**32
91          B = (b + B) % 2**32
92          C = (c + C) % 2**32
93          D = (d + D) % 2**32
94      return toHex(A) + toHex(B) + toHex(C) + toHex(D)
95  mot = "Demain"
96  print(md5(mot)) # b3a46a9f4cbf9bd55c64602ae9b70476
97  import hashlib
98  print(hashlib.md5(mot.encode('utf-8')).hexdigest()) #
99  b3a46a9f4cbf9bd55c64602ae9b70476
```

2) Hachage image

```
import matplotlib as plt
import matplotlib.image as mpimg
from subprocess import Popen, PIPE
import numpy as np

def hachage(mot) -> str:
    """hash mot avec md5"""
    return hashlib.new('md5', mot.encode('utf-8')).hexdigest()

def hacher(hach, fileout):
    nb_ligne, nb_pixel, nb_carreau_ligne, nb_carreau_colonne = 300, 4, 5,
3
    nb_colonne = int((nb_ligne / nb_carreau_ligne) * nb_carreau_colonne)
    liste = [int(hach[i*2:i*2+2], 16) for i in range(len(hach)//2)]
    image = np.zeros((nb_carreau_ligne, nb_carreau_colonne, nb_pixel))
    k = 0
    for i in range(nb_carreau_ligne):
        for j in range(nb_carreau_colonne):
            val = liste[k]
            image[i][j] = np.array(plt.cm.gist_rainbow(X=val))
            k += 1
    tableau_vide = np.zeros((nb_ligne, nb_colonne, nb_pixel))
    for i in range(nb_ligne):
        for j in range(nb_colonne):
            tableau_vide[i][j] = image[int(nb_carreau_ligne*i/nb_ligne),
int(nb_carreau_colonne*j/nb_colonne)]
    mpimg.imsave(fileout, tableau_vide)

file = "./Raphaël.png"
p = Popen(["md5", file], stdin=PIPE, stdout=PIPE, stderr=PIPE)
output, err = p.communicate(b"input data that is passed to subprocess'
stdin")
hach = output.decode("utf-8").strip("\n").split(" ")[-1]
fileout = "./Raphaël-Haché.png"
hacher(hach, fileout)
```

3) Force brute

```python
import time
import hashlib

def dec2base(i, caracteres):
    """Convertit i en base 10 en result en base len(caracteres) avec la liste de caractères caracteres"""
    l = len(caracteres)
    result = caracteres[i % l]
    i = (i//l) - 1
    while i > -1:
        i, result = (i // l) - 1, caracteres[i % l] + result
    return result

def hachage(mot):
    """hash mot avec md5"""
    return hashlib.new('md5', mot.encode('utf-8')).hexdigest()

def testValidite(mot : str, hach : str) -> bool :
    """Renvoie si mot est bien l'antécédent de hash avec la fonction du hachage de datas"""
    return hachage(mot) == hach

def forceBrute(mdp, mini = 1, maxi = 10, caracteres = ""):
    """Test par force brute jusqu'à ce que la valeur vaille mdp"""
    l = len(caracteres)
    N = sum([l**i for i in range(0, maxi+1)])
    cherche = sum([l**i for i in range(1, mini)])
    while hachage(dec2base(cherche, caracteres)) != mdp and cherche < N:
        cherche += 1
    return dec2base(cherche, caracteres)

caracteres =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
result = "TiP3" # Mot imposé
mini = 4
maxi = mini
hach = hachage(result) # Hache le mot choisi
print(hach, result) # Affiche le hach du mot cherché
t = time.time() # Démare le compteur de temps
mot = forceBrute(hach, mini, maxi, caracteres) # Cherche l'antécédent du hach
print(mot) # Affice le mot de passe trouvé
print(testValidite(mot, hach)) # Vérifie que c'est le bon mot
print("Temps pour recherche force brute : {} secondes".format(time.time() - t))
```

4) Temps force brute expérimental

```
1  caracteres = "abcdefghijklmnopqrstuvwxyz"
2  import time
3  import hashlib
4
5  def dec2base(i, caracteres):
6      """Convertit i en base 10 en result en base len(caracteres) avec la
7  liste de caractères caracteres"""
8      l = len(caracteres)
9      result = caracteres[i % l]
10     i = (i//l) - 1
11     while i > -1:
12         i, result = (i // l) - 1, caracteres[i % l] + result
13     return result
14
15 def hachage(mot):
16     """hash mot avec md5"""
17     return hashlib.new('md5', mot.encode('utf-8')).hexdigest()
18
19 def forceBrute(mdp, mini = 1, maxi = 10, caracteres = ""):
20     """Test par force brute jusqu'à ce que la valeur vaille mdp"""
21     tps = time.time()
22     l = len(caracteres)
23     N = sum([l**i for i in range(0, maxi+1)])
24     cherche = sum([l**i for i in range(1, mini)])
25     while hachage(dec2base(cherche, caracteres)) != mdp and cherche < N:
26         cherche += 1
27     return time.time() - tps
28
29 i = 10
30 for t in range(1, i):
31     a = forceBrute(hachage(caracteres[-1] * int(t)), t, t, caracteres)
32     print(t, a)
```

5) Temps force brute théorique

```python
import time, hashlib

def hachage(mot) -> str:
    """hash mot avec md5"""
    return hashlib.new('md5', mot.encode('utf-8')).hexdigest()

def dec2base(i, caracteres):
    """Convertit i en base 10 en result en base len(caracteres) avec la
liste de caractères caracteres"""
    l = len(caracteres)
    result = caracteres[i % l]
    i = (i//l) - 1
    while i > -1:
        i, result = (i // l) - 1, caracteres[i % l] + result
    return result

def red(h : str, i, t : int, n) -> int :
    """Transforme un hash en indice"""
    N = n + l**i
    h = str(h)
    return (int(h, 16) + t) % N

caracteres =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
l = len(caracteres)
e = 10000
y = []
for i in range(1, 15):
    n = sum([l**k for k in range(1, i+1)]) - 1
    h = hachage(dec2base(n, caracteres))
    st = time.time()
    for _ in range(e):
        # On réalise l'opération un certain nombre de fois
        hachage(dec2base(red(h, i, 1, n), caracteres))
    t = (time.time() - st) / e
    y.append(n*t)
print(y)

# iMac 2009
# [1, 2, 3, 4, 5, 6, 7]
# [0.00015807151794433594, 0.005568981170654297, 0.11891508102416992,
3.490976095199585, 86.7342791557312, 2402.337882757187,
65502.59010100365]
# MacBookAir 2017
# [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
# [0.0002609647989273071, 0.019876070737838748, 1.3695568049669264,
101.19039012982844, 6791.717667701912, 419072.78867693007,
26370887.313664626, 1774671384.1215305, 114197049354.61603,
7421171615163.042, 422434355080551.3, 3.093496404845014e+16,
2.04761186994996e+18, 1.3255018005042317e+20]
```

6) Courbe force brute

```
import matplotlib.pyplot as plt
import numpy as np

arrondi = lambda l, x, e : str(int(round(x/(10**(l-e)), 0))) + "e" +
str(l-e) if l > e else str(x)

def temps(s):
    s = round(s, 3)
    m, h, j, a = 60, 3600, 3600*24, 3600*24*365
    if s > a:
        x = int(round(s/a, 0))
        return arrondi(len(str(x)), x, 3) + "a"
    elif s > j:
        x = int(round(s/j, 0))
        return arrondi(len(str(x)), x, 3) + "j"
    elif s > h:
        x = int(round(s/h, 0))
        return arrondi(len(str(x)), x, 3) + "h"
    elif s > m:
        x = int(round(s/m, 0))
        return arrondi(len(str(x)), x, 3) + "min"
    else:
        return str(s) + "s"

def courbe(y, z):
    couleur = 'purple'
    plt.rcParams.update({'font.sans-serif':'Tahoma'})
    plt.clf()
    fig = plt.figure(1, figsize=(6, 7))
    ax1 = fig.add_subplot(2, 1, 1)
    plt.yticks(fontsize = 10)
    plt.xticks(fontsize = 10)
    plt.yscale('log')
    i = len(z)
    x = np.linspace(1, i, i)
    ax1.plot(x, z, couleur)
    (xmin, xmax) = ax1.xaxis.get_view_interval()
    (ymin, ymax) = ax1.yaxis.get_view_interval()
    ax1 = fig.add_subplot(2, 1, 1)
    plt.yscale('log')
    i = len(y)
    x = np.linspace(1, i, i)
    ax1.plot(x, y, couleur)
    ax2 = ax1.twinx()
    plt.yscale('log')
    ax2.plot(x, y, couleur)
    ax1.set_xlim(xmin, xmax)
    ax2.set_xlim(xmin, xmax)
    ax1.set_ylim(ymin, ymax)
    ax2.set_ylim(ymin, ymax)
    ax2.yaxis.set_ticklabels([temps(i) for i in ax1.get_yticks()],
fontsize = 8)
    ax1.set_ylabel('temps (s)', fontsize = 12)
    ax3 = fig.add_subplot(2, 1, 2)
    plt.yscale('log')
    i = len(z)
    x = np.linspace(1, i, i)
    ax3.plot(x, z, couleur)
    ax4 = ax3.twinx()
    plt.yscale('log')
    ax4.plot(x, z, couleur)
    ax4.yaxis.set_ticklabels([temps(i) for i in ax3.get_yticks()],
fontsize = 8)
```

6) Courbe force brute

```
64      ax3.set_xlabel('nombre lettres', fontsize = 12)
65      ax3.set_ylabel('temps (s)', fontsize = 12)
66      ax1.set_title("Temps pour générer tous les mots de n lettres\n",
67  fontsize = 16)
68      plt.savefig("Temps pour générer tous les mots jusqu'à {}
69  caractères.png".format(i))
70      plt.show()
71
72  courbe([0.00015807151794433594, 0.005568981170654297,
73  0.11891508102416992, 3.490976095199585,
74              86.7342791557312, 2402.337882757187,65502.59010100365],
75          [0.0002609647989273071, 0.019876070737838748, 1.3695568049669264,
76  101.19039012982844,
77              6791.717667701912, 419072.78867693007, 26370887.313664626,
78  1774671384.1215305, 114197049354.61603,
79              7421171615163.042, 422434355080551.3, 3.093496404845014e+16,
80  2.04761186994996e+18, 1.3255018005042317e+20])
```

7) Création base de données Mémoire

```
# Liste de tous les caractères possibles dans le mot de passe :
caracteres =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
# Création base de données avec tout d'une certaine longueur et de
certains caractères
import sqlite3
import hashlib
import time

def dec2base(i, caracteres):
    """Convertit i en base 10 en result en base len(caracteres) avec la
liste de caractères caracteres"""
    l = len(caracteres)
    result = caracteres[i % l]
    i = (i//l) - 1
    while i > -1:
        i, result = (i // l) - 1, caracteres[i % l] + result
    return result

def forceBrute(mini = 1, maxi = 10, caracteres = caracteres):
    """Test par force brute jusqu'à ce que la valeur vaille mdp"""
    liste = []
    dic = {}
    l = len(caracteres)
    cherche = sum([l**i for i in range(1, mini)])
    mot = dec2base(cherche, caracteres)
    while len(mot) <= maxi:
        mot = dec2base(cherche, caracteres)
        liste.append([mot, hachage(mot)])
        cherche += 1
    return liste

def hachage(mot):
    """hash mot avec une fonction de hachage"""
    return hashlib.md5(mot.encode('utf8')).hexdigest()

def creation(database, dic):
    conn = sqlite3.connect(database)
    cur = conn.cursor()
    cur.execute("""CREATE TABLE Memoire (mot CHARACTER, hash CHARACTER)
""")
    for mot in dic:
        cur.execute("""Insert Into Memoire (mot, hash) VALUES (?, ?)""",
mot)
    conn.commit()
    conn.close

def creer_table(n):
    database = "./Memoire{}.sqlite".format(n)
    dic = forceBrute(n, n, caracteres)
    creation(database, dic)

t = time.time()
creer_table(4)
print("Temps pour créer table mémoire : {} secondes".format(time.time() -
t))

# 0 0.0039310455322265625
# 1 0.009385108947753906
# 2 0.026437997817993164
# 3 0.5101959705352783
# 4 11.037984132766724
# 5 266.90790915489197
```

8) Recherche dans base de données Mémoire

```
import sqlite3
import hashlib
import time

def est_bon_mot(mot_de_passe, mot_haché):
    """Renvoie si mot_de_passe est bien l'antécédent de mot_haché avec
la fonction du hachage"""
    return hachage(mot_de_passe) == mot_haché

def hachage(mot):
    """hash mot avec une fonction de hachage"""
    return hashlib.md5(mot.encode('utf8')).hexdigest()

def recherche(database, hash):
    conn = sqlite3.connect(database)
    cur = conn.cursor()
    cur.execute("""SELECT mot FROM Memoire WHERE hash =
'{}'""".format(hash))
    mot = cur.fetchone()[0]
    conn.close
    return mot

caracteres = 'abcdefghijklmnopqrstuvwxyz'
mot = "TiP3"
mothash = hachage(mot)
longueur = 4
database = "./Memoire{}.sqlite".format(longueur)

t = time.time()
print(recherche(database, mothash))
print("Temps pour recherche dans table mémoire : {}
secondes".format(time.time() - t))
```

9) Courbe Mémoire

```
import matplotlib.pyplot as plt
import numpy as np
from math import log, exp

unite = ["o", "ko", "Mo", "Go", "To"]
arrondi = lambda l, x, e : str(int(round(x/(10**(l-e)), 0))) + "e" +
str(l-e) if l > e else str(x)

alphabet =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
l = len(alphabet)

def taille(n):
    nb_ligne = l**n
    mot = n
    hach = 32 # Hach codé en octets : taille 32
    carac = 2
    taille = nb_ligne * (carac + hach + mot)
    return taille

def tailleSTR(o):
    o_s = str(int(o))
    l = len(o_s)
    if l > len(unite)*3:
        o = int(round(o/(10**(12)), 0))
        l = len(str(o)) - 1
        o_s = arrondi(l, o, 0) + unite[-1]
    elif l > 3:
        l -= 1
        l1 = l % 3
        o_s = str(int(round(o/(10**(l - l1)), 0))) + unite[((l-l1) // 3)]
    else:
        o_s += unite[0]
    return o_s

def courbe(i):
    couleur = 'purple'
    plt.rcParams.update({'font.sans-serif':'Tahoma'})
    plt.clf()
    fig = plt.figure(2, figsize=(7, 6))
    ax1 = fig.add_subplot()
    plt.yticks(fontsize = 10)
    plt.xticks(fontsize = 10)
    plt.yscale('log', basey=10)
    ax2 = ax1.twinx()
    x = np.linspace(1, i, i)
    y = [taille(i) for i in x]
    plt.yscale('log', basey=10)
    ax1.plot(x, y, couleur)
    ax2.plot(x, y, couleur)
    ax2.yaxis.set_ticklabels([tailleSTR(i) for i in ax1.get_yticks()],
fontsize = 10)
    ax1.set_xlabel('nombre lettres', fontsize = 12)
    ax1.set_ylabel('poids (octets)', fontsize = 12)
    plt.title("Poids pour générer tous les mots de n lettres\n", fontsize
= 16)
    plt.savefig("Poids pour générer tous les mots jusqu'à {}
caractères.png".format(i))
    plt.show()

courbe(14)
```

10) Recherche dictionnaire

```python
import sqlite3
import hashlib

database = "./Mots.sqlite"

def hachage(mot):
    """hash mot avec md5"""
    return hashlib.new('md5', mot.encode('utf-8')).hexdigest()

def recherche(hash):
    conn = sqlite3.connect(database)
    cur = conn.cursor()
    cur.execute("""SELECT ortho FROM MOTS ORDER BY freqlemfilms DESC""")
    liste = cur.fetchall()
    for mot in liste:
        mot = mot[0]
        if mot is not None and hachage(mot) == hash:
            return mot
mot = "mot"
print(recherche(hachage(mot)))
```

11) Classe Arc-en-ciel

```
"""
Classe ArcEnciel
Contient * les données pour les tables de compromis temps-mémoires
         * les fonctions pour générer une table ou rechercher un mot
dedans
TIPE
"""
import hashlib
import random as rd
import sqlite3
import numpy as np
__version__ = "1.1.1"
__author__ = "Juliette Debono"

class ArcEnCiel:
    """Classe pour générer une table de compromis temps-mémoires
    Objet avec attributs : informations nécéssaires pour créer la table
    Fonctions : fonctions de base pour créer la table
    """
    def __init__(self, m : int, t : int, l : int, t_min : int, t_max :
int, carac : str, type : bool, database : str, hachage : 'function') ->
None:
        """Création d'un objet contenant les informations :
        ** Informations directements transmises :
        * m : Nombre de lignes de la base de données
        * t : Nombre de fois qu'on applique la réduction
        * l : Nombre de tables
        * t_min : longueur minimale des mots de passe possibles
        * t_max : longueur maximale des mots de passe possibles
        * carac : Liste des caractères possibles
        * type : Type de la table : ArcEnCiel ou Classique
        * database : Nom de la base de données
        * hachage : Fonction de hachage utilisée
        ** Généré à partir des autres valeurs :
        * N : nombre de mots possible avec les caractéristiques données
              (nombre de caractère et longueur des mots de passe)
        * nblettres : Nombre de caractères dans la chaine carac
        * couverture : Pourcentage de succès qu'un mot soit dans la
table
        """
        nblettres = len(carac)
        N = sum([nblettres**i for i in range(t_min, t_max+1)])
        self._m = m
        self._t = t
        self._l = l
        self._type = type
        self._t_min = t_min
        self._t_max = t_max
        self._carac = carac
        self._database = database
        self._hachage = hachage
        self._N = N
        self._nblettres = nblettres
        self._couverture = self.couv()

    def __repr__(self) -> str:
        """Montrer l'objet"""
        return """N : {}
t : {}
m : {}
l : {}
lettre entre {} et {}
couverture = {}
```

11) Classe Arc-en-ciel

```
database : {}
carac : {}
hachage : {}
nblettres : {}""".format(self.N, self.t, self.m, self.l, self.t_min,
self.t_max, self.couverture, self.database, self.carac, self.hachage,
self.nblettres)

    def _impossible(self, *args):
        """Empêche de supprimer / modifier un attribut"""
        print("Impossible")
    def _get_m(self):
        """Retourne m"""
        return self._m
    def _set_m(self, m):
        """Modifie m"""
        self._m = m
        self._couverture = self.couv()
    m = property(_get_m, _set_m, _impossible)

    def _get_t(self):
        """Retourne t"""
        return self._t
    def _set_t(self, t):
        """Modifie t"""
        self._t = t
        self._couverture = self.couv()
    t = property(_get_t, _set_t, _impossible)

    def _get_l(self):
        """Retourne l"""
        return self._l
    def _set_l(self, l):
        """Modifie T"""
        self._l = l
    l = property(_get_l, _set_l, _impossible)

    def _get_t_min(self):
        """Retourne t_min de mots de passe possibles"""
        return self._t_min
    def _set_t_min(self, t_min):
        """Modifie t_min"""
        self._t_min = t_min
        self._N = sum([self.nblettres**i for i in range(self._t_min,
self._t_max+1)])
        self._couverture = self.couv()
    t_min = property(_get_t_min, _set_t_min, _impossible)

    def _get_t_max(self):
        """Retourne les tailles de mots de passe possibles"""
        return self._t_max
    def _set_t_max(self, t_max):
        """Modifie taille"""
        self._t_max = t_max
        self._N = sum([self.nblettres**i for i in range(self._t_min,
self._t_max+1)])
        self._couverture = self.couv()
    t_max = property(_get_t_max, _set_t_max, _impossible)

    def _get_type(self):
        """Retourne type"""
        return self._type
    def _set_type(self, type):
        """Modifie type"""
```

11) Classe Arc-en-ciel

```
127             self._type = type
128             self._couverture = self.couv()
129             self._database = "/Users/juliettedebono/Documents/MP*/TIPE/III-
130     Compromis/III- {5}) {0}/Tables/{0} t = {1} m = {2} l = {3} len =
131     {4}.sqlite".format("{0}", self._t, self._m, self._l, self._t_max,
132     lettre(type))
133         type = property(_get_type, _set_type, _impossible)
134
135         def _get_carac(self):
136             """Retourne les caractères possibles"""
137             return self._carac
138         def _set_carac(self, carac):
139             """Modifie carac"""
140             self._carac = carac
141             self._nblettres = len(carac)
142             self._N = sum([self.nblettres**i for i in range(self._t_min,
143     self._t_max+1)])
144         carac = property(_get_carac, _set_carac, _impossible)
145
146         def _get_database(self):
147             """Retourne le nom de la base de données"""
148             return self._database.format(self.type)
149         def _set_database(self, database):
150             """Modifie database"""
151             self._database = database
152         database = property(_get_database, _set_database, _impossible)
153
154         def _get_hachage(self):
155             """Retourne la fonction de hachage"""
156             return self._hachage
157         def _set_hachage(self, hachage):
158             """Modifie la fonction de hachage"""
159             self._hachage = hachage
160         hachage = property(_get_hachage, _set_hachage, _impossible)
161
162         def _get_couverture(self):
163             """Retourne la valeur de la couverture"""
164             return self._couverture
165         couverture = property(_get_couverture, _impossible, _impossible)
166
167         def _get_N(self):
168             """Retourne N"""
169             return self._N
170         N = property(_get_N, _impossible, _impossible)
171
172         def _get_nblettres(self):
173             """Retourne le nombre de lettres"""
174             return self._nblettres
175         nblettres = property(_get_nblettres, _impossible, _impossible)
176
177         def couv(self):
178             """Renvoie la couverture de datas : le pourcentage de valeur
179     contenue dans la table type"""
180             if self.type == 'ArcEnCiel':
181                 m = self.m
182                 v = 1.0
183                 for _ in range(self.t):
184                     v *= (1 - m / self.N)
185                     m = self.N * (1 - np.exp(-m/self.N))
186                 p = (1 - (v**self.l))
187                 return round(100 * p, 2)
188             else:
189                 if self.m * (self.t**2) < 10 * self.N:
```

11) Classe Arc-en-ciel

```
                p = (self.m*self.t*self.l)/self.N
                return round(100 * p, 2)
            else:
                p = 0.8 * ((self.m*self.t*self.l)/self.N)
                return round(100 * p, 2)

    def ialea(self, tableName, l) -> int:
        """Indice aléatoire de départ d'une ligne"""
        conn = sqlite3.connect(self.database.format(tableName))
        cur = conn.cursor()
        i = 0
        def inter(cur, conn, tableNamel, l, i):
            """Choisi indice aleatoire qui n'est pas dans la table : si
il y est : récursion"""
            i_0 = rd.randint(0, self.N - 1)
            cur.execute("""SELECT * FROM {}{} WHERE i_0
= ?""".format(tableName, l),(i_0,))
            if cur.fetchone() is None:
                return i_0
            else:
                i += 1
                try:
                    return inter(cur, conn, tableName, l, i)
                except RecursionError:
                    return rd.randint(0, self.N)
        i_0 = inter(cur, conn, tableName, l, i)
        conn.close
        return i0

    def i2c(self, i : int) -> str:
        """Convertit i en base 10 en x en base len(carac) avec la liste
de caractères carac"""
        l = self.nblettres
        N = sum([l**i for i in range(1, self.t_min)])
        i += N
        result = self.carac[i % l]
        i = (i//l) - 1
        while i > -1 and len(result) < self.t_max+1:
            i, result = (i // l) - 1, self.carac[i % l] + result
        return result

     def h(self, c : str) -> str:
        """hash c avec une fonction de hachage de nom hachage[0] et
d'encodage hachage[1]"""
        return hashlib.new(self.hachage[0],
c.encode(self.hachage[1])).hexdigest()

    def h2i(self, h : str, t : int) -> int :
        """Transforme un hash en indice"""
        h = str(h)
        return (int(h, 16) + t) % self.N

    def h2h(self, h1 : int, t : int) -> str:
        """Passe d'un hash au suivant et renvoie le clair et le hash"""
        i2 = self.h2i(h1, t)
        c2 = self.i2c(i2)
        h2 = self.h(c2)
        return c2, h2

    def i2i(self, i1 : int, t : int) -> int:
        """Réalise la génération de l'indice d'après avec les fonctions
préalablement établies"""
        c1 = self.i2c(i1)
```

11) Classe Arc-en-ciel

```
253             h1 = self.h(c1)
254             i2 = self.h2i(h1, t)
255             return i2
256 
257         def pick(self) -> str:
258             """Choisi un hash stocké dans la table aléatoirement"""
259             conn = sqlite3.connect(self.database)
260             cur = conn.cursor()
261             l = rd.randint(0, self.l - 1)
262             cur.execute("""SELECT i_0 FROM {}{} ORDER BY
263 RANDOM()""".format(self.type, l))
264             a = cur.fetchone()[0]
265             conn.close
266             for i in range(rd.randint(1, self.t - 1)):
267                 if self.type != "ArcEnCiel":
268                     i = l
269                 else:
270                     i += 1
271                 a = self.i2i(a, i)
272             return self.i2c(a)
273 
274 # Informations :
275 carac = 'abcdefghijklmnopqrstuvwxyz'+'abcdefghijklmnopqrstuvwxyz'.upper()
276 + '0123456789'
277 t_min = 4
278 t_max = t_min
279 t = 1000
280 m = 100000
281 l = 1
282 type = "ArcEnCiel"
283 hachage = 'md5', 'utf-8' # md5
284 lettre = lambda a : "b" if a == 'ArcEnCiel' else "a"
285 database = "/Users/juliettedebono/Documents/MP*/TIPE/III- Compromis/III-
286 {5}) {0}/Tables/{0} t = {1} m = {2} l = {3} len =
287 {4}.sqlite".format("{0}", t, m, l, t_max, lettre(type))
288 datas = ArcEnCiel(m, t, l, t_min, t_max, carac, type, database, hachage)
289 # Objet contenant les informations
290 del carac, type, t, m, l, hachage, database, t_min, t_max # Suppression
291 valeurs inutiles
```

12) Création table compromis

```
import time
import sqlite3
from ArcEnCiel import *

def table(datas : ArcEnCiel) -> None:
    """Création table de compromis
    Renvoie une base de donnée d'une table arc en ciel :
    Colonne 1 : indice d'origine
    Colonne 2 : indice t fois de l'indice d'origine
    """
    start_time = time.time()
    tableName = datas.type
    conn = sqlite3.connect(datas.database)
    cur = conn.cursor()
    for l in range(datas.l):
        print("Table {}".format(l+1))
        tableNamel = tableName + str(l)
        try:
            cur.execute("""DROP TABLE {}""".format(tableNamel))
        except sqlite3.OperationalError:
            pass
        cur.execute("""CREATE TABLE {}(
            i_0 INT,
            i_t INT);""".format(tableNamel))
        for m in range(datas.m):
            i_0 = datas.ialea(tableName, l)
            i_t = i_0
            for t in range(datas.t):
                if tableName != 'ArcEnCiel':
                    t = l
                i_t = datas.i2i(i_t, t)
            cur.execute("""
                INSERT INTO {}
                (i_0, i_t)
                VALUES(?, ?);""".format(tableNamel), (i_0, i_t))
            conn.commit()
            if m % 200 == 0:
                print("{} secondes pour {}
lignes".format(round(time.time() - start_time), m))
    conn.commit()
    conn.close

# Générer table
def creer():
    start_time = time.time()
    table(datas) # Création table AEC
    tableName = datas.type
    print("Création table {} : {} secondes\n".format(tableName,
time.time() - start_time))

creer()
```

13) Recherche table compromis

```
import time
import random as rd
from ArcEnCiel import *

def recherche(h : int, hash : str, l : int, i : int, datas : ArcEnCiel) -
> str:
    """
    * On donne le hash qu'on cherche et le premier hash de la ligne de
notre table où est situé le mot
    * Applique l'algo jusqu'à ce que le hash i fois :
        - Si c'est le bon : renvoie la valeur avant hachage
        - Sinon renvoie False
    """
    for t in range(1, i):
        if h == hash:
            return c
        if not datas.type:
            t = l
        c, h = datas.h2h(h, t)
    if h == hash:
        return c
    return False

def inter(ligne : tuple, l : int, i : int, hash : str, datas :
ArcEnCiel):
    if ligne is not None:
        for id in ligne:
            c1 = datas.i2c(id[0])
            h1 = datas.h(c1)
            if h1 == hash:
                return c1
            else:
                result = recherche(h1, hash, l, i, datas)
                if result:
                    return result # c'est la bonne ligne
    return False

def inverse(hash : str, datas : ArcEnCiel) -> str:
    """Cherche le mot de passe d'origine ayant hash comme image
hachée"""
    tableName = datas.type
    conn = sqlite3.connect(datas.database)
    cur = conn.cursor()
    if datas.type == 'ArcEnCiel':
        for i in reversed(range(datas.t + 1)):
            indice = datas.h2i(hash, i)
            for t in range(i, datas.t):
                indice = datas.i2i(indice, t)
            for l in range(datas.l):
                # On récupère les dernières lignes lorsque i_t est dans
colonne dans toutes les l tables
                tableNamel = tableName + str(l)
                cur.execute("""SELECT i_0 FROM {} WHERE i_t
= ?""".format(tableNamel), (indice,))
                # t = time.time()
                result = inter(cur.fetchall(), l, i, hash, datas)
                # print(time.time() - t)
                if result:
                    return result
            if i % 100 == 0:
                print("Colonne {}".format(i))
    else:
        for l in range(datas.l):
```

13) Recherche table compromis

```
64                print("Table {}".format(l))
65                # Dans chaque table
66                indice = datas.h2i(hash, l)
67                tableNamel = tableName + str(l)
68                for t in range(datas.t):
69                    indice = datas.i2i(indice, l)
70                    cur.execute("""SELECT i_0 FROM {} WHERE i_t
71    = ?""".format(tableNamel), (indice,))
72                    result = inter(cur.fetchall(), l, t, hash, datas)
73                    if result:
74                        return result
75                    if t % 100 == 0:
76                        print("Colonne {}".format(t))
77        conn.close
78        return "Pas dans la table"
79
80    def testValidite(mot : str, hach : str, datas : ArcEnCiel) -> bool :
81        """Renvoie si mot est bien l'antécédent de hash avec la fonction du
82    hachage de datas"""
83        return datas.h(mot) == hach
84
85    def alea(datas : ArcEnCiel):
86        """Génère un mot aléatoire suivant les caractéristiques de la
87    table"""
88        result = ""
89        for _ in range(rd.randint(datas.t_min, datas.t_max)):
90            result += rd.choice(datas.carac)
91        return result
92
93    # result = datas.pick() # Choisi aléatoirement un hach qui est dans la
94    base de données
95    # result = alea(datas) # Choisi aléatoirement un mot qui pourrait être
96    dans la base de données
97    result = "TiP3" # Mot imposé
98    hach = datas.h(result) # Hache le mot choisi
99    print(hach, result) # Affiche le hach du mot cherché
100   t = time.time() # Démare le compteur de temps
101   mot = inverse(hach, datas) # Cherche l'antécédent du hach
102   print(mot) # Affice le mot de passe trouvé
103   print(testValidite(mot, hach, datas)) # Vérifie que c'est le bon mot
104   print("Temps pour recherche table {} : {} secondes".format(datas.type,
105   time.time() - t))
```

14) Base de données temps de création table arc-en-ciel

```
from creationtable import table
import ArcEnCiel
import time
import sqlite3

carac = 'abcdefghijklmnopqrstuvwxyz'
t_min, t_max = 4, 4
database = "./T M temps EnCours.sqlite"
hachage = 'md5', 'utf-8' # md5
type = 'ArcEnCiel'
datas = ArcEnCiel.ArcEnCiel(1, 1, 1, t_min, t_max, carac, type, database,
hachage) # Objet contenant les informations

database = "./T M temps T = 9000 M = 12000 Pas = 1000.sqlite"
conn = sqlite3.connect(database)
cur = conn.cursor()
try:
    cur.execute("""CREATE TABLE Temps(
        M INT,
        T INT,
        temps INT);""")
except sqlite3.OperationalError:
    pass
else:
    print("Nouvelle Table")

for T in range(3000, 9001, 500):
    for M in range(3000, 12001, 500):
        cur.execute("""SELECT * FROM Temps WHERE T = ? AND M = ?""",(T,
M))
        if cur.fetchone() is None:
            # T et M n'a pas encore été calculé : on le calcule
            print("T :", T, ", M :", M)
            datas.t = T
            datas.m = M
            temps = time.time()
            text = table(datas)
            temps = time.time() - temps
            cur.execute("""
            INSERT INTO Temps
            (M, T, temps)
            VALUES(?, ?, ?);""",(M, T, temps))
            conn.commit()
            print("Création table arc en ciel T : {} , M : {} : {}
secondes\n".format(T, M, temps))
        else:
            print(T, M, "Déjà dans la table")
conn.close
```

15) Courbe 3D temps création table arc-en-ciel

```
tempsf = lambda x : ((6.5*(10**(-6))) * (x**2)) - 0.00063 * x + 2.65
# Temps de recherche en fonction de T de la table (expérimental)
arrondi = lambda l, x, e : str(int(round(x/(10**(l-e)), 0))) + "e" +
str(l-e) if l > e else str(x)
taille = lambda nb_ligne : nb_ligne * (2 + 4*2)
unite = ["o", "ko", "Mo", "Go", "To"]

def tempsSTR(s):
    s = round(s, 3)
    m, h, j, a = 60, 3600, 3600*24, 3600*24*365
    if s > a:
        x, string = int(round(s/a, 0)), "a"
    elif s > j:
        x, string = int(round(s/j, 0)), "j"
    elif s > h:
        x, string = int(round(s/h, 0)), "h"
    elif s > m:
        x, string = int(round(s/m, 0)), "min"
    else:
        return str(s) + "s"
    return arrondi(len(str(x)), x, 3) + string

def tailleSTR(o):
    o_s = str(int(o))
    l = len(o_s)
    if l > len(unite)*3:
        o = int(round(o/(10**(12)), 0))
        l = len(str(o)) - 1
        return arrondi(l, o, 0) + unite[-1]
    elif l > 3:
        l -= 1
        l1 = l % 3
        return str(int(round(o/(10**(l - l1)), 0))) + unite[((l-l1) //
3)]
    else:
        return o_s + unite[0]

import matplotlib.pyplot as plt
import numpy as np
import sqlite3

database = "./T M temps T = 9000 M = 12000 Pas = 1000.sqlite"
conn = sqlite3.connect(database)
cur = conn.cursor()
cur.execute("""SELECT SUM(temps) FROM Temps""")
val = cur.fetchone()[0]
cur.execute("""SELECT T, M, temps FROM Temps""")
tableau = np.array(cur.fetchall())
T, M, temps = tableau[:,0], tableau[:,1], tableau[:,2]

plt.rcParams.update({'font.sans-serif':'Tahoma'})
fig = plt.figure(1, figsize=(8, 7))
ax = fig.gca(projection = '3d')
my_cmap = plt.get_cmap('rainbow')
trisurf = ax.plot_trisurf(T, M, temps, linewidth = 0.1, antialiased =
False, cmap = my_cmap)
ax.scatter(T, M, temps, marker='_', color = "r", alpha = 0.1) # Points
ax.yaxis.set_ticklabels([tailleSTR(taille(i)) for i in ax.get_yticks()],
fontsize = 10)
ax.xaxis.set_ticklabels([tempsSTR(tempsf(i)) for i in ax.get_xticks()],
fontsize = 10)
ax.zaxis.set_ticklabels([tempsSTR(i) for i in ax.get_zticks()], fontsize
= 10)
```

15) Courbe 3D temps création table arc-en-ciel

```
64  cb = fig.colorbar(trisurf, ax = ax, shrink = 0.8)
65  cb.ax.set_yticklabels([tempsSTR(i) for i in ax.get_zticks()], fontsize =
66  10)
67  ax.view_init(*(25, -155))
68  ax.set_title('Temps en fonction de M et T', fontsize = 16)
69  ax.set_xlabel('\n\ntemps de recherche\nproportionnel à T', fontsize = 12)
70  ax.set_ylabel('\nmémoire\nproportionnelle à M', fontsize = 12)
71  ax.set_zlabel('\ntemps création', fontsize = 12)
72  plt.savefig("./Courbes T = 9000 M = 12000 Pas = 1000")
```

16) Comparaison fusions

```python
# Fusion
# On cherche un mot dans la table : on regarde dans combien de lignes il
apparait
import time
import random as rd
from ArcEnCiel import *

def recherche(h : int, hash : str, l : int, i : int, datas : ArcEnCiel) -
> str:
    """
    * On donne le hash qu'on cherche et le premier hash de la ligne de
notre table où est situé le motj
    * Applique l'algo jusqu'à ce que le hash i fois :
        - Si c'est le bon : renvoie la valeur avant hachage
        - Sinon renvoie False
    """
    for t in range(1, i):
        if h == hash:
            return c
        if datas.type != 'ArcEnCiel':
            t = l
        c, h = datas.h2h(h, t)
    if h == hash:
        return c
    return False

def inter(ligne : tuple, l : int, i : int, hash : str, datas :
ArcEnCiel):
    if ligne is not None:
        for id in ligne:
            c1 = datas.i2c(id[0])
            h1 = datas.h(c1)
            if h1 == hash:
                return c1
            else:
                result = recherche(h1, hash, l, i, datas)
                if result:
                    return result # c'est la bonne ligne
    return False

def inverse(hash : str, datas : ArcEnCiel) -> str:
    """Cherche le mot de passe d'origine ayant hash comme image
hachée"""
    tableName = datas.type
    conn = sqlite3.connect(datas.database)
    cur = conn.cursor()
    count = 0
    if datas.type == 'ArcEnCiel':
        for i in reversed(range(datas.t + 1)):
            indice = datas.h2i(hash, i)
            for t in range(i, datas.t):
                indice = datas.i2i(indice, t)
            indiceList = []
            for l in range(datas.l):
                # On récupère les dernières lignes lorsque i_t est dans
colonne dans toutes les l tables
                tableNamel = tableName + str(l)
                cur.execute("""SELECT i_0 FROM {} WHERE i_t
= ?""".format(tableNamel), (indice,))
                result = inter(cur.fetchall(), l, i, hash, datas)
                if result:
                    count += 1
    else:
```

16) Comparaison fusions

```
64          for l in range(datas.l):
65              # Dans chaque table
66              indice = datas.h2i(hash, l)
67              tableNamel = tableName + str(l)
68              for t in range(datas.t):
69                  indice = datas.i2i(indice, l)
70                  cur.execute("""SELECT i_0 FROM {} WHERE i_t
71  = ?""".format(tableNamel), (indice,))
72                  result = inter(cur.fetchall(), l, t, hash, datas)
73                  if result:
74                      count += 1
75      conn.close
76      return count
77
78  def alea(datas : ArcEnCiel):
79      """Génère un mot aléatoire suivant les caractéristiques de la
80  table"""
81      result = ""
82      for _ in range(datas.tailles[rd.randint(0, len(datas.tailles) - 1)]):
83          result += rd.choice(datas.carac)
84      return result
85
86  def fusions(datas):
87      listeClassic = []
88      listeAEC = []
89      listeMot = []
90      for i in range(10):
91          datas.type = "Classique"
92          mot = datas.pick()
93          listeMot.append(mot)
94          hash = datas.h(mot)
95          listeClassic.append(inverse(hash, datas))
96          datas.type = 'ArcEnCiel'
97          listeAEC.append(inverse(hash, datas))
98          print("ET DE {}".format(i+1))
99          print(listeClassic, listeAEC)
100     return listeClassic, listeAEC, listeMot
101
102 listeClassic, listeAEC, listeMot = fusions(datas)
103 print(listeClassic, listeAEC, listeMot)
104 """
105 t = 1000
106 m = 5000
107 l = 1
108 N = 456976
109 listeMot = ['hlzw', 'mlcb', 'eesr', 'bgko', 'nqon', 'qtpx', 'qcvx',
110 'vutc', 'hiai', 'wnbz']
111 listeClassic = [343, 686, 681, 838, 687, 716, 663, 737, 839, 841]
112 listeAEC = [11, 4, 11, 2, 3, 4, 3, 6, 5, 9]
113 """
```

17) Comparaison méthodes

```
import random as rd
import time
from ArcEnCiel import *
import numpy as np

database = "/Users/juliettedebono/Documents/MP*/TIPE/II- Basique/II- b)
Attaque memoire/Memoire4.sqlite"
n = 100
mots = [alea(datas) for _ in range(n)]
temps_recherche = np.zeros((4, n))

for i in range(len(mots)):
    mot = mots[i]
    hach = hachage(mot)
    print(i)

    print("Force Brute")
    t = time.time()
    mot2 = forceBrute(hach, 4, 4, caracteres)
    temps_recherche[0, i] = time.time() - t

    print("Mémoire")
    t = time.time()
    mot2 = rechercheM(database, hach)
    temps_recherche[1, i] = time.time() - t

    print("Classique")
    datas.type = "Classique"
    t = time.time()
    mot2 = inverse(hach, datas)
    temps_recherche[2, i] = time.time() - t

    print("Arc en ciel")
    datas.type = "ArcEnCiel"
    t = time.time()
    mot2 = inverse(hach, datas)
    temps_recherche[3, i] = time.time() - t

print(mots)
print(temps_recherche)
print(np.mean(temps_recherche, axis = 1))
print(np.median(temps_recherche, axis = 1))
print(np.std(temps_recherche, axis = 1))


#                   Force Brute    Mémoire    Classique   ArcEnCiel
# Temps création      [Liste]      [Liste]     [Liste]     [Liste]
# Temps recherche     [Liste]      [Liste]     [Liste]     [Liste]
# Mémoire             [Liste]      [Liste]     [Liste]     [Liste]

# mots = ['kICZ', 'ZTK4', 'xO4u', 'p0wK', 'wXxk', 'OrYy', 'WClb', '58H7',
'o3HW', 'ytXF', 'PGI6', '2WDc', 'UIzl', 'sw7T', 'NBIw', 'FNAF', '67qO',
'wbil', 'J6vI', 'F5B5', '9rmg', 'Orjf', '0YNc', 'SomR', 'ds8s', 'drCk',
'qTwc', 'Q1a7', 'T9sH', 'bbyV', 'l5kX', 'izEM', 'YqJ2', 'Xs2X', 'u8Nw',
'7yND', 'K1UY', 'jnC0', '9Oqd', 'aXYp', 'plSW', 'w72z', 'VBBp', 'lnfl',
'rFPF', 'a3aF', '87JV', 'xaYn', 'sRbM', 'spl4', 'yT7g', 'JvBi', 'ZSOD',
'yOe4', '1CgS', 'A2aQ', 'DmeR', 'qBFB', 'v2YO', 'pyA1', 'UEB2', 'ImQf',
'VxWO', 'DQ81', 'LGvF', 'GGkA', 'IsPK', 'VFH1', 'Skbi', 'jQjI', 'VFRZ',
'SGkC', 'UjXP', '3dKJ', 'ZFXz', 'StMQ', 'nldR', 'yFqD', '4yov', 'yptt',
'ZfUi', 'A7Ii', 'XG08', 'dxXL', 't6XK', 'DJIp', 'Qgx8', 'AvzM', '1M8N',
'ad7X', 'I4FM', '0zzw', '7i85', 'jaYh', 'oSdt', 'L0rO', 'MZBv', 'tnpd',
'5mu0', 'lP7a']
```

17) Comparaison méthodes

```
temps_recherche = [
np.array([7.48408079, 37.04957628, 16.94084716, 11.42316318, 16.38663292,
       28.08847308, 33.91678691, 40.57834911, 10.40874481, 17.05675411,
       29.05584073, 39.17067099, 33.03121591, 12.8329339 , 27.13533998,
       21.43732309, 39.96533799, 14.88723397, 25.19401789, 21.72123194,
       41.89045   , 27.12084198, 35.6866219 , 29.95848989,  2.21866322,
        2.42368412, 11.30582213, 28.95326209, 31.03416491,  0.68216491,
        8.09176302,  5.65447497, 33.97140479, 33.38341713, 14.1082418 ,
       40.37034583, 24.92904925,  6.21411896, 42.01368237,  0.54065776,
       10.26698494, 15.58138108, 32.10026526,  7.52308822, 11.79262781,
        0.59974289, 41.3691721 , 15.51225495, 12.59202504, 12.38531995,
       16.68569589, 23.81572604, 35.07833886, 17.41827679, 37.21420383,
       18.57530904, 23.12404203, 13.33703375, 17.67671514, 11.983881  ,
       37.39547825, 27.86402988, 38.18298793, 24.10632396, 26.18967986,
       22.92229033, 25.57449293, 32.92228198, 30.75970507,  7.17288709,
       33.15767884, 31.070894  , 32.15044308, 45.07487488, 42.41560507,
       35.80665898, 10.65250707, 19.4522717 , 39.17097592, 16.39935398,
       34.40831184, 18.22198701, 33.51514816,  2.2523098 , 14.3225522 ,
       21.0275538 , 29.86340714, 18.48089695, 36.99392414,  0.06312013,
       29.88530993, 43.45615697, 49.69982386,  7.75746512, 12.18912411,
       31.76574111, 32.43323803, 16.18395996, 45.54383802,  8.14673924]),
    np.array([1.43245602, 1.43737197, 1.41070509, 1.46778107, 1.40777707,
       1.3904829 , 1.43355584, 1.40450001, 1.38287401, 1.39033318,
       1.42546201, 1.39034486, 1.39766598, 1.40922594, 1.39569521,
       1.34461689, 1.33362579, 1.37518096, 1.34968925, 1.37745833,
       1.35476112, 1.34682608, 1.33327103, 1.37440205, 1.33617997,
       1.38018131, 1.34323978, 1.33002877, 1.36886406, 1.34010768,
       1.32396793, 1.32001114, 1.36882305, 1.31236792, 1.33002901,
       1.35186481, 1.34340978, 1.35472226, 1.38009501, 1.38117504,
       1.33095598, 1.32085609, 1.35105205, 1.36898112, 1.34605813,
       1.34290409, 1.36824703, 1.33355212, 1.36715984, 1.32673573,
       1.37318397, 1.32694292, 1.32756996, 1.42543697, 1.41018009,
       1.43014503, 1.78435302, 1.78444219, 1.89947891, 1.57158399,
       1.64389896, 1.99729896, 1.6955781 , 1.85767388, 1.43537593,
       1.42075586, 6.42162204, 1.76911592, 1.42410994, 1.9610889 ,
       1.46152186, 1.38136292, 1.39121985, 1.66234398, 1.85984802,
       1.86130381, 1.74134111, 1.65289617, 1.46193886, 1.35255599,
       1.33797669, 1.32899284, 1.32318473, 1.31750798, 1.44033384,
       1.44705677, 2.07856107, 1.42398524, 1.36390591, 1.65233302,
       1.71778083, 1.61293101, 1.90185094, 1.74857306, 2.03505778,
       1.92666006, 1.87554693, 1.92644906, 1.75284004, 1.38193488]),
    np.array([36.51550007,  9.37046194, 31.90311027, 12.13274908,
        35.15078998, 43.85159993, 25.89137197, 13.95896602,
        31.78128409, 25.9654119 , 28.49303079, 23.16991782,
        54.59172797, 45.76273513,  9.64292383, 34.01576591,
        13.23105383, 36.69477797, 30.72434592,  9.38007283,
        36.83163714, 12.61273193, 18.04264975, 18.89306784,
         9.45942783, 28.6610539 , 42.93341494, 44.12296796,
        44.94183803, 35.17541289, 16.74342299, 24.91473794,
        35.52758503, 21.40189314,  7.13093591, 13.94211102,
        32.62125778, 14.7743392 , 50.37439919, 23.00323391,
        14.23680687, 49.76016593, 22.96452308, 10.36806202,
        49.17733121, 55.84600186, 17.29855204, 22.87439895,
        19.17474127, 29.47931099, 32.34591699, 24.70991731,
        22.59567595, 16.08691883, 26.60836673, 37.81304979,
        25.136621  , 33.79707479, 14.1265521 , 36.21914101,
        37.77051425, 39.25299883, 35.74921131, 24.71619105,
        32.81830502, 278.16359115, 11.14688587, 32.96459389,
        32.49235916, 23.12126017, 31.98077011,  8.15811419,
       364.69406104, 35.66222596, 10.27587414, 59.5743432 ,
         8.59454417, 16.334728  , 38.48093486, 43.08903289,
        12.65628171, 45.97774696, 28.63385797, 19.32925916,
        30.23185611, 33.89700198, 15.65830922, 36.18696809,
```

17) Comparaison méthodes

```
          18.00773406,   7.74706984,  34.23829389,  56.03851914,
          19.2253871 ,  35.52157974,  38.9081068 ,  59.23415518,
          46.97769499,   8.2677331 ,  16.35617709,  46.43442106]),
     np.array([2.22605419,  5.64695001,  3.26801991,  0.3662858 ,
7.28929615,
         10.10549903, 10.15042925,  6.83593702,  4.31013036,  3.20083284,
          2.50505614, 22.48185325,  5.01994896,  1.05129385,  0.44761705,
         16.61516094,  0.11037087, 15.84094   ,  8.55689073, 21.51310325,
          2.44440603, 12.91054606, 17.99175   ,  9.85523582,  1.12190175,
          3.04914284,  4.857162  ,  9.22664595,  0.13038778, 21.76274896,
         14.03879333,  1.47834921, 20.12752414,  3.58041596, 16.66767621,
          4.85608888, 17.40743399,  2.55598116,  1.12551427,  1.3454771 ,
         15.87514329,  8.25224209, 14.92631817, 11.7137742 ,  6.44177914,
         14.80449605, 17.39310312,  5.14437914, 17.08847308,  3.96504712,
          9.33138418, 12.65140295,  2.1654191 ,  7.59183311,  1.42565775,
         20.59430814, 11.61556506,  0.21904421,  3.57400775,  1.16023898,
          4.06094503,  1.50145316,  2.25377464,  9.92789292,  2.28378797,
          1.98066711,  0.4639039 , 21.60963726,  9.30563307,  3.72771096,
          9.29973483,  7.71987271,  1.33492589,  7.08227706,  6.78416777,
         26.43807316, 27.19942403,  1.31878495,  8.99495792, 11.73372293,
          1.47233605,  0.1953249 , 13.18100786,  2.23198485,  7.98771882,
          7.53292418,  5.58712602,  1.77688694,  0.12983799,  3.66454029,
          2.398314  , 14.281569  ,  6.75143385,  1.99435997, 10.86505485,
          0.25706911, 11.64684796,  4.80289316,  9.80327177,  6.08203793])]
```

17) Application

1) MainActivity.java :

```java
package fr.juliette.thecode;
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.content.ClipboardManager;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.app.AppCompatDelegate;
import android.text.Editable;
import android.text.SpannableString;
import android.text.TextWatcher;
import android.text.method.LinkMovementMethod;
import android.text.util.Linkify;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.SeekBar;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.security.MessageDigest;
public class MainActivity extends AppCompatActivity {
    // MARK : Introduction des connexions
    TextView securiteTextView;
    TextView longueurTextView;
    EditText siteEditText;
    EditText clefEditText;
    EditText motPasseEditText;
    Button questionButton;
    Button copierButton;
    Switch minSwitch;
    Switch majSwitch;
    Switch symSwitch;
    Switch chiSwitch;
    SeekBar longueurSeekBar;
    SeekBar securiteSeekBar;
    private String fileName = "modeSombre.txt";
    @SuppressLint({"SetTextI18n", "ResourceType"})
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // MARK : Connexion avec élément graphiques
        securiteTextView = findViewById(R.id.securiteTextView);
        longueurTextView = findViewById(R.id.longueurTextView);
        clefEditText = findViewById(R.id.clefEditText);
        siteEditText = findViewById(R.id.siteEditText);
        motPasseEditText = findViewById(R.id.motPasseEditText);
        questionButton = findViewById(R.id.questionButton);
        copierButton = findViewById(R.id.copierButton);
        minSwitch = findViewById(R.id.minSwitch);
        majSwitch = findViewById(R.id.majSwitch);
```

```
65          symSwitch = findViewById(R.id.symSwitch);
66          chiSwitch = findViewById(R.id.chiSwitch);
67          securiteSeekBar = findViewById(R.id.securiteSeekBar);
68          longueurSeekBar = findViewById(R.id.longueurSeekBar);
69          // MARK : Connexion avec fonctions
70          clefEditText.addTextChangedListener(textWatcher);
71          siteEditText.addTextChangedListener(textWatcher);
72          longueurSeekBar.setOnSeekBarChangeListener(longueurListener);
73          securiteSeekBar.setOnSeekBarChangeListener(securiteListener);
74          //clefEditText.setInputType(InputType.TYPE_CLASS_TEXT |
75  InputType.TYPE_TEXT_VARIATION_PASSWORD);
76          lancer();
77      }
78
79      // MARK : Initialisations
80      private String base = "";
81      private static boolean darkMode;
82      // MARK : Fonctions
83      private void lancer() {
84          File mFile = new File(Environment.getExternalStorageDirectory().getPath() +
85  "/Android/data/" + getPackageName() + "/files/" + fileName);
86          String text = read(mFile);
87          int actualMode = AppCompatDelegate.getDefaultNightMode(); // 1 : Mode Sombre,
88  2 : Mode clair
89
90          darkMode = "DARK".equals(text);
91          if (darkMode == (actualMode == 2)){
92              setDarkMode();
93          }
94      }
95      private String setDarkMode() {
96          File mFile = new File(Environment.getExternalStorageDirectory().getPath() +
97  "/Android/data/" + getPackageName() + "/files/" + fileName);
98          String message;
99          if (!darkMode){
100             AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
101             message = "Mode clair activé";
102             write("LIGHT", mFile);
103         }
104         else {
105             AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
106             message = "Mode sombre activé";
107             write("DARK", mFile);
108         }
109         startActivity(new Intent(getApplicationContext(), MainActivity.class));
110         finish();
111         return message;
112     }
113     private void write(String text, File mFile) {
114         try {
115             // Flux interne
116             FileOutputStream output = openFileOutput(fileName, MODE_PRIVATE);
117             // On écrit dans le flux interne
118             output.write(text.getBytes());
119             if(output != null)
120                 output.close();
121             // Si le fichier est lisible et qu'on peut écrire dedans
122             if(Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())
123                     && !
124 Environment.MEDIA_MOUNTED_READ_ONLY.equals(Environment.getExternalStorageState())) {
```

```
125                     // On crée un nouveau fichier. Si le fichier existe déjà, il ne sera
126 pas créé
127                     mFile.createNewFile();
128                     output = new FileOutputStream(mFile);
129                     String darkName = "DARK";
130                     output.write(darkName.getBytes());
131                     if(output != null)
132                         output.close();
133                 }
134             }
135             catch (FileNotFoundException e) {
136                 e.printStackTrace();
137             } catch (IOException e) {
138                 e.printStackTrace();
139             }
140         }
141         private String read(File mFile) {
142             String text = "";
143             try{
144                 FileInputStream input = openFileInput(fileName);
145                 int value;
146                 // On utilise un StringBuffer pour construire la chaîne au fur et à mesure
147                 StringBuffer lu = new StringBuffer();
148                 // On lit les caractères les uns après les autres
149                 while ((value = input.read()) != -1) {
150                     // On écrit dans le fichier le caractère lu
151                     lu.append((char) value);
152                 }
153                 text = lu.toString();
154                 if (input != null)
155                     input.close();
156                 if
157 (Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())) {
158                     lu = new StringBuffer();
159                     try {
160                         input = new FileInputStream(mFile);
161                     } catch (FileNotFoundException ex) {
162                         ex.printStackTrace();
163                     }
164                     while ((value = input.read()) != -1)
165                         lu.append((char) value);
166                     text = lu.toString();
167                     if (input != null)
168                         input.close();
169                 }
170             }catch (FileNotFoundException e) {
171                 e.printStackTrace();
172             } catch (IOException e) {
173                 e.printStackTrace();
174             }
175             return text;
176         }
177         private void copier(java.lang.String mot) {
178             // Copie le mot dans le presse papier
179             if (mot != null) {
180                 ClipboardManager clipboard = (ClipboardManager)
181 getSystemService(CLIPBOARD_SERVICE);
182                 assert clipboard != null;
183                 android.content.ClipData clip = android.content.ClipData.newPlainText("Mot
184 de passe", mot);
185                 clipboard.setPrimaryClip(clip);
```

```
186             }
187         }
188         private static String dec2Base(BigInteger x, String base) {
189             // Convertit un BigInteger dans une base ayant base comme support
190             BigInteger b = new BigInteger(String.valueOf(base.length()));
191             StringBuilder result = new StringBuilder();
192             BigInteger zero = new BigInteger("0");
193             if (x.equals(zero)) {
194                 result = new StringBuilder(base.charAt(0));
195             } else {
196                 while (!x.equals(zero)) {
197                     int inter = x.mod(b).intValue();
198                     result.insert(0, base.charAt(inter));
199                     x = x.divide(b);
200                 }
201             }
202             return result.toString();
203         }
204         @SuppressLint("SetTextI18n")
205         private void generer() {
206             // Génère le mot de passe
207             modifBase();
208             motPasseEditText.setText("Il manque des valeurs");
209             if (clefEditText.getText().toString().length() == 0 ||
210     siteEditText.getText().toString().length() == 0) {
211                 // Rien dans site ou dans clef
212             } else if (!(minSwitch.isChecked() || majSwitch.isChecked() ||
213     chiSwitch.isChecked() || symSwitch.isChecked())) {
214                 // Aucun checkbuttons cliqués : Toast pour prévenir
215                 Toast.makeText(MainActivity.this, "Aucun type de caractères n'est coché",
216     Toast.LENGTH_LONG).show();
217             } else {
218                 // On peut générer le mot de passe
219                 String clef = clefEditText.getText().toString();
220                 String site = siteEditText.getText().toString();
221                 String[] result = modification(site + clef);
222                 motPasseEditText.setText(result[0]);
223             }
224             modifSecurite();
225         }
226         private void modifBase() {
227             // Modifie la base suivant les caractères cochés
228             base = "";
229             if (minSwitch.isChecked()) {
230                 base += "portezcviuxwhskyajgblndqfm";
231             }
232             if (majSwitch.isChecked()) {
233                 base += "THEQUICKBROWNFXJMPSVLAZYDG";
234             }
235             if (symSwitch.isChecked()) {
236                 base += "@#&!)-%;<:*$+=/?>(";
237             }
238             if (chiSwitch.isChecked()) {
239                 base += "567438921";
240             }
241         }
242         @SuppressLint("SetTextI18n")
243         private void modifSecurite() {
244             // Modifie la sécurité en fonction des paramètres cochés
245             int len2 = longueurSeekBar.getProgress();
246             int len = len2 * len2 + 3 * len2 + 10;
```

```
247             longueurTextView.setText("Longueur : " + len);
248             int nb_carac = base.length();
249             int bits = (int) ((Math.round(Math.log(Math.pow(nb_carac, len)) /
250 Math.log(2))));
251             if (bits == 0) {
252                 securiteSeekBar.setProgress(bits);
253             } else {
254                 securiteSeekBar.setProgress(bits - 32);
255             }
256             String[] result = securite(bits);
257             securiteTextView.setText(result[0] + bits + " bits");
258             securiteTextView.setTextColor(Color.parseColor(result[1]));
259         }
260         private String[] modification(String mot) {
261             // Modifie le site et la clef en un mot de passe (mot = site + clef)
262             int len = longueurSeekBar.getProgress();
263             int len2 = len * len + 3 * len + 10;
264             BigInteger code = new BigInteger(sha256(mot), 16);
265             int nb_carac = base.length();
266             String code2 = dec2Base(code, base).substring(0, len2);
267             int bits = (int) ((Math.round(Math.log(Math.pow(nb_carac, code2.length())) /
268 Math.log(2))));
269             String[] result = securite(bits);
270             if (bits == 0) {
271                 code2 = "";
272             }
273             return new String[]{code2, result[0] + bits + " bits", Integer.toString(bits),
274 result[1]};
275         }
276         private String[] securite(int bits) {
277             // Renvoie la bonne couleur ainsi que la sécurité suivant le nombre de bits
278             String secure;
279             String color;
280             if (bits == 0) {
281                 secure = " Aucune ";
282                 color = "#FE0101";
283             } else if (bits < 64) {
284                 secure = " Très Faible ";
285                 color = "#FE0101";
286             } else if (bits < 80) {
287                 secure = " Faible ";
288                 color = "#FE4501";
289             } else if (bits < 100) {
290                 secure = " Moyenne ";
291                 color = "#FE7601";
292             } else if (bits < 126) {
293                 secure = " Forte ";
294                 color = "#53FE38";
295             } else {
296                 secure = " Très Forte ";
297                 color = "#1CD001";
298             }
299             return new String[]{secure, color};
300         }
301         private static String sha256(String mot) {
302             // Modifie mot en un chiffre en hexadécimal suivant la fonction de hachage
303 sha256
304             try {
305                 MessageDigest digest = MessageDigest.getInstance("SHA-256");
306                 byte[] hash = digest.digest(mot.getBytes("UTF-8"));
307                 StringBuilder hexString = new StringBuilder();
```

```
308                 for (byte b : hash) {
309                     String hex = Integer.toHexString(0xff & b);
310                     if (hex.length() == 1) hexString.append('0');
311                     hexString.append(hex);
312                 }
313                 return hexString.toString();
314             } catch (Exception ex) {
315                 throw new RuntimeException(ex);
316             }
317         }
318         // MARK : Actions
319         @SuppressLint({"SetTextI18n", "ResourceAsColor", "ResourceType"})
320         public void questionChange(View view) {
321             // TextView Message Informatif
322             final TextView message = new TextView(MainActivity.this);
323             message.setText(getString(R.string.info_questions));
324             message.setTextSize(15);
325             message.setTextColor(Color.parseColor(getString(R.color.colorText)));
326             message.setPadding(50, 0, 50, 0);
327             // Questions dans les CheckButtons
328             String[] questions = {"nom de jeune fille de votre mère", "nom de votre premier
329 animal de compagnie",
330                     "rue de votre maison d'enfance", "pas de question"};
331             final boolean[] checkedItems = {false, false, false, false};
332             // Création alert box
333             AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this,
334 R.style.AlertDialogCustom)
335                     .setView(message) // Connexion avec message
336                     .setTitle("Question personnelle :") // Titre
337                     .setNegativeButton("Annuler", null) // Bouton Annuler
338                     .setMultiChoiceItems(questions, checkedItems, new
339 DialogInterface.OnMultiChoiceClickListener() {
340                         @Override
341                         public void onClick(DialogInterface dialog, int which, boolean
342 isChecked) {
343                             if (checkedItems[0]) {
344                                 clefEditText.setHint("nom jeune fille mère");
345                                 clefEditText.setText("");
346                                 dialog.cancel();
347                             } else if (checkedItems[1]) {
348                                 clefEditText.setHint("nom premier animal de compagnie");
349                                 clefEditText.setText("");
350                                 dialog.cancel();
351                             } else if (checkedItems[2]) {
352                                 clefEditText.setHint("rue maison enfance");
353                                 clefEditText.setText("");
354                                 dialog.cancel();
355                             } else if (checkedItems[3]) {
356                                 clefEditText.setHint("mot clef");
357                                 clefEditText.setText("");
358                                 dialog.cancel();
359                             }
360                         }
361                     });
362             builder.show(); // Monter Alert box
363         }
364         public void copierChange(View view) {
365             java.lang.String code = motPasseEditText.getText().toString();
366             if (!(code.length() == 0) && !(code.equals("Il manque des valeurs"))) {
367                 copier(code);
```

```
368                 Toast.makeText(MainActivity.this, "Mot de passe copié dans le presse-
369 papier", Toast.LENGTH_LONG).show();
370             } else {
371                 Toast.makeText(MainActivity.this, "Vous n'avez aucun mot de passe à
372 copier", Toast.LENGTH_LONG).show();
373             }
374         }
375         public void checkChange(View view) {
376             // Switch Changé
377             generer();
378         }
379         private TextWatcher textWatcher = new TextWatcher() {
380             //Text Watch : génère le mot de passe quand on a fini de modifier le texte
381             @Override
382             public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2)
383 {
384             }
385             @Override
386             public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
387             }
388             @Override
389             public void afterTextChanged(Editable editable) {
390                 generer();
391             }
392         };
393         private SeekBar.OnSeekBarChangeListener longueurListener = new
394 SeekBar.OnSeekBarChangeListener() {
395             @SuppressLint("SetTextI18n")
396             @Override
397             public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
398                 // Longueur Slider en changement
399                 generer();
400             }
401             @Override
402             public void onStartTrackingTouch(SeekBar seekBar) {
403             }
404             @Override
405             public void onStopTrackingTouch(SeekBar longueur) {
406             }
407         };
408         private SeekBar.OnSeekBarChangeListener securiteListener = new
409 SeekBar.OnSeekBarChangeListener() {
410             @SuppressLint("SetTextI18n")
411             @Override
412             public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
413                 // Sécurité Slider en changement
414                 int bits = securiteSeekBar.getProgress() + 32;
415                 String[] result = securite(bits);
416                 securiteTextView.setText(result[0] + bits + " bits");
417                 securiteTextView.setTextColor(Color.parseColor(result[1]));
418             }
419             @Override
420             public void onStartTrackingTouch(SeekBar seekBar) {
421             }
422             @SuppressLint("SetTextI18n")
423             @Override
424             public void onStopTrackingTouch(SeekBar seekBar) {
425                 // Sécurité Slider une fois changée
426                 int bits = securiteSeekBar.getProgress() + 32;
427                 if (bits < 42) {
428                     longueurSeekBar.setProgress(0);
```

```
429                 minSwitch.setChecked(false);
430                 majSwitch.setChecked(false);
431                 symSwitch.setChecked(false);
432                 chiSwitch.setChecked(true);
433             } else if (bits < 47) {
434                 longueurSeekBar.setProgress(0);
435                 minSwitch.setChecked(false);
436                 majSwitch.setChecked(false);
437                 symSwitch.setChecked(true);
438                 chiSwitch.setChecked(false);
439             } else if (bits < 48) {
440                 longueurSeekBar.setProgress(0);
441                 minSwitch.setChecked(true);
442                 majSwitch.setChecked(false);
443                 symSwitch.setChecked(false);
444                 chiSwitch.setChecked(false);
445             } else if (bits < 51) {
446                 longueurSeekBar.setProgress(0);
447                 minSwitch.setChecked(false);
448                 majSwitch.setChecked(false);
449                 symSwitch.setChecked(true);
450                 chiSwitch.setChecked(true);
451             } else if (bits < 55) {
452                 longueurSeekBar.setProgress(0);
453                 minSwitch.setChecked(true);
454                 majSwitch.setChecked(false);
455                 symSwitch.setChecked(false);
456                 chiSwitch.setChecked(true);
457             } else if (bits < 57) {
458                 longueurSeekBar.setProgress(0);
459                 minSwitch.setChecked(true);
460                 majSwitch.setChecked(false);
461                 symSwitch.setChecked(true);
462                 chiSwitch.setChecked(false);
463             } else if (bits < 61) {
464                 longueurSeekBar.setProgress(0);
465                 minSwitch.setChecked(true);
466                 majSwitch.setChecked(true);
467                 symSwitch.setChecked(false);
468                 chiSwitch.setChecked(false);
469             } else if (bits < 63) {
470                 longueurSeekBar.setProgress(0);
471                 minSwitch.setChecked(true);
472                 majSwitch.setChecked(true);
473                 symSwitch.setChecked(true);
474                 chiSwitch.setChecked(false);
475             } else if (bits < 66) {
476                 longueurSeekBar.setProgress(0);
477                 minSwitch.setChecked(true);
478                 majSwitch.setChecked(true);
479                 symSwitch.setChecked(true);
480                 chiSwitch.setChecked(true);
481             } else if (bits < 67) {
482                 longueurSeekBar.setProgress(1);
483                 minSwitch.setChecked(true);
484                 majSwitch.setChecked(false);
485                 symSwitch.setChecked(false);
486                 chiSwitch.setChecked(false);
487             } else if (bits < 72) {
488                 longueurSeekBar.setProgress(1);
489                 minSwitch.setChecked(false);
```

```
490                 majSwitch.setChecked(false);
491                 symSwitch.setChecked(true);
492                 chiSwitch.setChecked(true);
493             } else if (bits < 76) {
494                 longueurSeekBar.setProgress(1);
495                 minSwitch.setChecked(true);
496                 majSwitch.setChecked(false);
497                 symSwitch.setChecked(false);
498                 chiSwitch.setChecked(true);
499             } else if (bits < 80) {
500                 longueurSeekBar.setProgress(1);
501                 minSwitch.setChecked(true);
502                 majSwitch.setChecked(false);
503                 symSwitch.setChecked(true);
504                 chiSwitch.setChecked(false);
505             } else if (bits < 86) {
506                 longueurSeekBar.setProgress(1);
507                 minSwitch.setChecked(true);
508                 majSwitch.setChecked(true);
509                 symSwitch.setChecked(false);
510                 chiSwitch.setChecked(false);
511             } else if (bits < 88) {
512                 longueurSeekBar.setProgress(1);
513                 minSwitch.setChecked(true);
514                 majSwitch.setChecked(true);
515                 symSwitch.setChecked(true);
516                 chiSwitch.setChecked(false);
517             } else if (bits < 94) {
518                 longueurSeekBar.setProgress(1);
519                 minSwitch.setChecked(true);
520                 majSwitch.setChecked(true);
521                 symSwitch.setChecked(true);
522                 chiSwitch.setChecked(true);
523             } else if (bits < 95) {
524                 longueurSeekBar.setProgress(2);
525                 minSwitch.setChecked(true);
526                 majSwitch.setChecked(false);
527                 symSwitch.setChecked(false);
528                 chiSwitch.setChecked(false);
529             } else if (bits < 103) {
530                 longueurSeekBar.setProgress(2);
531                 minSwitch.setChecked(false);
532                 majSwitch.setChecked(false);
533                 symSwitch.setChecked(true);
534                 chiSwitch.setChecked(true);
535             } else if (bits < 109) {
536                 longueurSeekBar.setProgress(2);
537                 minSwitch.setChecked(true);
538                 majSwitch.setChecked(false);
539                 symSwitch.setChecked(false);
540                 chiSwitch.setChecked(true);
541             } else if (bits < 114) {
542                 longueurSeekBar.setProgress(2);
543                 minSwitch.setChecked(true);
544                 majSwitch.setChecked(false);
545                 symSwitch.setChecked(true);
546                 chiSwitch.setChecked(false);
547             } else if (bits < 115) {
548                 longueurSeekBar.setProgress(2);
549                 minSwitch.setChecked(true);
550                 majSwitch.setChecked(true);
```

```
551                     symSwitch.setChecked(false);
552                     chiSwitch.setChecked(false);
553                 } else if (bits < 123) {
554                     longueurSeekBar.setProgress(2);
555                     minSwitch.setChecked(true);
556                     majSwitch.setChecked(false);
557                     symSwitch.setChecked(true);
558                     chiSwitch.setChecked(true);
559                 } else if (bits < 126) {
560                     longueurSeekBar.setProgress(2);
561                     minSwitch.setChecked(true);
562                     majSwitch.setChecked(true);
563                     symSwitch.setChecked(true);
564                     chiSwitch.setChecked(false);
565                 } else {
566                     longueurSeekBar.setProgress(2);
567                     minSwitch.setChecked(true);
568                     majSwitch.setChecked(true);
569                     symSwitch.setChecked(true);
570                     chiSwitch.setChecked(true);
571                 }
572                 generer();
573             }
574         };
575         @Override
576         public boolean onCreateOptionsMenu(Menu menu) {
577             // Création menu
578             super.onCreateOptionsMenu(menu);
579             getMenuInflater().inflate(R.menu.menu_main, menu);
580             if (darkMode){
581                 // Item light
582                 menu.findItem(R.id.darkmode).setIcon(R.drawable.light);
583             }
584             else {
585                 // Item night
586                 menu.findItem(R.id.darkmode).setIcon(R.drawable.night);
587             }
588             return true;
589         }
590         @SuppressLint("Assert")
591         @Override
592         public boolean onOptionsItemSelected(MenuItem item) {
593             // Action boutons menu
594             switch (item.getItemId()) {
595                 case R.id.darkmode:
596                     // DarkMode
597                     darkMode = !darkMode;
598                     String message  = setDarkMode();
599                     //Toast.makeText(MainActivity.this, message, Toast.LENGTH_LONG).show();
600                     break;
601                 case R.id.aide:
602                     // Aide lorsque bouton help est pressé
603                     final SpannableString s = new
604 SpannableString(getString(R.string.info_app));
605                     Linkify.addLinks(s, Linkify.ALL);
606                     final AlertDialog d = new AlertDialog.Builder(this,
607 R.style.AlertDialogCustom)
608                             .setTitle("Informations")
609                             .setPositiveButton(android.R.string.ok, null)
610                             .setMessage(s)
611                             .create();
```

```
                    d.show();
                    ((TextView)
d.findViewById(android.R.id.message)).setMovementMethod(LinkMovementMethod.getInstance()
);
                    break;
                case R.id.partager:
                    // Partage le mot de passe lorsque shareButton est pressé
                    java.lang.String code = motPasseEditText.getText().toString();
                    if (!(code.length() == 0) && !(code.equals("Il manque des valeurs"))) {
                        java.lang.String site = siteEditText.getText().toString();
                        Intent share = new Intent(android.content.Intent.ACTION_SEND);
                        share.setType("text/plain");
                        share.putExtra(Intent.EXTRA_TEXT, "Mon mot de passe pour " + site
+ " est :\n" + code);
                        startActivity(Intent.createChooser(share, "Mot de passe"));
                    } else {
                        Toast.makeText(MainActivity.this, "Vous n'avez aucun mot de passe
à partager", Toast.LENGTH_LONG).show();
                    }
                    break;
            }
            return super.onOptionsItemSelected(item);
        }
}
```

2) activity_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimaryDark"
    android:clickable="true"
    android:clipChildren="false"
    android:orientation="vertical"
    android:scrollbarStyle="insideOverlay"
    android:focusable="true">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:orientation="vertical">
        <RadioGroup
            android:id="@+id/questionGroup"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:layout_marginTop="15dp"
            android:checkedButton="@+id/radio2"
            android:orientation="horizontal">
        <Button
            android:id="@+id/questionButton"
            android:layout_width="wrap_content"
            android:layout_height="30dp"
            android:background="@drawable/button"
            android:textAllCaps="false"
            android:text="Question personnelle"
            android:paddingRight="10dp"
            android:paddingLeft="10dp"
```

```
672             android:textColor="@color/colorPrimaryDark"
673             android:onClick="questionChange"
674             android:textSize="20sp" />
675         </RadioGroup>
676         <RadioGroup
677             android:id="@+id/groupClef"
678             android:layout_width="fill_parent"
679             android:layout_height="wrap_content"
680             android:layout_marginStart="20dp"
681             android:layout_marginLeft="20dp"
682             android:layout_marginTop="5dp"
683             android:checkedButton="@+id/radio2"
684             android:orientation="horizontal">
685             <TextView
686                 android:id="@+id/typedecodage"
687                 android:layout_width="wrap_content"
688                 android:layout_height="wrap_content"
689                 android:text="Clef :"
690                 android:textColor="@color/colorText"
691                 android:textSize="20sp" />
692             <EditText
693                 android:id="@+id/clefEditText"
694                 android:layout_width="match_parent"
695                 android:layout_height="wrap_content"
696                 android:layout_marginLeft="10dp"
697                 android:layout_weight="1"
698                 android:layout_marginRight="10dp"
699                 android:autofillHints=""
700                 android:hint="mot clef"
701                 android:maxLines="1"
702                 android:singleLine="true"
703                 android:textColor="@color/colorText"
704                 android:textColorHint="@color/colorText"
705                 android:backgroundTint="@color/gray"
706                 android:textSize="17sp"
707                 android:typeface="serif"
708                 android:inputType="textVisiblePassword"/>
709         </RadioGroup>
710         <RadioGroup
711             android:id="@+id/groupSite"
712             android:layout_width="fill_parent"
713             android:layout_height="wrap_content"
714             android:layout_marginStart="20dp"
715             android:layout_marginLeft="20dp"
716             android:layout_marginTop="5dp"
717             android:checkedButton="@+id/radio2"
718             android:orientation="horizontal">
719             <TextView
720                 android:id="@+id/siteTextView"
721                 android:layout_width="wrap_content"
722                 android:layout_height="wrap_content"
723                 android:text="Nom du site :"
724                 android:textColor="@color/colorText"
725                 android:textSize="20sp" />
726             <EditText
727                 android:id="@+id/siteEditText"
728                 android:layout_width="fill_parent"
729                 android:layout_height="wrap_content"
730                 android:layout_marginLeft="10dp"
731                 android:layout_marginRight="10dp"
732                 android:autofillHints=""
```

```
733                 android:hint="nom du site"
734                 android:maxLines="1"
735                 android:singleLine="true"
736                 android:backgroundTint="@color/gray"
737                 android:textColor="@color/colorText"
738                 android:textColorHint="@color/colorText"
739                 android:textSize="17sp"
740                 android:typeface="serif"
741                 tools:ignore="LabelFor,TextFields" />
742         </RadioGroup>
743         <ImageView
744             android:id="@+id/separate1"
745             android:layout_width="fill_parent"
746             android:layout_height="1dp"
747             android:layout_marginLeft="10dp"
748             android:layout_marginTop="15dp"
749             android:layout_marginRight="10dp"
750             android:background="@color/colorButton"
751             android:gravity="center"
752             android:textSize="20sp" />
753         <TextView
754             android:id="@+id/caracTextView"
755             android:layout_width="fill_parent"
756             android:layout_height="wrap_content"
757             android:layout_marginStart="20dp"
758             android:layout_marginLeft="20dp"
759             android:textStyle="normal|bold"
760             android:layout_marginTop="10dp"
761             android:text="Paramètres du mot de passe :"
762             android:textColor="@color/colorText"
763             android:textSize="20sp" />
764         <RadioGroup
765             android:id="@+id/groupLongueur"
766             android:layout_width="fill_parent"
767             android:layout_height="wrap_content"
768             android:layout_marginStart="20dp"
769             android:layout_marginLeft="20dp"
770             android:layout_marginTop="5dp"
771             android:checkedButton="@+id/radio2"
772             android:orientation="horizontal">
773             <TextView
774                 android:id="@+id/longueurTextView"
775                 android:layout_width="wrap_content"
776                 android:layout_height="wrap_content"
777                 android:text="Longueur : 20"
778                 android:textColor="@color/colorText"
779                 android:textSize="20sp" />
780             <SeekBar
781                 android:id="@+id/longueurSeekBar"
782                 android:theme="@style/TickMarkSeekBar"
783                 android:layout_width="fill_parent"
784                 android:layout_height="fill_parent"
785                 android:maxHeight="100000dp"
786                 android:minHeight="100000dp"
787                 android:focusable="true"
788                 android:thumb="@drawable/thumb"
789                 android:indeterminate="false"
790                 android:max="2"
791                 android:progress="2"
792                 />
793             <!--
```

```
794             android:background="@drawable/button"
795             android:progressBackgroundTint="@color/colorAccent"
796             android:thumb="@drawable/thumb"
797             android:tickMarkTint="@color/colorSecurity"
798             android:tickMark="@drawable/thumb"
799             android:progressTint="@color/colorAccent"
800             -->
801         </RadioGroup>
802         <TextView
803             android:id="@+id/caracteresTextView"
804             android:layout_width="fill_parent"
805             android:layout_height="wrap_content"
806             android:layout_marginStart="20dp"
807             android:layout_marginEnd="20dp"
808             android:layout_marginTop="10dp"
809             android:text="Caractères :"
810             android:textColor="@color/colorText"
811             android:textSize="20sp" />
812         <RadioGroup
813             android:id="@+id/groupMajMin"
814             android:layout_width="fill_parent"
815             android:layout_height="wrap_content"
816             android:layout_marginTop="10dp"
817             android:layout_marginStart="20dp"
818             android:layout_marginEnd="20dp"
819             android:checkedButton="@+id/radio2"
820             android:gravity="center"
821             android:orientation="horizontal">
822             <Switch
823                 android:id="@+id/minSwitch"
824                 android:layout_width="140dp"
825                 android:layout_weight="1"
826                 android:layout_height="20dp"
827                 android:background="#00FFFFFF"
828                 android:checked="true"
829                 android:text="Minuscules"
830                 android:textColor="@color/colorText"
831                 android:textSize="16sp"
832                 android:onClick="checkChange"/>
833             <Switch
834                 android:id="@+id/majSwitch"
835                 android:layout_width="140dp"
836                 android:layout_weight="1"
837                 android:layout_height="20dp"
838                 android:layout_marginStart="15dp"
839                 android:layout_marginLeft="15dp"
840                 android:background="#00FFFFFF"
841                 android:checked="true"
842                 android:text="Majuscules"
843                 android:textColor="@color/colorText"
844                 android:textSize="16sp"
845                 android:onClick="checkChange"/>
846         </RadioGroup>
847         <RadioGroup
848             android:id="@+id/groupSymChi"
849             android:layout_width="fill_parent"
850             android:layout_height="wrap_content"
851             android:layout_marginTop="10dp"
852             android:layout_marginStart="20dp"
853             android:layout_marginEnd="20dp"
854             android:checkedButton="@+id/radio2"
```

```
            android:gravity="center"
            android:orientation="horizontal">
            <Switch
                android:id="@+id/symSwitch"
                android:layout_width="140dp"
                android:layout_weight="1"
                android:layout_height="20dp"
                android:background="#00FFFFFF"
                android:checked="true"
                android:text="Symboles"
                android:textColor="@color/colorText"
                android:textSize="16sp"
                android:onClick="checkChange"/>
            <Switch
                android:id="@+id/chiSwitch"
                android:layout_width="140dp"
                android:layout_weight="1"
                android:layout_height="20dp"
                android:layout_marginStart="15dp"
                android:layout_marginLeft="15dp"
                android:background="#00FFFFFF"
                android:checked="true"
                android:text="Chiffres"
                android:textColor="@color/colorText"
                android:textSize="16sp"
                android:onClick="checkChange"/>
            <!--android:layout_width="130dp"-->
        </RadioGroup>
        <ImageView
            android:id="@+id/separate"
            android:layout_width="fill_parent"
            android:layout_height="1dp"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="15dp"
            android:layout_marginRight="10dp"
            android:background="@color/colorButton"
            android:gravity="center"
            android:textSize="20sp" />
        <TextView
            android:id="@+id/motPasseTextView"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="20dp"
            android:textStyle="normal|bold"
            android:layout_marginLeft="20dp"
            android:layout_marginTop="10dp"
            android:text="Le mot de passe est :"
            android:textColor="@color/colorText"
            android:textSize="20sp" />
        <EditText
            android:id="@+id/motPasseEditText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="30dp"
            android:layout_marginRight="50dp"
            android:focusableInTouchMode="false"
            android:gravity="center"
            android:text=""
            android:textColor="@color/colorText"
            android:textSize="18sp"
            android:typeface="serif"
```

```
916                 android:layout_marginStart="30dp"
917                 android:backgroundTint="@color/gray"
918                 android:digits=""
919                 android:layout_marginEnd="50dp">
920                 <!--android:typeface="monospace"-->
921             </EditText>
922             <RadioGroup
923                 android:id="@+id/groupMotPasse"
924                 android:layout_width="fill_parent"
925                 android:layout_height="wrap_content"
926                 android:layout_marginStart="40dp"
927                 android:layout_marginLeft="30dp"
928                 android:layout_marginTop="-40dp"
929                 android:layout_marginEnd="20dp"
930                 android:layout_marginRight="30dp"
931                 android:checkedButton="@+id/radio2"
932                 android:gravity="right"
933                 android:orientation="horizontal">
934                 <Button
935                     android:id="@+id/copierButton"
936                     android:layout_width="35dp"
937                     android:layout_height="35dp"
938                     android:background="@drawable/clipboard"
939                     android:onClick="copierChange"/>
940             </RadioGroup>
941             <RadioGroup
942                 android:id="@+id/groupSecurite"
943                 android:layout_width="fill_parent"
944                 android:layout_height="wrap_content"
945                 android:layout_marginTop="10dp"
946                 android:checkedButton="@+id/radio2"
947                 android:gravity="center"
948                 android:orientation="horizontal">
949                 <TextView
950                     android:id="@+id/securiteSimple"
951                     android:layout_width="wrap_content"
952                     android:layout_height="wrap_content"
953                     android:text="Sécurite :"
954                     android:textColor="@color/colorText"
955                     android:textSize="20sp" />
956                 <TextView
957                     android:id="@+id/securiteTextView"
958                     android:layout_width="wrap_content"
959                     android:layout_height="wrap_content"
960                     android:text=""
961                     android:textSize="20sp" />
962             </RadioGroup>
963             <SeekBar
964                 android:id="@+id/securiteSeekBar"
965                 android:layout_width="fill_parent"
966                 android:layout_height="wrap_content"
967                 android:layout_marginLeft="50dp"
968                 android:layout_marginTop="10dp"
969                 android:layout_marginRight="50dp"
970                 android:thumb="@drawable/thumb"
971                 android:clickable="true"
972                 android:gravity="center"
973                 android:max="94"
974                 android:progress="94" />
975         </LinearLayout>
976     </ScrollView>
```