

# Déchiffrement de mots de passe : Étude de différentes méthodes.

*Juliette DEBONO – 4897 – 2020/2021*

*Quelles sont les meilleures méthodes  
pour déchiffrer un mot de passe ?*

- **Introduction**
- **Techniques basiques de recherche**
- **Tables de compromis temps mémoire**
- **Comparaison**
- **Conclusion**

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Introduction



Assistance et prévention  
en sécurité numérique



## 10 CONSEILS POUR GÉRER VOS MOTS DE PASSE

mémo

1

Utilisez un mot de passe différent pour chaque service



6

Ne communiquez jamais votre mot de passe à un tiers



2

Utilisez un mot de passe suffisamment long et complexe



7

N'utilisez pas vos mots de passe sur un ordinateur partagé



3

Utilisez un mot de passe impossible à deviner



8

Activez la double authentification lorsque c'est possible



4

Utilisez un gestionnaire de mots de passe



9

Changez les mots de passe par défaut des différents services auxquels vous accédez



5

Changez votre mot de passe au moindre soupçon



10

Choisissez un mot de passe particulièrement robuste pour votre messagerie



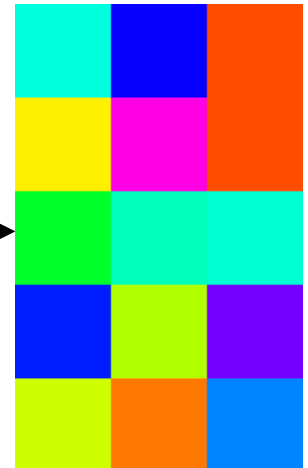
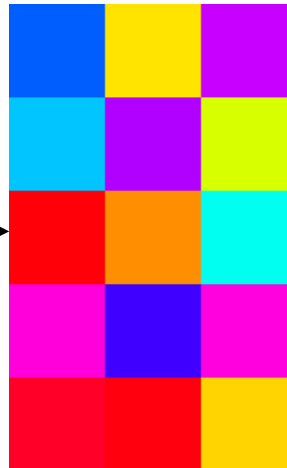
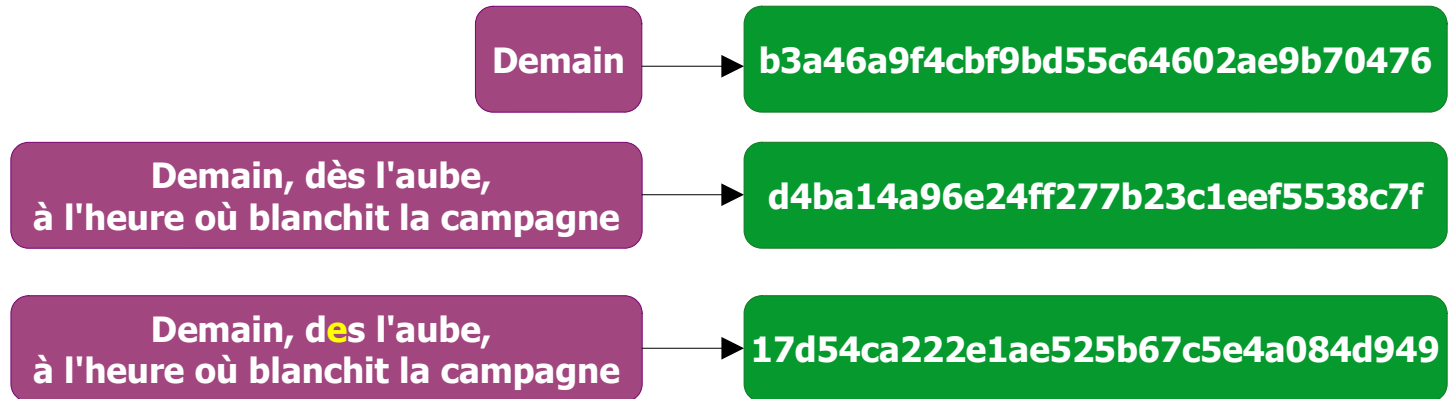
Pour en savoir plus ou vous faire assister, rendez-vous sur [Cybermalveillance.gouv.fr](https://cybermalveillance.gouv.fr)

ADOPTER LES BONNES PRATIQUES

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Hachage

## Hachage par MD5



- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Utilisation du hachage

Base de données du site web

Identifiants	Mot de passe haché
Juliette	83ea007bfdd589f29b820552b3f94260
Utilisateur 1	0cc175b9c0f1b6a831c399e269772661
Utilisateur 2	92eb5ffee6ae2fec3ad71c777531578f

AZERTY

MD5

83ea007bfdd589f29b820552b3f94260

83ea007bfdd589f29b820552b3f94260

c3981fa8d26e95d911fe8eae6570f2f

LOGIN

Identifiant :

Juliette

Mot de passe :

AZERTY

= Bon mot de passe

≠ Mauvais mot de passe

MD5

QWERTY

LOGIN

Identifiant :

Juliette

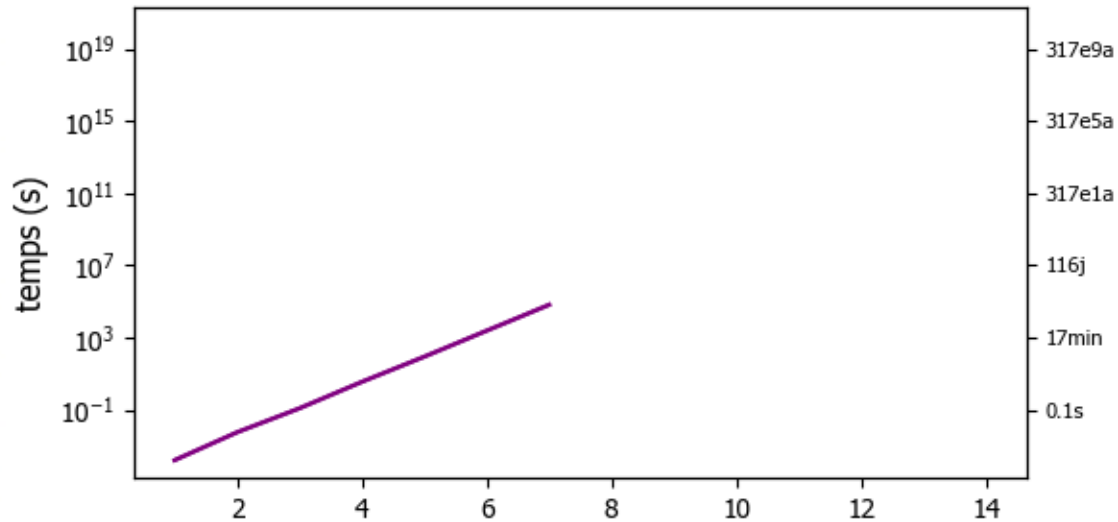
Mot de passe :

QWERTY

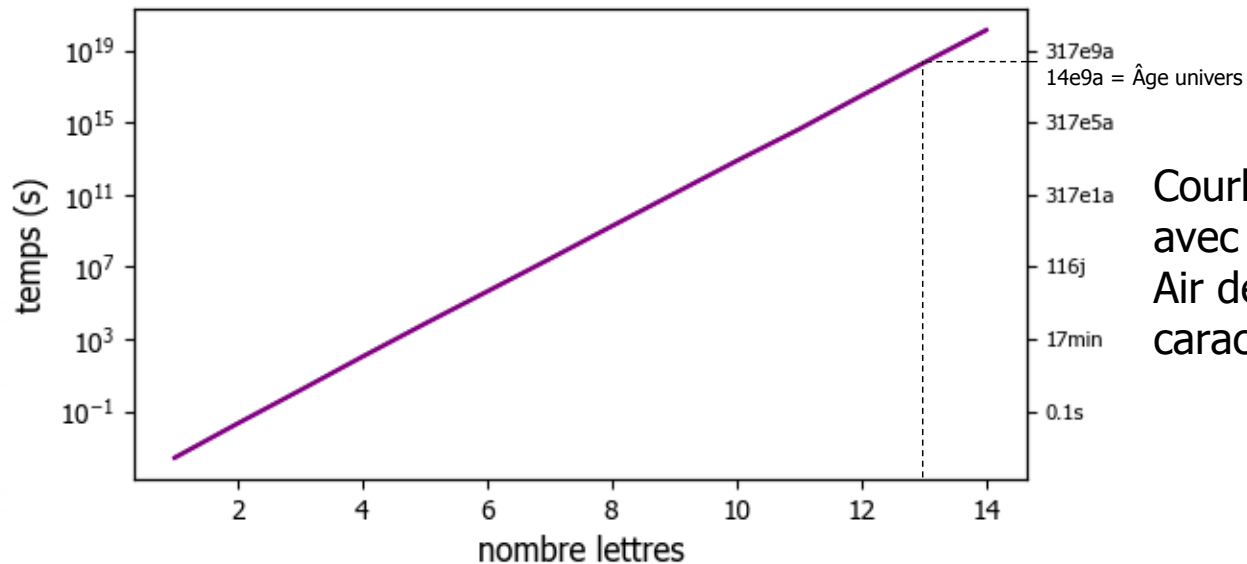
- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Force Brute

Temps pour générer tous les mots de n lettres



Courbe expérimentale avec un iMac de 2009 et 26 caractères



Courbe théorique avec un MacBook Air de 2017 et 62 caractères

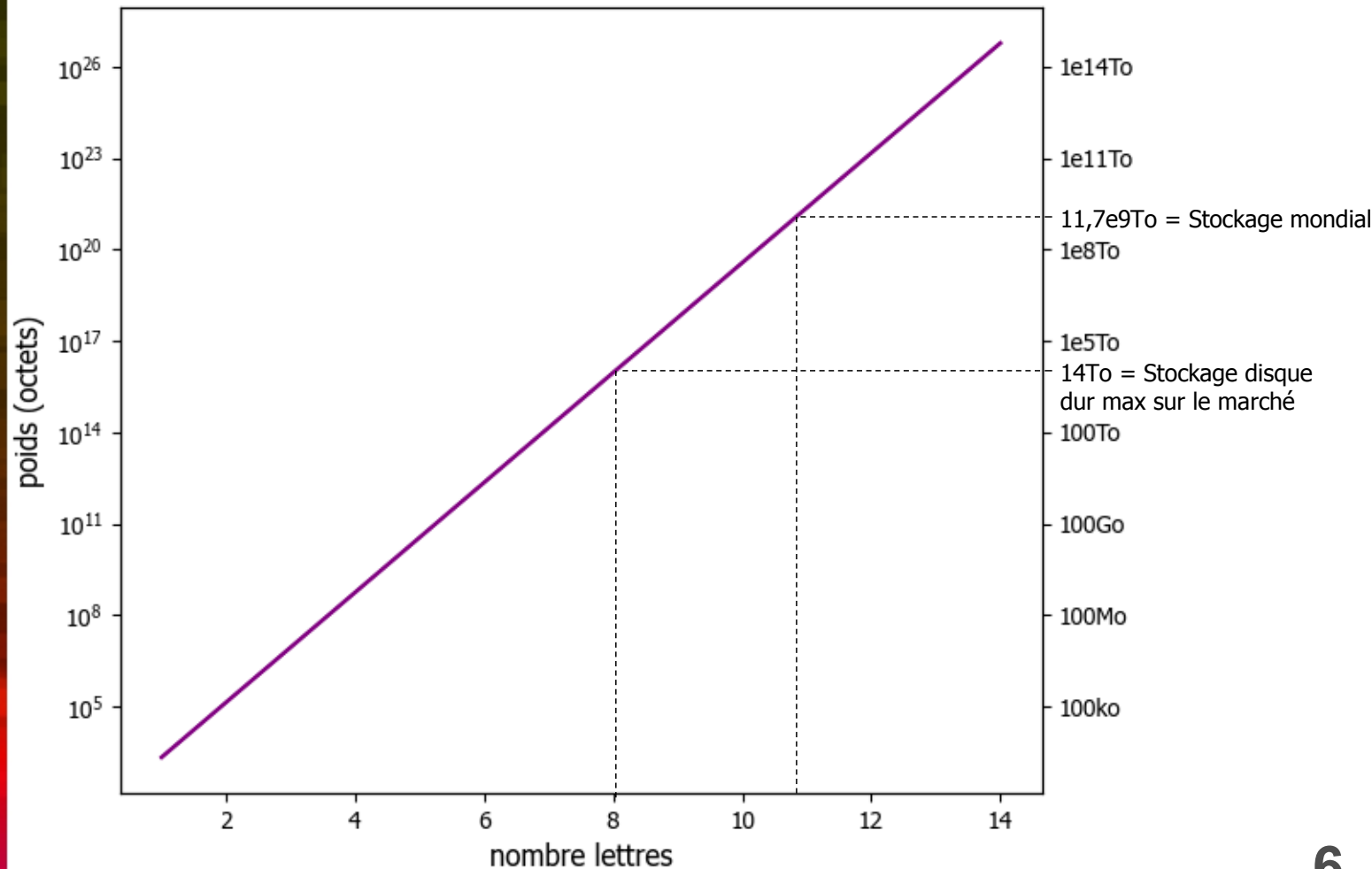


- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Mémoire

Courbe théorique avec 62 caractères

Poids pour générer tous les mots de n lettres



- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Mots de passe les plus utilisés en 2019

## En France

1	123456
2	123456789
3	azerty
4	1234561
5	qwerty
6	marseille
7	000000
8	1234567891
9	doudou
10	12345
11	loulou
12	123
13	password
14	azertyuiop
15	soleil

Source : projet Richelieu

## Dans le monde

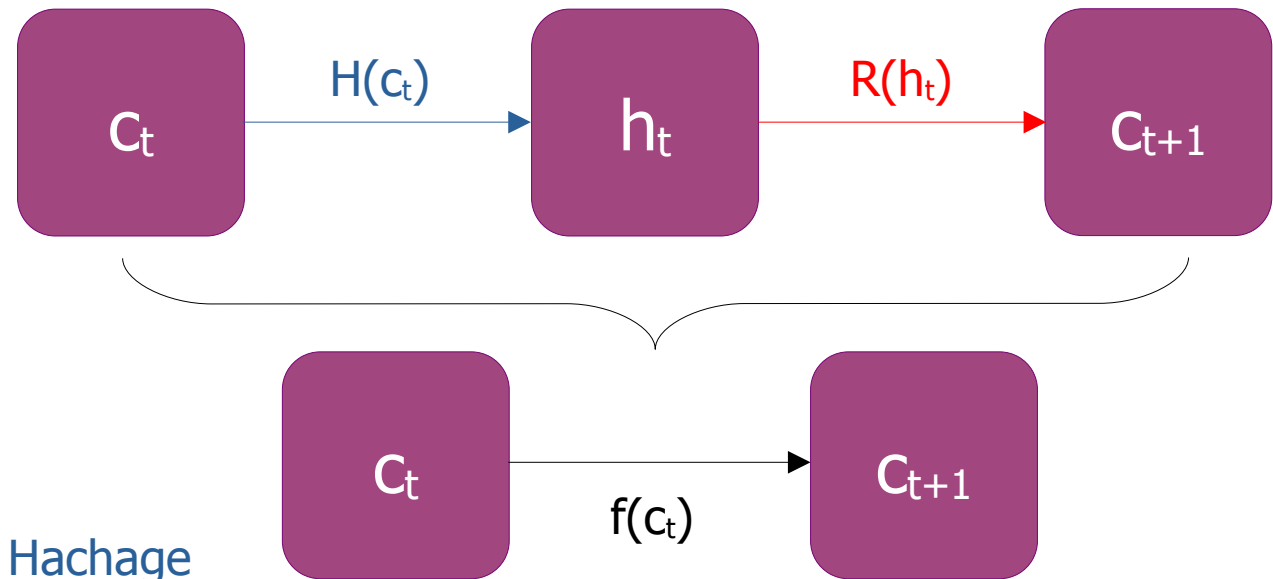
1	123456
2	123456789
3	qwerty
4	password
5	1234567
6	12345678
7	12345
8	iloveyou
9	111111
10	123123
11	abc123
12	qwerty123
13	1q2w3e4r
14	admin
15	qwertyuiop

Source : SplashData

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Table classique - Explications

1980 – Martin Hellman



$H \rightarrow$  Hachage  
 $R \rightarrow$  Réduction  
 $f \rightarrow R \circ H$

Par exemple :

$c = \text{TIPE}$

$h(c) = 8c776680b59aeb2f0ba0179eebe066f$

$R(h(c)) = \text{ORAL}$

Donc  $f(\text{TIPE}) = \text{ORAL}$

Exemple de fonction de réduction :

$R : h \rightarrow B(h [N])$  avec  $B$  un changement de base particulier

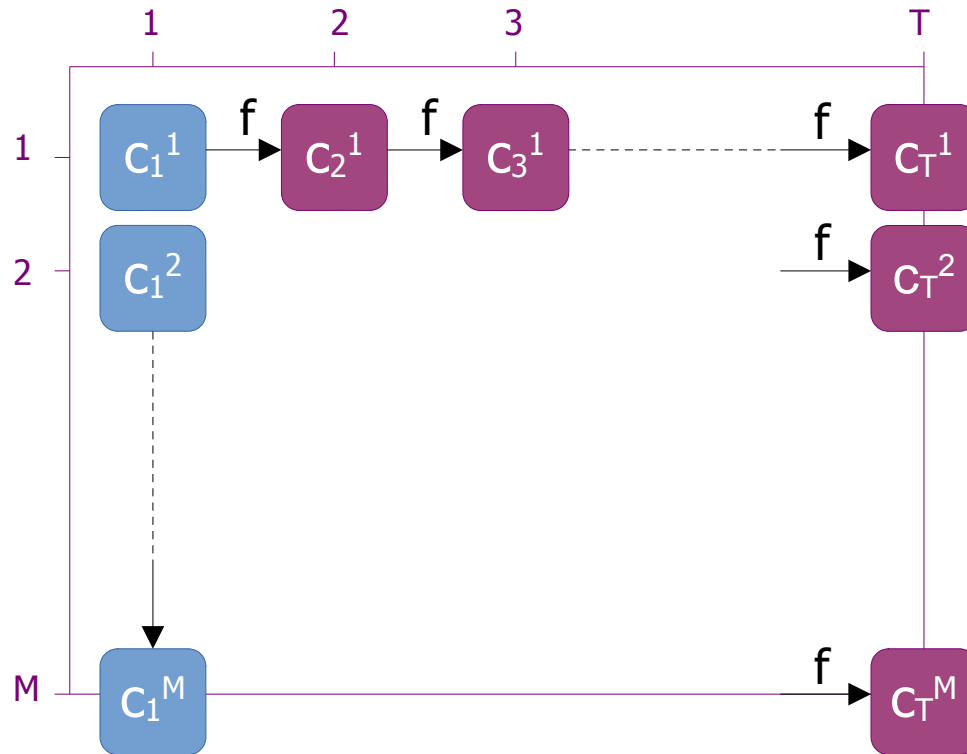


- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Table classique - Explications

1980 – Martin Hellman

Aspect théorique de la table



Stockage réel dans la base de données

$C_1^1$	$C_T^1$
$C_1^2$	$C_T^2$
...	...
$C_1^M$	$C_T^M$

$M \rightarrow$  nombre de lignes  
 $T \rightarrow$  nombre de colonnes

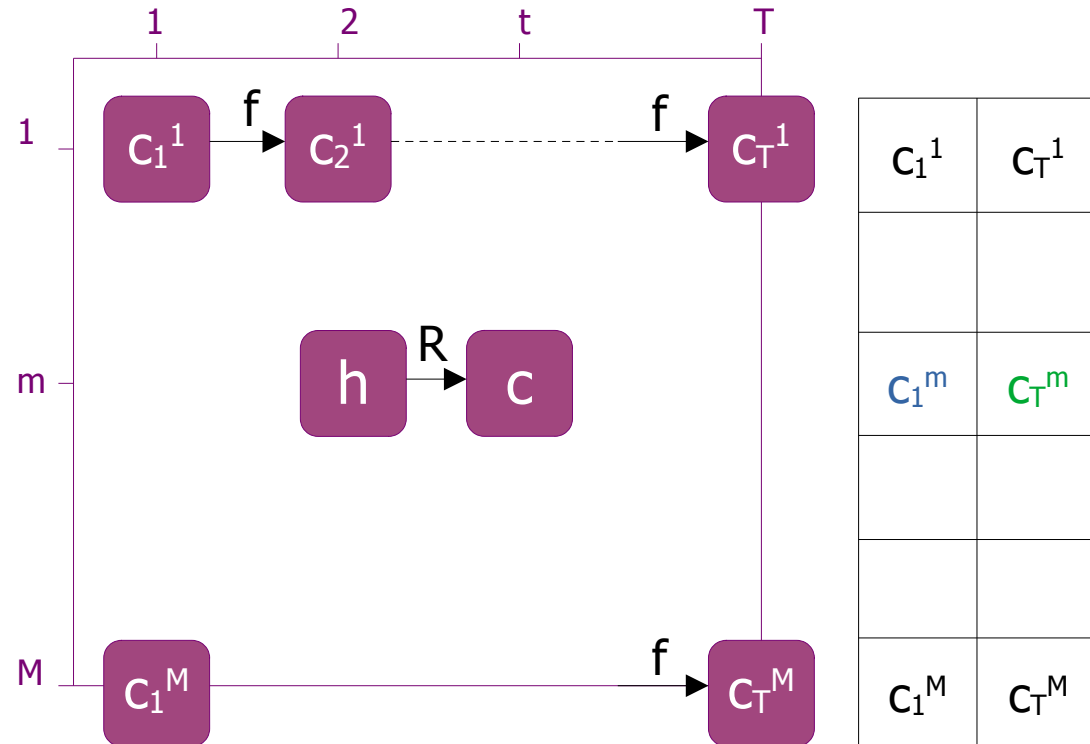
- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Table classique - Explications

1980 – Martin Hellman

Recherche de h dans la table

Soit c tel que  $R(h) = c$



Recherche ligne de h

$$f^{T-t}(c) = C_T^m$$

Ligne  $\rightarrow m$

Recherche antécédent h

On récupère  $C_1^m$

$$H(f^t(C_1^m)) = h$$

Alors l'antécédent est :  $f^{k'}(C_1^m)$

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Table classique - Explications

1980 – Martin Hellman

Par exemple :

Cherchons 7fea dans cette table

Table théorique

$c_1$	$h_1$	$c_2$	$h_2$	$c_3$	$h_3$	$c_4$
a	4fg7	g	7fea	i	6bc3	e
d	93ea	b	ab53	c	aac5	f

Table réelle

$c_1$	$c_T$
a	e
d	f

## Recherche ligne de 7fea

$R(7fea) = i$

i n'est pas dans la table

$f(i) = e$

e est dans la table, à la ligne 1

## Recherche antécédent 7fea

On récupère  $c_1 = a$  ligne 1

$H(a) = 4fg7 \neq 7fea$

$R(4fg7) = g$

$H(g) = 7fea$

g est l'antécédent de 7fea

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

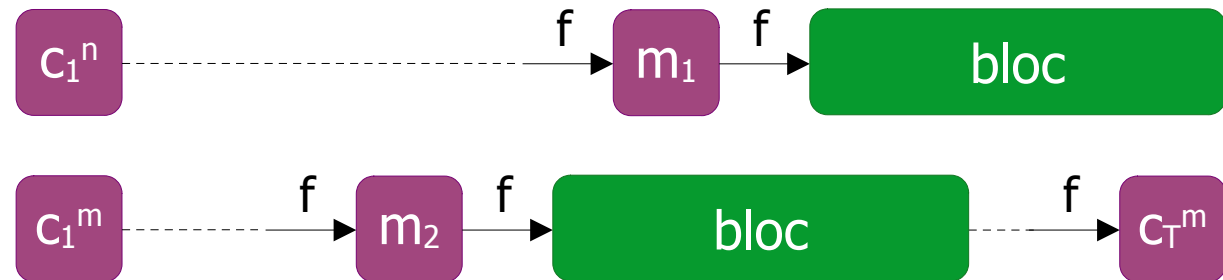
# Table classique - Limites

*1980 – Martin Hellman*

## 1 - Fusions

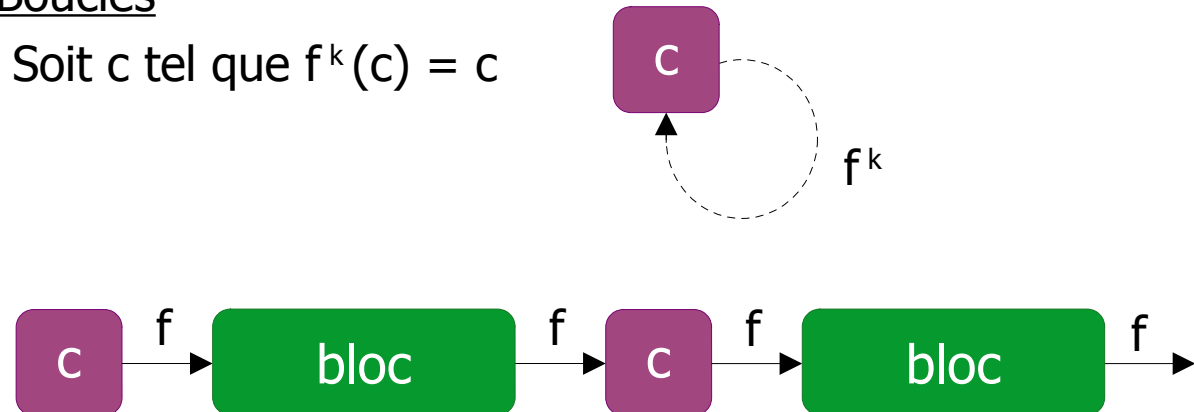
On suppose une collision :

soient  $m_1$  et  $m_2$  tels que  $f(m_1) = f(m_2)$



## 2 - Boucles

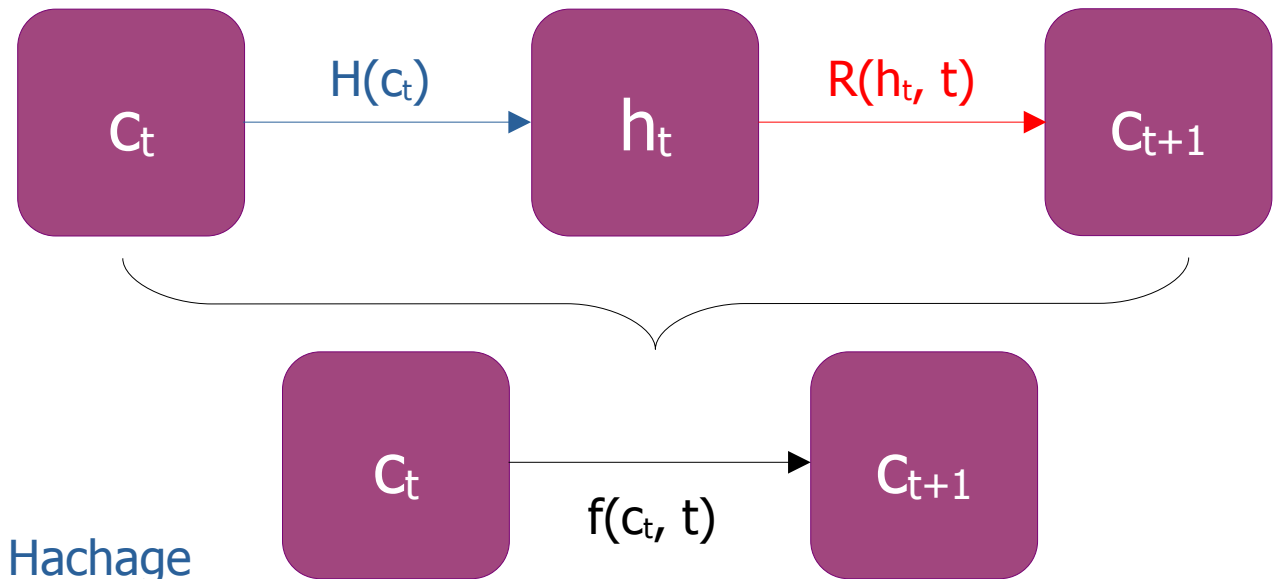
Soit  $c$  tel que  $f^k(c) = c$



- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Table arc-en-ciel - Explications

2003 – Philippe Oechslin



H → Hachage

R → Réduction qui dépend de t

f →  $R \circ H$  qui dépend de t

Par exemple :

c = TIPE

$h(c) = 8c776680b59aeb2f0ba0179eebe066f$

$R(h(c), 1) = \text{ORAL}$

Donc  $f(\text{TIPE}, 1) = \text{ORAL}$

Exemple de fonction de réduction :

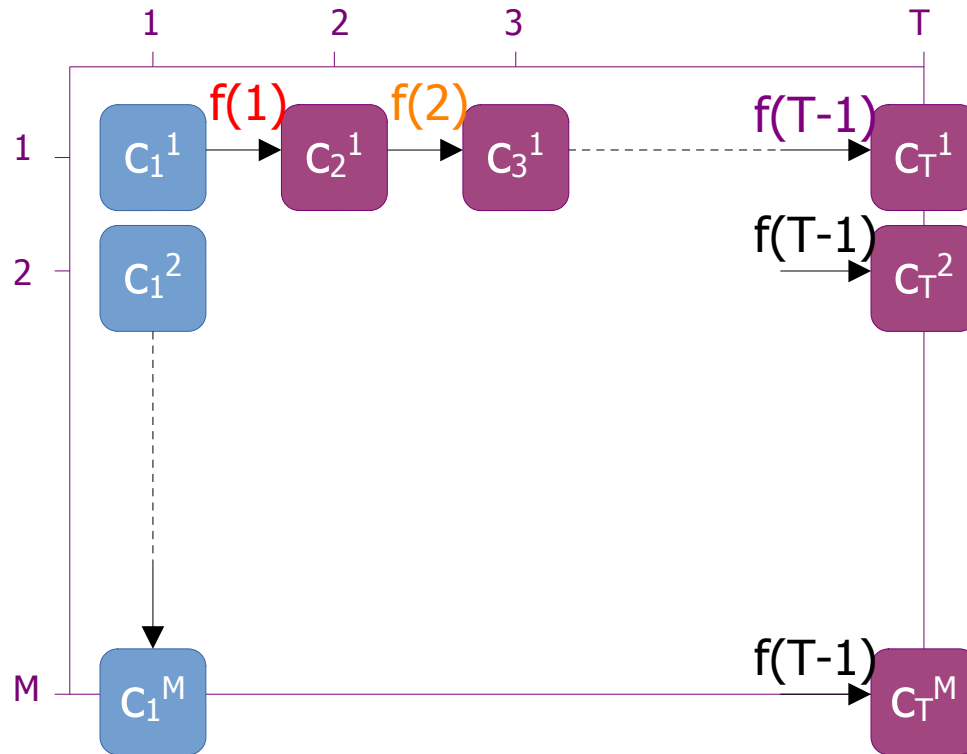
$R : h, t \rightarrow B((h+t) [N])$  avec B le changement de base

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Table arc-en-ciel - Explications

2003 – Philippe Oechslin

Aspect théorique de la table



Stockage réel dans la base de données

$C_1^1$	$C_T^1$
$C_1^2$	$C_T^2$
...	...
$C_1^M$	$C_T^M$

M → nombre de lignes  
T → nombre de colonnes

Table arc en ciel :  
 $f(1) \rightarrow f(2) \rightarrow f(3) \rightarrow f(4) \rightarrow f(5) \rightarrow f(6) \rightarrow f(7)$



- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Table arc-en-ciel - Explications

2003 – Philippe Oechslin

Par exemple :

Cherchons 7fea dans cette table

Table théorique

c <sub>1</sub>	h <sub>1</sub>	c <sub>2</sub>	h <sub>2</sub>	c <sub>3</sub>	h <sub>3</sub>	c <sub>4</sub>
a	4fg7	g	7fea	i	6bc3	e
d	93ea	b	ab53	c	aac5	f

Table réelle

c <sub>1</sub>	c <sub>T</sub>
a	e
d	f

## Recherche ligne de 7fea

On suppose 7fea dans la dernière colonne

$R(7fea, 3) = j$  (pas dans la table)

On suppose 7fea dans l'avant dernière colonne

$R(7fea, 2) = i$

$f(i, 3) = e$

e est dans la table, à la ligne 1

## Recherche antécédent 7fea

On récupère c<sub>1</sub> = a ligne 2

$H(a) = 4fg7 \neq 7fea$

$R(4fg7, 1) = g$

$H(g) = 7fea$

g est l'antécédent de 7fea

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

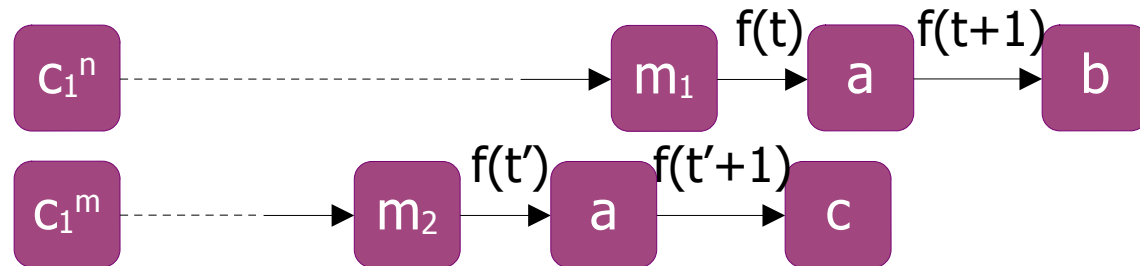
# Table arc-en-ciel - Limites

## 1 - Fusions

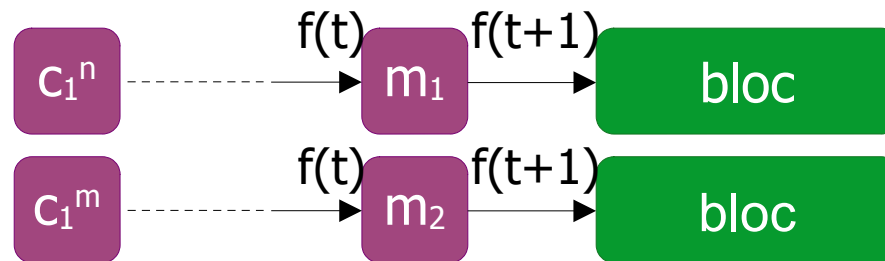
On suppose une collision :

soient  $m_1$  et  $m_2$  tels que  $f(m_1, t) = f(m_2, t') = a$

Alors • si  $t \neq t'$ ,  $f(a, t+1) \neq f(a, t'+1)$



• si  $t = t'$ ,  $f(a, t+1) = f(a, t'+1)$

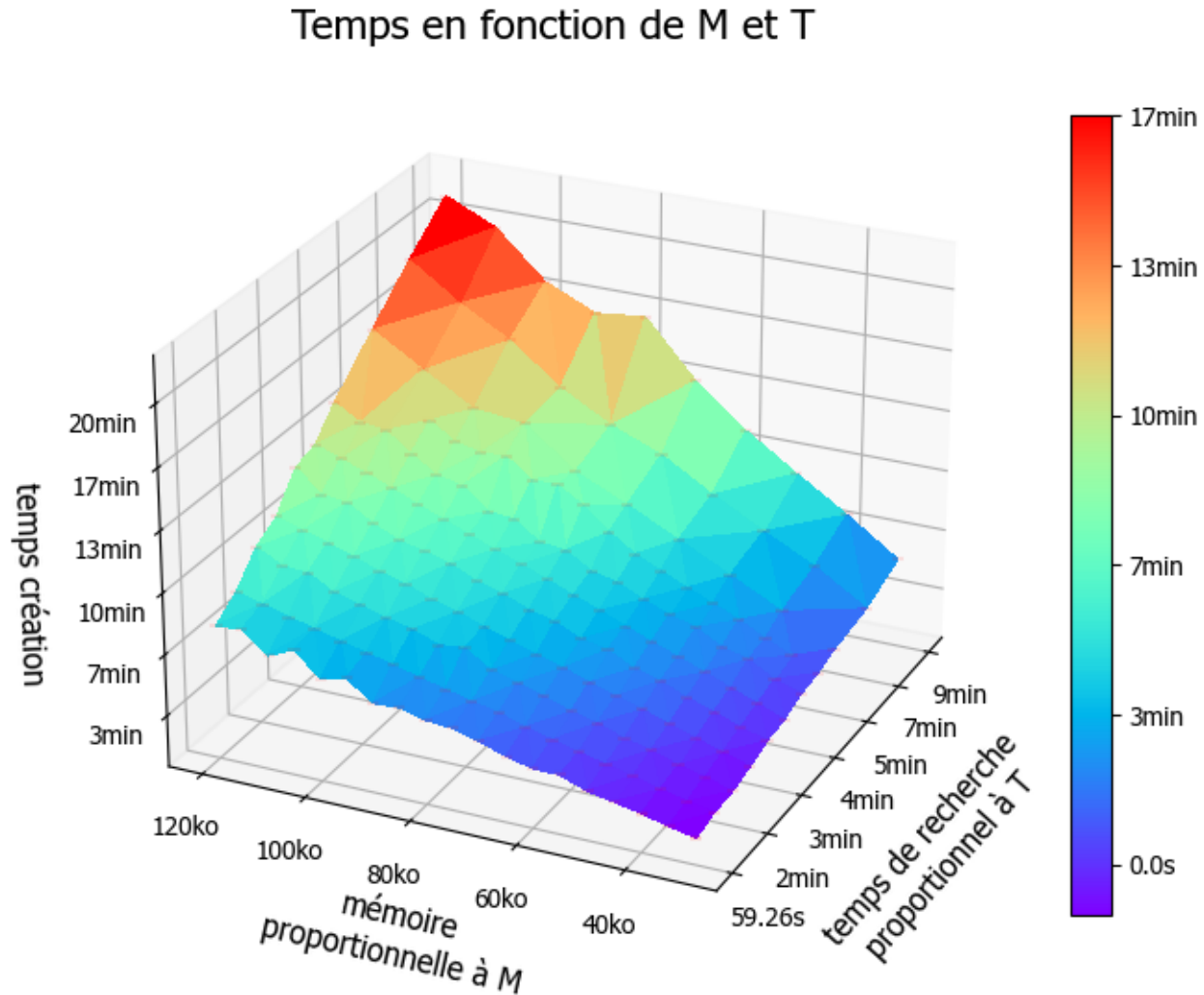


Exemple : Nombre de fois qu'un mot apparaît dans une table  $T = 1000$   $M = 5000$

Mot	hlzw	mlcb	eesr	bgko	nqon	qtpx	qcvx	vutc	hiai	wnbz
Classique	343	686	681	838	687	716	663	737	839	841
Arc-en-ciel	11	4	11	2	3	4	3	6	5	9

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Table arc-en-ciel - Compromis



- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Comparaison

'abcdefghijklmnopqrstuvwxyz  
 ABCDEFGHIJKLMNOPQRSTUVWXYZ  
 0123456789'

On recherche sur 100 mots générés aléatoirement

	Force brute 14 776 336 mots	Mémoire 14 776 336 mots	Table classique T = 1000 M = 100 000	Table Arc-en-ciel T = 1000 M = 100 000
Temps création (s)	0	90,51	2241,43	2265,46
Tps recherche médian (s)	23,96	1,39	29,07*	6,26
Écart type tps recherche (s)	12,55	0,53	43,55	6,66
Mémoire (Mo)	0	561,50	1,585	1,585

\* Long car souvent le mot recherché n'est pas dans la table donc l'algorithme parcourt toute la table avant de s'arrêter

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion

# Salage

Base de données du site web

Identifiants	Sel	Mot de passe haché
Juliette	ABCD	ba922f5bbc0bc606d94722bc39ed1533
Utilisateur 1	EFGH	0a9a8e35ec7f87a77439430b406851d1
Utilisateur 2	IJKL	b5283f3c3a3d5744a14e2c4062c90bc9

AZERTY  
+ ABCD

MD5

ba922f5bbc0bc606d94722bc39ed1533

ba922f5bbc0bc606d94722bc39ed1533

8416887187c13c56ac45cbe5ab348785

=

Bon mot de passe

≠

Mauvais mot de passe

MD5




QWERTY  
+ ABCD



- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Application The Code

TheCode




Question personnelle


Clef :


Nom du site :


Paramètres du mot de passe :


Longueur : 20 

Caractères :


Minuscules 

Majuscules 


Symboles 

Chiffres 

Le mot de passe est :



Sécurité : Très Forte 126 bits

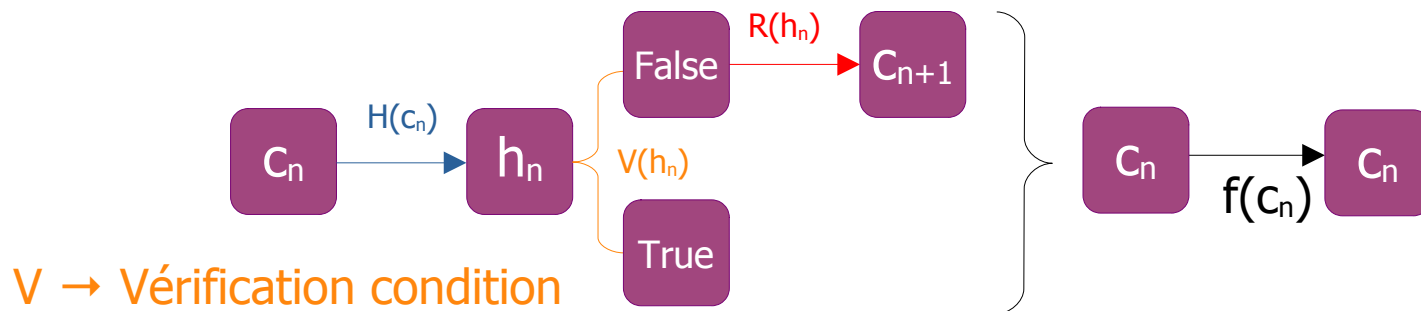




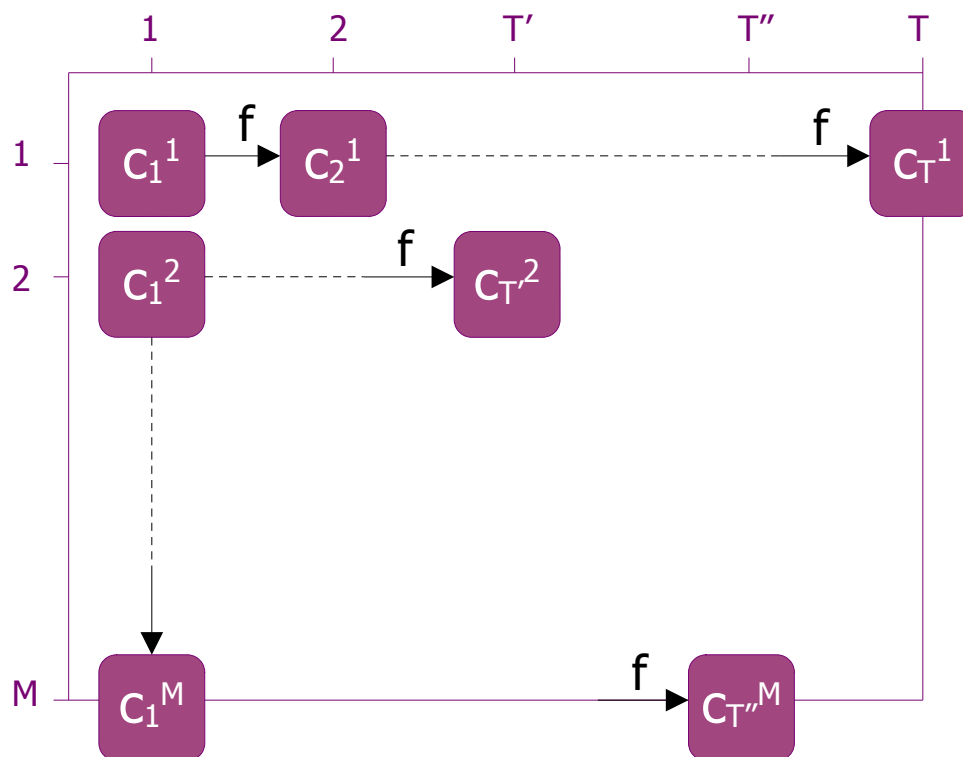
- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Table points distingués

1982 – Ronald Rivest



Aspect théorique de la table



Stockage réel dans la base de données

$C_1^1$	$C_T^1$
$C_1^2$	$C_T^2$
...	...
$C_1^M$	$C_{T''}^M$

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Table de compromis - Probabilités

## 1 – Table classique

Probabilité qu'un mot soit dans la table en fonction de M et T

$$P_{table} \geq \frac{1}{N} \sum_{i=1}^M \sum_{j=0}^{T-1} \left(1 - \frac{i * T}{N}\right)^{j+1}$$

$$\text{Si } M \times T^2 \ll N \quad \text{Alors } P(S) \geq \frac{M \times T}{N}$$

$$\text{Si } M \times T^2 \approx N \quad \text{Alors } P(S) \geq 0.8 \frac{M \times T}{N}$$

et M et T grands

## 2 – Table arc-en-ciel

Probabilité qu'un mot soit dans la table en fonction de M et T

$$P_{table} = 1 - \prod_{i=1}^T \left(1 - \frac{m_i}{N}\right)$$

$$\text{Avec } m_1 = 1 \text{ et } m_{n+1} = N \left(1 - e^{-\frac{m_n}{N}}\right)$$

$$\text{Si } N = 26^4, \quad M = 5\,000 \text{ et } T = 1000, \quad P = 97,64 \%$$

$$\text{Si } N = 26^4, \quad M = 2\,000 \text{ et } T = 500, \quad P = 77,27 \%$$

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Hachage – Algorithme MD5

## 1 – Préparation du message

- Convertir en binaire
- Ajouter 1
- Remplir avec plusieurs 0
- Ajouter la longueur du message codé sur 64 bits en little endian
- On obtient un message de longueur multiple de 512

## 2 – Constantes

$$K(i) = \lfloor \lfloor 2^{32} \times \sin(i+1) \rfloor \rfloor$$

## 3 – Boucle

- Travail sur 128 bits, divisé en 4 mots de 32 bits A, B, C, D :
- Initialisation de A, B, C, D
- Travail sur blocs de 512 bits du message :
  - Modification de A, B, C, D avec les opérations :
    - Quatre fonctions :
 
$$F = (B \wedge C) \vee (\neg B \wedge D)$$

$$G = (B \wedge D) \vee (C \wedge \neg D)$$

$$H = B \oplus C \oplus D$$

$$I = C \oplus (B \wedge \neg D)$$
    - Des additions
    - Une rotation binaire vers la gauche.

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# MD5 – Codes p2

```
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"""
    return [int(l[i:i+n], 2) for i in range(0, len(l), n)]

def normalisation(mot):
    """Normalise le mot"""
    # 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 en hexadécimal"""
    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))])
```

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# MD5 – Codes p3

```
def md5(mot):
    """Fonction MD5"""
    r = [7, 12, 17, 22] * 4 + [5, 9, 14, 20] * 4 + [4, 11, 16, 23] * 4 +
    [6, 10, 15, 21] * 4
    K = [int(abs((2**32) * math.sin(i + 1))) for i in range(64)]

    x = normalisation(mot)
    x = decomposition(x, 32)
    A, B, C, D = 0x67452301, 0xEFCDAB89, 0x98BADCFE, 0x10325476
    for j in range(0, len(x), 16):
        a, b, c, d = A, B, C, D

        for t in range(64):
            if t >= 0 and t < 16:
                f = (B & C) | ((~B) & D)
                g = t
            elif t >= 16 and t < 32:
                f = (B & D) | (C & (~D))
                g = (5 * t + 1) % 16
            elif t >= 32 and t < 48:
                f = B ^ C ^ D
                g = (3 * t + 5) % 16
            elif t >= 48 and t < 64:
                f = C ^ (B | (~D))
                g = (7 * t) % 16

            new_B = (A + f + x[j + g] + K[t]) % 2**32
            new_B = (ROTL(new_B, r[t]) % 2**32) + B
            A, B, C, D = D, new_B, B, C

        A = (a + A) % 2**32
        B = (b + B) % 2**32
        C = (c + C) % 2**32
        D = (d + D) % 2**32
    return toHex(A) + toHex(B) + toHex(C) + toHex(D)
```

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Fonctions utiles – Codes

```
import hashlib
```

```
def hachage(mot):  
    """hash mot avec md5"""  
    return hashlib.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
```



- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Hachage Image – Codes p4

```
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]

    hacher(hach, file)
```

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Force brute – Codes p5

```
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"
mini, maxi = 4, 4
hach = hachage("TiP3") # Hache le mot choisi
mot = forceBrute(hach, mini, maxi, caracteres) # Cherche l'antécédent du hach

# Pour calculer le temps pour chaque longueur de lettre expérimentalement
# forceBrute renvoie le temps de recherche et non le mot trouvé
caracteres = "abcdefghijklmnopqrstuvwxyz"
i = 10
for t in range(1, i):
    a = forceBrute(hachage(caracteres[-1] * int(t)), t, t)
    print(t, a)

alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
l = len(alphabet)
t = 2 / 1000000 # Temps pour une opération de forceBrute
x = [i for i in range(1, 15)]
y = [(l**n) * t for n in x] # Liste du temps théorique
# l**n est le nombre d'opération dans le programme
```

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Mémoire – Codes p10

```
# forceBrute transmet la liste des mots testés
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)
    creer_table(4)

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'
hash = hachage("TiP3")
mot = recherche(database, hash)
```

Table : Memoire

	mot	hash
	Filtre	Filtre
1	aaaa	74b87337454200d4d33f80c4663dc5e5
2	aaab	4c189b020ceb022e0ecc42482802e2b8
3	aaac	3963a2ba65ac8eb1c6e2140460031925
4	aaad	aa836f154f3bf01eed8df286a1fbb388
5	aaae	5f83cfb6ca6b50d3323a6377e1241b1f
6	aaaf	14acbed198e2456a400321cd9b065ce3

Allure d'une table

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Dictionnaire – Codes p13

```
import sqlite3
import hashlib

database = "./Mots.sqlite"

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)))
```

Table : MOTS

	id	ortho	cgram	freqlemfilms	freqlemlivres	freqfilms	freqlivres
	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	a	NOM	81.36	58.65	81.36	58.65
2	2	a	AUX	18559.22	12800.81	6350.91	2926.69
3	3	a	VER	13572.4	6426.49	5498.34	1669.39
4	4	a capella	ADV	0.04	0.07	0.04	0.07
5	5	a cappella	ADV	0.04	0.07	0.04	0.07
6	6	a contrario	ADV	0.0	0.27	0.0	0.27
7	7	a fortiori	ADV	0.04	0.88	0.04	0.88

Base de données  
des mots de la  
langue française

142 694 lignes

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Tables – Codes p17

```
def ialea(self, tableName, l) :
    """Indice aléatoire de départ d'une ligne"""
    conn = sqlite3.connect(self.database.format(tableName))
    cur = conn.cursor()
    I = 0

    def inter(cur, conn, tableName, l, i):
        """Choisi indice aleatoire qui n'est pas dans la table"""
        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
        i_0 = inter(cur, conn, tableName, l, i)
        conn.close
        return i_0
```

```
# i2c est dec2base
# h est le hachage
```

```
def h2i(self, h, t) :
    """Réduit le hash en un entier (R)"""
    h = str(h)
    return (int(h, 16) + t) % self.N
```

```
def h2h(self, h1, t) :
    """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, t) :
    """Passe d'un entier au suivant (f)"""
    c1 = self.i2c(i1)
    h1 = self.h(c1)
    i2 = self.h2i(h1, t)
    return i2
```

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Tables – Codes p18

```
# Création objet datas :
```

```
carac = 'abcdefghijklmnopqrstuvwxyz'  
t_min = 4 # Longueur mot min  
t_max = t_min # Longueur mot max  
t = 500 # Nombre colonnes  
m = 2000 # Nombre lignes  
l = 1 # Nombre de tables  
type = 'ArcEnCiel' # Type de table  
hachage = 'md5', 'utf-8' # md5
```

```
database = "./Table {0} t = {1} m = {2} l = {3} len = {4}.sqlite".format(type,  
t, m, l, t_max)
```

```
datas = ArcEnCiel(m, t, l, t_min, t_max, carac, AEC, database, hachage) # Objet  
contenant les informations
```



- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Créer tables – Codes p19

```
import sqlite3
from ArcEnCiel import *

def table(datas):
    """Création table Arc en ciel si AEC, Classique sinon
    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
    """
    tableName = datas.AEC
    conn = sqlite3.connect(datas.database)
    cur = conn.cursor()

    for l in range(datas.l):
        cur.execute("""CREATE TABLE {
            (i_0 INT,i_t INT);""".format(tableName + str(l)))
        for _ 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 {
                VALUES(?, ?);""".format(tableName + str(l)), (i_0, i_t))
            conn.commit()
        conn.commit()
    conn.close()

table(datas) # Création table AEC
```

Table: ArcEnCiel0

	i_0	i_t
	Filtre	Filtre
1	513452	9225022
2	13826029	72460
3	1994746	1204279
4	1687533	865496
5	5813746	4052945
6	8221375	6758875

Allure d'une table

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Recherche tables – Codes p20

```
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.AEC:
            t = 1
        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
```

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Recherche tables – Codes p21

```
def inverse(hash : str, datas : ArcEnCiel) -> str:
    """Cherche le mot de passe d'origine ayant hash comme image hachée"""
    tableName = datas.AEC
    conn = sqlite3.connect(datas.database)
    cur = conn.cursor()

    if datas.AEC == '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
                cur.execute("""SELECT i_0 FROM {} WHERE i_t = ?""".format(tableName + str(l)), (indice,))
                result = inter(cur.fetchall(), l, i, hash, datas)
                if result:
                    return result
    else:
        for l in range(datas.l):
            # Dans chaque table
            indice = datas.h2i(hash, l)
            for t in range(datas.t):
                indice = datas.i2i(indice, l)
                cur.execute("""SELECT i_0 FROM {} WHERE i_t = ?""".format(tableName + str(l)), (indice,))
                result = inter(cur.fetchall(), l, t, hash, datas)
                if result:
                    return result

    conn.close()
    return "Pas dans la table"
```

hach = datas.h("TiP3") # Hache le mot choisi  
 mot = inverse(hach, datas) # Cherche l'antécédent du hach

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Courbe Arc-en-ciel – Codes p22

```

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()

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

            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()

conn.close

```

Table : Temps

	M	T	temps
	Filtre	Filtre	Filtre
1	3000	3000	119.390758991241
2	4000	3000	159.327769756317
3	3000	3000	115.968366861343
4	4000	3000	154.968197822571
5	5000	3000	193.092617988586
6	6000	3000	227.517811059952
7	7000	3000	277.67861199379
8	8000	3000	307.514044284821

Allure de la table

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Courbe Arc-en-ciel – Codes p23

```
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**(1-e)), 0))) + "e" + str(1-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]
```

- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Courbe Arc-en-ciel – Codes p23

```
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)
cb = fig.colorbar(trisurf, ax = ax, shrink = 0.8)
cb.ax.set_yticklabels([tempsSTR(i) for i in ax.get_zticks()], fontsize = 10)

ax.view_init(*(25, -155))
ax.set_title('Temps en fonction de M et T', fontsize = 16)
ax.set_xlabel('\n\ntemps de recherche\nproportionel à T', fontsize = 12)
ax.set_ylabel('\n\nmémoire\nproportionelle à M', fontsize = 12)
ax.set_zlabel('\n\ntemps création', fontsize = 12)
plt.savefig("./Courbes T = 9000 M = 12000 Pas = 1000")
```



- Introduction
- Techniques basiques de recherche
- Tables de compromis temps mémoire
- Comparaison
- Conclusion
- Annexes

# Comparaison – Codes p27

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



```

# 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

```

MD5

```

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))])

def md5(mot):
    """ Fonction MD5 """
    r = [7, 12, 17, 22] * 4 + [5, 9, 14, 20] * 4 + [4, 11, 16, 23] * 4 + [6, 10, 15, 21] * 4
    K = [int(abs((2**32) * math.sin(i + 1))) for i in range(64)]
    x = normalisation(mot)
    x = decomposition(x, 32)
    A, B, C, D = 0x67452301, 0xEFCDAB89, 0x98BADCFE, 0x10325476
    for j in range(0, len(x), 16):
        a, b, c, d = A, B, C, D
        for t in range(64):
            if t >= 0 and t < 16:
                f = (B & C) | ((~B) & D)
                g = t
            elif t >= 16 and t < 32:
                f = (B & D) | (C & (~D))
                g = (5 * t + 1) % 16
            elif t >= 32 and t < 48:
                f = B ^ C ^ D
                g = (3 * t + 5) % 16
            elif t >= 48 and t < 64:
                f = C ^ (B | (~D))
                g = (7 * t) % 16
            new_B = (A + f + x[j + g] + K[t]) % 2**32
            new_B = (ROTL(new_B, r[t]) % 2**32) + B
            A, B, C, D = D, new_B, B, C
        A = (a + A) % 2**32
        B = (b + B) % 2**32
        C = (c + C) % 2**32
        D = (d + D) % 2**32
    return toHex(A) + toHex(B) + toHex(C) + toHex(D)

mot = "Demain"
print(md5(mot)) # b3a46a9f4cbf9bd55c64602ae9b70476
import hashlib
print(hashlib.md5(mot.encode('utf-8')).hexdigest()) # b3a46a9f4cbf9bd55c64602ae9b70476

```

MD5

```

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))

```

Force Brute

```

caracteres = "abcdefghijklmnopqrstuvwxyz"
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 forceBrute(mdp, mini = 1, maxi = 10, caracteres = ""):
    """Test par force brute jusqu'à ce que la valeur vaille mdp"""
    tps = time.time()
    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 time.time() - tps

i = 10
for t in range(1, i):
    a = forceBrute(hachage(caracteres[-1] * int(t)), t, t, caracteres)
    print(t, a)

```

Temps force brute  
expérimental

```

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 + 1**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([1**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)

```

Temps force brute  
théorique

```
# 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]
```

Temps force brute  
théorique

```

import matplotlib.pyplot as plt
import numpy as np

arrondi = lambda l, x, e : str(int(round(x/(10**(1-e)), 0))) + "e" + str(1-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()

```

Courbe force brute



```

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)
ax3.set_xlabel('nombre lettres', fontsize = 12)
ax3.set_ylabel('temps (s)', fontsize = 12)
ax1.set_title("Temps pour générer tous les mots de n lettres\n", fontsize = 16)
plt.savefig("Temps pour générer tous les mots jusqu'à {} caractères.png".format(i))
plt.show()

courbe([0.00015807151794433594, 0.005568981170654297, 0.11891508102416992, 3.490976095199585,
        86.7342791557312, 2402.337882757187, 65502.59010100365],
        [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])

```

Courbe force brute

```

# 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

```

```

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))

```

```

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**(1-e)), 0))) + "e" + str(1-e) if l > e else str(x)

alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
l = len(alphabet)

def taille(n):
    nb_ligne = 1*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**(1 - 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)

```

```

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)))

```



```

"""Classe ArcEnCiel
Contient * les données pour les tables de compromis temps-mémoires
        * les fonctions pour générer une table ou rechercher un mot dedans"""
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 nécessaires 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 directement 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

```

Classe Arc en ciel

```

        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 = {}
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)

```

Classe Arc en ciel

```

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"""
    self._type = type
    self._couverture = self.couv()
    self._database = "/Users/juliettedebono/Documents/MP*/TIPE/III- Compromis/III- {5}
{0}/Tables/{0} t = {1} m = {2} l = {3} len = {4}.sqlite".format("{0}", self._t, self._m, self._l,
self._t_max, lettre(type))
type = property(_get_type, _set_type, _impossible)

```

```

def _get_carac(self):
    """Retourne les caractères possibles"""
    return self._carac
def _set_carac(self, carac):
    """Modifie carac"""
    self._carac = carac
    self._nblettres = len(carac)
    self._N = sum([self.nblettres**i for i in range(self._t_min, self._t_max+1)])
carac = property(_get_carac, _set_carac, _impossible)

def _get_database(self):
    """Retourne le nom de la base de données"""
    return self._database.format(self.type)
def _set_database(self, database):
    """Modifie database"""
    self._database = database
database = property(_get_database, _set_database, _impossible)

def _get_hachage(self):
    """Retourne la fonction de hachage"""
    return self._hachage
def _set_hachage(self, hachage):
    """Modifie la fonction de hachage"""
    self._hachage = hachage
hachage = property(_get_hachage, _set_hachage, _impossible)

def _get_couverture(self):
    """Retourne la valeur de la couverture"""
    return self._couverture
couverture = property(_get_couverture, _impossible, _impossible)

def _get_N(self):
    """Retourne N"""
    return self._N
N = property(_get_N, _impossible, _impossible)

def _get_nblettres(self):
    """Retourne le nombre de lettres"""
    return self._nblettres
nblettres = property(_get_nblettres, _impossible, _impossible)

```

Classe Arc en ciel

```

def couv(self):
    """Renvoie la couverture de datas : le pourcentage de valeur contenue dans la table type"""
    if self.type == 'ArcEnCiel':
        m = self.m
        v = 1.0
        for _ in range(self.t):
            v *= (1 - m / self.N)
            m = self.N * (1 - np.exp(-m/self.N))
        p = (1 - (v**self.l))
        return round(100 * p, 2)
    else:
        if self.m * (self.t**2) < 10 * self.N:
            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, tableName, 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 i_0

```

Classe Arc en ciel

```

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)
    h1 = self.h(c1)
    i2 = self.h2i(h1, t)
    return i2

```

```

def pick(self) -> str:
    """Choisi un hash stocké dans la table aléatoirement"""
    conn = sqlite3.connect(self.database)
    cur = conn.cursor()
    l = rd.randint(0, self.l - 1)
    cur.execute("""SELECT i_0 FROM {}{} ORDER BY RANDOM()""".format(self.type, l))
    a = cur.fetchone()[0]
    conn.close
    for i in range(rd.randint(1, self.t - 1)):
        if self.type != "ArcEnCiel":
            i = 1
        else:
            i += 1
        a = self.i2i(a, i)
    return self.i2c(a)

```

```

# Informations :
carac = 'abcdefghijklmnopqrstuvwxyz'+'abcdefghijklmnopqrstuvwxyz'.upper() + '0123456789'
t_min = 4
t_max = t_min
t = 1000
m = 100000
l = 1
type = "ArcEnCiel"
hachage = 'md5', 'utf-8' # md5
lettre = lambda a : "b" if a == 'ArcEnCiel' else "a"
database = "/Users/juliettedebono/Documents/MP*/TIPE/III- Compromis/III- {5}) {0}/Tables/{0} t = {1} m
= {2} l = {3} len = {4}.sqlite".format("{0}", t, m, l, t_max, lettre(type))
datas = ArcEnCiel(m, t, l, t_min, t_max, carac, type, database, hachage) # Objet contenant les
informations
del carac, type, t, m, l, hachage, database, t_min, t_max # Suppression valeurs inutiles

```



```

import time
import sqlite3
from ArcEnCiel import *

def table(datas : ArcEnCiel) -> None:
    """Création table Arc en ciel ou Classique
    Renvoie une base de donnée d'une table de compromis :
    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))
        tableName1 = tableName + str(l)
        try:
            cur.execute("""DROP TABLE {}""".format(tableName1))
        except sqlite3.OperationalError:
            pass
        cur.execute("""CREATE TABLE {}(
            i_0 INT,
            i_t INT);""".format(tableName1))
        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 = 1
                i_t = datas.i2i(i_t, t)
            cur.execute("""
                INSERT INTO {}
                (i_0, i_t)
                VALUES(?, ?);""".format(tableName1), (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

```

Création table  
compromis

```
# 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()
```

```

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 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 = 1
        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
            tableName1 = tableName + str(l)
            cur.execute("""SELECT i_0 FROM {} WHERE i_t = ?""".format(tableName1), (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):
            print("Table {}".format(l))
            # Dans chaque table
            indice = datas.h2i(hash, l)
            tableName1 = tableName + str(l)
            for t in range(datas.t):
                indice = datas.i2i(indice, l)
                cur.execute("""SELECT i_0 FROM {} WHERE i_t = ?""".format(tableName1), (indice,))
                result = inter(cur.fetchall(), l, t, hash, datas)
                if result:
                    return result
            if t % 100 == 0:
                print("Colonne {}".format(t))
    conn.close
    return "Pas dans la table"

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

```

```

def alea(datas : ArcEnCiel):
    """Génère un mot aléatoire suivant les caractéristiques de la table"""
    result = ""
    for _ in range(rd.randint(datas.t_min, datas.t_max)):
        result += rd.choice(datas.carac)
    return result

# result = datas.pick() # Choisi aléatoirement un hach qui est dans la base de données
# result = alea(datas) # Choisi aléatoirement un mot qui pourrait être dans la base de données
result = "TiP3" # Mot imposé
hach = datas.h(result) # Hache le mot choisi
print(hach, result) # Affiche le hach du mot cherché
t = time.time() # Démare le compteur de temps
mot = inverse(hach, datas) # Cherche l'antécédent du hach
print(mot) # Affice le mot de passe trouvé
print(testValidite(mot, hach, datas)) # Vérifie que c'est le bon mot
print("Temps pour recherche table {} : {} secondes".format(datas.type, time.time() - t))

```

```

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()

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

```

```

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**(1-e)), 0))) + "e" + str(1-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**(1 - l1)), 0))) + unite[((l-l1) // 3)]
    else:
        return o_s + unite[0]

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

```

Courbe 3D temps  
création table AEC

```

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

```

Courbe 3D temps  
création table AEC



```

# 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 = 1
        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

```

Comparaison fusion

```

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
                tableName1 = tableName + str(l)
                cur.execute("""SELECT i_0 FROM {} WHERE i_t = ?""".format(tableName1), (indice,))
                result = inter(cur.fetchall(), l, i, hash, datas)
                if result:
                    count += 1
    else:
        for l in range(datas.l):
            # Dans chaque table
            indice = datas.h2i(hash, l)
            tableName1 = tableName + str(l)
            for t in range(datas.t):
                indice = datas.i2i(indice, l)
                cur.execute("""SELECT i_0 FROM {} WHERE i_t = ?""".format(tableName1), (indice,))
                result = inter(cur.fetchall(), l, t, hash, datas)
                if result:
                    count += 1
    conn.close
    return count

def alea(datas : ArcEnCiel):
    """Génère un mot aléatoire suivant les caractéristiques de la table"""
    result = ""
    for _ in range(datas.tailles[rd.randint(0, len(datas.tailles) - 1)]):
        result += rd.choice(datas.carac)
    return result

```

Comparaison fusion

```

def fusions(datas):
    listeClassic = []
    listeAEC = []
    listeMot = []
    for i in range(10):
        datas.type = "Classique"
        mot = datas.pick()
        listeMot.append(mot)
        hash = datas.h(mot)
        listeClassic.append(inverse(hash, datas))
        datas.type = 'ArcEnCiel'
        listeAEC.append(inverse(hash, datas))
        print("ET DE {}".format(i+1))
        print(listeClassic, listeAEC)
    return listeClassic, listeAEC, listeMot

listeClassic, listeAEC, listeMot = fusions(datas)
print(listeClassic, listeAEC, listeMot)
"""
t = 1000
m = 5000
l = 1
N = 456976
listeMot = ['hlzw', 'mlcb', 'eesr', 'bgko', 'nqon', 'qtpx', 'qcvx', 'vutc', 'hiai', 'wnbz']
listeClassic = [343, 686, 681, 838, 687, 716, 663, 737, 839, 841]
listeAEC = [11, 4, 11, 2, 3, 4, 3, 6, 5, 9]
"""

```

Comparaison fusion

```

def fusions(datas):
    listeClassic = []
    listeAEC = []
    listeMot = []
    for i in range(10):
        datas.type = "Classique"
        mot = datas.pick()
        listeMot.append(mot)
        hash = datas.h(mot)
        listeClassic.append(inverse(hash, datas))
        datas.type = 'ArcEnCiel'
        listeAEC.append(inverse(hash, datas))
        print("ET DE {}".format(i+1))
        print(listeClassic, listeAEC)
    return listeClassic, listeAEC, listeMot

listeClassic, listeAEC, listeMot = fusions(datas)
print(listeClassic, listeAEC, listeMot)
"""
t = 1000
m = 5000
l = 1
N = 456976
listeMot = ['hlzw', 'mlcb', 'eesr', 'bgko', 'nqon', 'qtpx', 'qcvx', 'vutc', 'hiai', 'wnbz']
listeClassic = [343, 686, 681, 838, 687, 716, 663, 737, 839, 841]
listeAEC = [11, 4, 11, 2, 3, 4, 3, 6, 5, 9]
"""

```

Comparaison fusion

```

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

```

Comparaison  
méthodes

```

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', 'x04u', 'p0wK', 'wXxk', 'OrYy', 'WClb', '58H7', 'o3HW', 'ytXF', 'PGI6',
'2WDc', 'UIzl', 'sw7T', 'NBIw', 'FNAF', '67qO', 'wbil', 'J6vI', 'F5B5', '9rmg', 'Orjf', '0YNc', 'SomR',
'ds8s', 'drCk', 'qTwc', 'Qla7', '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', 'lCgS', 'A2aQ', 'DmeR', 'qBFB', 'v2YO', 'pyA1', 'UEB2', 'ImQf', 'VxWO',
'DQ8l', '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']

```

```

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]),

```

Comparaison  
méthodes

```

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, 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]])]
```



```

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.AppCompatActivity$AppCompatActivityDelegate;
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);
    symSwitch = findViewById(R.id.symSwitch);
    chiSwitch = findViewById(R.id.chiSwitch);
    securiteSeekBar = findViewById(R.id.securiteSeekBar);
    longueurSeekBar = findViewById(R.id.longueurSeekBar);
    // MARK : Connexion avec fonctions
    clefEditText.addTextChangedListener(textWatcher);
    siteEditText.addTextChangedListener(textWatcher);
    longueurSeekBar.setOnSeekBarChangeListener(longueurListener);
    securiteSeekBar.setOnSeekBarChangeListener(securiteListener);
    //clefEditText.setInputType(InputType.TYPE_CLASS_TEXT | InputType.TYPE_TEXT_VARIATION_PASSWORD);
    lancer();
}
// MARK : Initialisations
private String base = "";
private static boolean darkMode;
// MARK : Fonctions
private void lancer() {
    File mFile = new File(Environment.getExternalStorageDirectory().getPath() + "/Android/data/" + getPackageName() +
"/files/" + fileName);
    String text = read(mFile);
    int actualMode = AppCompatDelegate.getDefaultNightMode(); // 1 : Mode Sombre, 2 : Mode clair

    darkMode = "DARK".equals(text);
    if (darkMode == (actualMode == 2)){
        setDarkMode();
    }
}
private String setDarkMode() {
    File mFile = new File(Environment.getExternalStorageDirectory().getPath() + "/Android/data/" + getPackageName() +
"/files/" + fileName);
    String message;
    if (!darkMode){
        AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
        message = "Mode clair activé";
        write("LIGHT", mFile);}
}

```

Application  
MainActivity.java

```

else {
    AppCompatActivity.setDefaultNightMode(AppCompatActivity.MODE_NIGHT_NO);
    message = "Mode sombre activé";
    write("DARK", mFile);
}
startActivity(new Intent(getApplicationContext(), MainActivity.class));
finish();
return message;
}

private void write(String text, File mFile) {
    try {
        // Flux interne
        FileOutputStream output = openFileOutput(fileName, MODE_PRIVATE);
        // On écrit dans le flux interne
        output.write(text.getBytes());
        if(output != null)
            output.close();
        // Si le fichier est lisible et qu'on peut écrire dedans
        if(Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())
            && !Environment.MEDIA_MOUNTED_READ_ONLY.equals(Environment.getExternalStorageState())) {
            // On crée un nouveau fichier. Si le fichier existe déjà, il ne sera pas créé
            mFile.createNewFile();
            output = new FileOutputStream(mFile);
            String darkName = "DARK";
            output.write(darkName.getBytes());
            if(output != null)
                output.close();
        }
    }
    catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private String read(File mFile) {
    String text = "";
    try{
        FileInputStream input = openFileInput(fileName);
        int value;
        // On utilise un StringBuffer pour construire la chaîne au fur et à mesure
        StringBuffer lu = new StringBuffer();
        // On lit les caractères les uns après les autres
        while ((value = input.read()) != -1) {
            // On écrit dans le fichier le caractère lu
            lu.append((char) value);
        }
        text = lu.toString();
    }
}

```

```

        if (input != null)
            input.close();
    if (Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())) {
        lu = new StringBuffer();
        try {
            input = new FileInputStream(mFile);
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        }
        while ((value = input.read()) != -1)
            lu.append((char) value);
        text = lu.toString();
        if (input != null)
            input.close();
    }
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
return text;
}

private void copier(java.lang.String mot) {
    // Copie le mot dans le presse papier
    if (mot != null) {
        ClipboardManager clipboard = (ClipboardManager) getSystemService(CLIPBOARD_SERVICE);
        assert clipboard != null;
        android.content.ClipData clip = android.content.ClipData.newPlainText("Mot de passe", mot);
        clipboard.setPrimaryClip(clip);
    }
}

private static String dec2Base(BigInteger x, String base) {
    // Convertit un BigInteger dans une base ayant base comme support
    BigInteger b = new BigInteger(String.valueOf(base.length()));
    StringBuilder result = new StringBuilder();
    BigInteger zero = new BigInteger("0");
    if (x.equals(zero)) {
        result = new StringBuilder(base.charAt(0));
    } else {
        while (!x.equals(zero)) {
            int inter = x.mod(b).intValue();
            result.insert(0, base.charAt(inter));
            x = x.divide(b);
        }
    }
    return result.toString();
}
}

```

Application  
MainActivity.java

```

@SuppressLint("SetTextI18n")
private void generer() {
    // Génère le mot de passe
    modifBase();
    motPasseEditText.setText("Il manque des valeurs");
    if (clefEditText.getText().toString().length() == 0 || siteEditText.getText().toString().length() == 0) {
        // Rien dans site ou dans clef
    } else if (!(minSwitch.isChecked() || majSwitch.isChecked() || chiSwitch.isChecked() || symSwitch.isChecked())) {
        // Aucun checkbuttons cliqués : Toast pour prévenir
        Toast.makeText(MainActivity.this, "Aucun type de caractères n'est coché", Toast.LENGTH_LONG).show();
    } else {
        // On peut générer le mot de passe
        String clef = clefEditText.getText().toString();
        String site = siteEditText.getText().toString();
        String[] result = modification(site + clef);
        motPasseEditText.setText(result[0]);
    }
    modifSecurite();
}

private void modifBase() {
    // Modifie la base suivant les caractères cochés
    base = "";
    if (minSwitch.isChecked()) {
        base += "portezcviuxwhskyaajgblndqfm";
    }
    if (majSwitch.isChecked()) {
        base += "THEQUICKBROWNFXJMPSVLAZYDG";
    }
    if (symSwitch.isChecked()) {
        base += "@#&!)-%;<:*$+=/?>(";
    }
    if (chiSwitch.isChecked()) {
        base += "567438921";
    }
}

@SuppressLint("SetTextI18n")
private void modifSecurite() {
    // Modifie la sécurité en fonction des paramètres cochés
    int len2 = longueurSeekBar.getProgress();
    int len = len2 * len2 + 3 * len2 + 10;
    longueurTextView.setText("Longueur : " + len);
    int nb_carac = base.length();
    int bits = (int) ((Math.round(Math.log(Math.pow(nb_carac, len)) / Math.log(2))));
    if (bits == 0) {
        securiteSeekBar.setProgress(bits);
    } else {
        securiteSeekBar.setProgress(bits - 32);
    }
    String[] result = securite(bits);
    securiteTextView.setText(result[0] + bits + " bits");
    securiteTextView.setTextColor(Color.parseColor(result[1]));
}

```

Application  
MainActivity.java

```

private String[] modification(String mot) {
    // Modifie le site et la clef en un mot de passe (mot = site + clef)
    int len = longueurSeekBar.getProgress();
    int len2 = len * len + 3 * len + 10;
    BigInteger code = new BigInteger(sha256(mot), 16);
    int nb_carac = base.length();
    String code2 = dec2Base(code, base).substring(0, len2);
    int bits = (int) ((Math.round(Math.log(Math.pow(nb_carac, code2.length()))) / Math.log(2)));
    String[] result = securite(bits);
    if (bits == 0) {
        code2 = "";
    }
    return new String[]{code2, result[0] + bits + " bits", Integer.toString(bits), result[1]};
}

private String[] securite(int bits) {
    // Renvoie la bonne couleur ainsi que la sécurité suivant le nombre de bits
    String secure;
    String color;
    if (bits == 0) {
        secure = " Aucune ";
        color = "#FE0101";
    } else if (bits < 64) {
        secure = " Très Faible ";
        color = "#FE0101";
    } else if (bits < 80) {
        secure = " Faible ";
        color = "#FE4501";
    } else if (bits < 100) {
        secure = " Moyenne ";
        color = "#FE7601";
    } else if (bits < 126) {
        secure = " Forte ";
        color = "#53FE38";
    } else {
        secure = " Très Forte ";
        color = "#1CD001";
    }
    return new String[]{secure, color};
}

private static String sha256(String mot) {
    // Modifie mot en un chiffre en hexadécimal suivant la fonction de hachage sha256
    try {
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        byte[] hash = digest.digest(mot.getBytes("UTF-8"));
        StringBuilder hexString = new StringBuilder();
        for (byte b : hash) {
            String hex = Integer.toHexString(0xff & b);
            if (hex.length() == 1) hexString.append('0');
            hexString.append(hex);
        }
        return hexString.toString();
    } catch (Exception ex) {
        throw new RuntimeException(ex);
    }
}

```

Application  
MainActivity.java

```

// MARK : Actions
@SuppressLint({"SetTextI18n", "ResourceAsColor", "ResourceType"})
public void questionChange(View view) {
    // TextView Message Informatif
    final TextView message = new TextView(MainActivity.this);
    message.setText(getString(R.string.info_questions));
    message.setTextSize(15);
    message.setTextColor(Color.parseColor(getString(R.color.colorText)));
    message.setPadding(50, 0, 50, 0);
    // Questions dans les CheckButtons
    String[] questions = {"nom de jeune fille de votre mère", "nom de votre premier animal de compagnie",
        "rue de votre maison d'enfance", "pas de question"};
    final boolean[] checkedItems = {false, false, false, false};
    // Création alert box
    AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this, R.style.AlertDialogCustom)
        .setView(message) // Connexion avec message
        .setTitle("Question personnelle :") // Titre
        .setNegativeButton("Annuler", null) // Bouton Annuler
        .setMultiChoiceItems(questions, checkedItems, new DialogInterface.OnMultiChoiceClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which, boolean isChecked) {
                if (checkedItems[0]) {
                    clefEditText.setHint("nom jeune fille mère");
                    clefEditText.setText("");
                    dialog.cancel();
                } else if (checkedItems[1]) {
                    clefEditText.setHint("nom premier animal de compagnie");
                    clefEditText.setText("");
                    dialog.cancel();
                } else if (checkedItems[2]) {
                    clefEditText.setHint("rue maison enfance");
                    clefEditText.setText("");
                    dialog.cancel();
                } else if (checkedItems[3]) {
                    clefEditText.setHint("mot clef");
                    clefEditText.setText("");
                    dialog.cancel();
                }
            }
        });
    builder.show(); // Monter Alert box
}

public void copierChange(View view) {
    java.lang.String code = motPasseEditText.getText().toString();
    if (!(code.length() == 0) && !(code.equals("Il manque des valeurs"))) {
        copier(code);
        Toast.makeText(MainActivity.this, "Mot de passe copié dans le presse-papier", Toast.LENGTH_LONG).show();
    } else {
        Toast.makeText(MainActivity.this, "Vous n'avez aucun mot de passe à copier", Toast.LENGTH_LONG).show();
    }
}

```

Application

MainActivity.java

```

public void checkChange(View view) {
    // Switch Changé
    generer();
}
private TextWatcher textWatcher = new TextWatcher() {
    //Text Watch : génère le mot de passe quand on a fini de modifier le texte
    @Override
    public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {
    }
    @Override
    public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
    }
    @Override
    public void afterTextChanged(Editable editable) {
        generer();
    }
};
private SeekBar.OnSeekBarChangeListener longueurListener = new SeekBar.OnSeekBarChangeListener() {
    @SuppressWarnings("SetTextI18n")
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        // Longueur Slider en changement
        generer();
    }
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
    }
    @Override
    public void onStopTrackingTouch(SeekBar longueur) {
    }
};
private SeekBar.OnSeekBarChangeListener securiteListener = new SeekBar.OnSeekBarChangeListener() {
    @SuppressWarnings("SetTextI18n")
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        // Sécurité Slider en changement
        int bits = securiteSeekBar.getProgress() + 32;
        String[] result = securite(bits);
        securiteTextView.setText(result[0] + bits + " bits");
        securiteTextView.setTextColor(Color.parseColor(result[1]));
    }
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
    }
    @SuppressWarnings("SetTextI18n")
    @Override

```

Application  
MainActivity.java



```

public void onStopTrackingTouch(SeekBar seekBar) {
    // Sécurité Slider une fois changée
    int bits = securiteSeekBar.getProgress() + 32;
    if (bits < 42) {
        longueurSeekBar.setProgress(0);
        minSwitch.setChecked(false);
        majSwitch.setChecked(false);
        symSwitch.setChecked(false);
        chiSwitch.setChecked(true);
    } else if (bits < 47) {
        longueurSeekBar.setProgress(0);
        minSwitch.setChecked(false);
        majSwitch.setChecked(false);
        symSwitch.setChecked(true);
        chiSwitch.setChecked(false);
    } else if (bits < 48) {
        longueurSeekBar.setProgress(0);
        minSwitch.setChecked(true);
        majSwitch.setChecked(false);
        symSwitch.setChecked(false);
        chiSwitch.setChecked(false);
    } else if (bits < 51) {
        longueurSeekBar.setProgress(0);
        minSwitch.setChecked(false);
        majSwitch.setChecked(false);
        symSwitch.setChecked(true);
        chiSwitch.setChecked(true);
    } else if (bits < 55) {
        longueurSeekBar.setProgress(0);
        minSwitch.setChecked(true);
        majSwitch.setChecked(false);
        symSwitch.setChecked(false);
        chiSwitch.setChecked(true);
    } else if (bits < 57) {
        longueurSeekBar.setProgress(0);
        minSwitch.setChecked(true);
        majSwitch.setChecked(false);
        symSwitch.setChecked(true);
        chiSwitch.setChecked(false);
    } else if (bits < 61) {
        longueurSeekBar.setProgress(0);
        minSwitch.setChecked(true);
        majSwitch.setChecked(true);
        symSwitch.setChecked(false);
        chiSwitch.setChecked(false);
    }
}

```

Application  
MainActivity.java

```

} else if (bits < 63) {
    longueurSeekBar.setProgress(0);
    minSwitch.setChecked(true);
    majSwitch.setChecked(true);
    symSwitch.setChecked(true);
    chiSwitch.setChecked(false);
} else if (bits < 66) {
    longueurSeekBar.setProgress(0);
    minSwitch.setChecked(true);
    majSwitch.setChecked(true);
    symSwitch.setChecked(true);
    chiSwitch.setChecked(true);
} else if (bits < 67) {
    longueurSeekBar.setProgress(1);
    minSwitch.setChecked(true);
    majSwitch.setChecked(false);
    symSwitch.setChecked(false);
    chiSwitch.setChecked(false);
} else if (bits < 72) {
    longueurSeekBar.setProgress(1);
    minSwitch.setChecked(false);
    majSwitch.setChecked(false);
    symSwitch.setChecked(true);
    chiSwitch.setChecked(true);
} else if (bits < 76) {
    longueurSeekBar.setProgress(1);
    minSwitch.setChecked(true);
    majSwitch.setChecked(false);
    symSwitch.setChecked(false);
    chiSwitch.setChecked(true);
} else if (bits < 80) {
    longueurSeekBar.setProgress(1);
    minSwitch.setChecked(true);
    majSwitch.setChecked(false);
    symSwitch.setChecked(true);
    chiSwitch.setChecked(false);
} else if (bits < 86) {
    longueurSeekBar.setProgress(1);
    minSwitch.setChecked(true);
    majSwitch.setChecked(true);
    symSwitch.setChecked(false);
    chiSwitch.setChecked(false);
} else if (bits < 88) {
    longueurSeekBar.setProgress(1);
    minSwitch.setChecked(true);
    majSwitch.setChecked(true);
    symSwitch.setChecked(true);
    chiSwitch.setChecked(false);
}

```

Application  
MainActivity.java

```

} else if (bits < 94) {
    longueurSeekBar.setProgress(1);
    minSwitch.setChecked(true);
    majSwitch.setChecked(true);
    symSwitch.setChecked(true);
    chiSwitch.setChecked(true);
} else if (bits < 95) {
    longueurSeekBar.setProgress(2);
    minSwitch.setChecked(true);
    majSwitch.setChecked(false);
    symSwitch.setChecked(false);
    chiSwitch.setChecked(false);
} else if (bits < 103) {
    longueurSeekBar.setProgress(2);
    minSwitch.setChecked(false);
    majSwitch.setChecked(false);
    symSwitch.setChecked(true);
    chiSwitch.setChecked(true);
} else if (bits < 109) {
    longueurSeekBar.setProgress(2);
    minSwitch.setChecked(true);
    majSwitch.setChecked(false);
    symSwitch.setChecked(false);
    chiSwitch.setChecked(true);
} else if (bits < 114) {
    longueurSeekBar.setProgress(2);
    minSwitch.setChecked(true);
    majSwitch.setChecked(false);
    symSwitch.setChecked(true);
    chiSwitch.setChecked(false);
} else if (bits < 115) {
    longueurSeekBar.setProgress(2);
    minSwitch.setChecked(true);
    majSwitch.setChecked(true);
    symSwitch.setChecked(false);
    chiSwitch.setChecked(false);
} else if (bits < 123) {
    longueurSeekBar.setProgress(2);
    minSwitch.setChecked(true);
    majSwitch.setChecked(false);
    symSwitch.setChecked(true);
    chiSwitch.setChecked(true);
} else if (bits < 126) {
    longueurSeekBar.setProgress(2);
    minSwitch.setChecked(true);
    majSwitch.setChecked(true);
    symSwitch.setChecked(true);
    chiSwitch.setChecked(false);
}

```

```

    } else {
        longueurSeekBar.setProgress(2);
        minSwitch.setChecked(true);
        majSwitch.setChecked(true);
        symSwitch.setChecked(true);
        chiSwitch.setChecked(true);}
    generer();
}
};
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Création menu
    super.onCreateOptionsMenu(menu);
    getMenuInflater().inflate(R.menu.menu_main, menu);
    if (darkMode){
        // Item light
        menu.findItem(R.id.darkmode).setIcon(R.drawable.light);
    }
    else {
        // Item night
        menu.findItem(R.id.darkmode).setIcon(R.drawable.night);
    }
    return true;
}
@SuppressLint("Assert")
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Action boutons menu
    switch (item.getItemId()) {
        case R.id.darkmode:
            // DarkMode
            darkMode = !darkMode;
            String message = setDarkMode();
            //Toast.makeText(MainActivity.this, message, Toast.LENGTH_LONG).show();
            break;
        case R.id.aide:
            // Aide lorsque bouton help est pressé
            final SpannableString s = new SpannableString(getString(R.string.info_app));
            Linkify.addLinks(s, Linkify.ALL);
            final AlertDialog d = new AlertDialog.Builder(this, R.style.AlertDialogCustom)
                .setTitle("Informations")
                .setPositiveButton(android.R.string.ok, null)
                .setMessage(s)
                .create();
            d.show();
            ((TextView) d.findViewById(android.R.id.message)).setMovementMethod(LinkMovementMethod.getInstance());
            break;
    }
}

```

Application

MainActivity.java

```

    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);
}
}

```

```

<?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"
                android:textColor="@color/colorPrimaryDark"
                android:onClick="questionChange"
                android:textSize="20sp" />
            </RadioGroup>
            <RadioGroup
                android:id="@+id/groupClef"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="20dp"
                android:layout_marginLeft="20dp"
                android:layout_marginTop="5dp"
                android:checkedButton="@+id/radio2"
                android:orientation="horizontal">

```

```

<TextView
    android:id="@+id/typedecodage"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Clef :"
    android:textColor="@color/colorText"
    android:textSize="20sp" />
<EditText
    android:id="@+id/clefEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:layout_weight="1"
    android:layout_marginRight="10dp"
    android:autofillHints=""
    android:hint="mot clef"
    android:maxLines="1"
    android:singleLine="true"
    android:textColor="@color/colorText"
    android:textColorHint="@color/colorText"
    android:backgroundTint="@color/gray"
    android:textSize="17sp"
    android:typeface="serif"
    android:inputType="textVisiblePassword" />
</RadioGroup>
<RadioGroup
    android:id="@+id/groupSite"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="20dp"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="5dp"
    android:checkedButton="@+id/radio2"
    android:orientation="horizontal">
    <TextView
        android:id="@+id/siteTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nom du site :"
        android:textColor="@color/colorText"
        android:textSize="20sp" />
    <EditText
        android:id="@+id/siteEditText"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:autofillHints=""

```

Application  
activity\_main.xml

```

        android:hint="nom du site"
        android:maxLines="1"
        android:singleLine="true"
        android:backgroundTint="@color/gray"
        android:textColor="@color/colorText"
        android:textColorHint="@color/colorText"
        android:textSize="17sp"
        android:typeface="serif"
        tools:ignore="LabelFor,TextFields" />
</RadioGroup>
<ImageView
    android:id="@+id/separate1"
    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/caracTextView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="20dp"
    android:layout_marginLeft="20dp"
    android:textStyle="normal|bold"
    android:layout_marginTop="10dp"
    android:text="Paramètres du mot de passe :"
    android:textColor="@color/colorText"
    android:textSize="20sp" />
<RadioGroup
    android:id="@+id/groupLongueur"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="20dp"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="5dp"
    android:checkedButton="@+id/radio2"
    android:orientation="horizontal">
    <TextView
        android:id="@+id/longueurTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Longueur : 20"
        android:textColor="@color/colorText"
        android:textSize="20sp" />

```

Application  
activity\_main.xml



```

        android:id="@+id/longueurSeekBar"
        android:theme="@style/TickMarkSeekBar"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:maxHeight="100000dp"
        android:minHeight="100000dp"
        android:focusable="true"
        android:thumb="@drawable/thumb"
        android:indeterminate="false"
        android:max="2"
        android:progress="2"
    />
</RadioGroup>
<TextView
    android:id="@+id/caracteresTextView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="20dp"
    android:layout_marginEnd="20dp"
    android:layout_marginTop="10dp"
    android:text="Caractères : "
    android:textColor="@color/colorText"
    android:textSize="20sp" />
<RadioGroup
    android:id="@+id/groupMajMin"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:layout_marginStart="20dp"
    android:layout_marginEnd="20dp"
    android:checkedButton="@+id/radio2"
    android:gravity="center"
    android:orientation="horizontal">
    <Switch
        android:id="@+id/minSwitch"
        android:layout_width="140dp"
        android:layout_weight="1"
        android:layout_height="20dp"
        android:background="#00FFFFFF"
        android:checked="true"
        android:text="Minuscules"
        android:textColor="@color/colorText"
        android:textSize="16sp"
        android:onClick="checkChange" />

```

```

<Switch
    android:id="@+id/majSwitch"
    android:layout_width="140dp"
    android:layout_height="1"
    android:layout_height="20dp"
    android:layout_marginStart="15dp"
    android:layout_marginLeft="15dp"
    android:background="#00FFFFFF"
    android:checked="true"
    android:text="Majuscules"
    android:textColor="@color/colorText"
    android:textSize="16sp"
    android:onClick="checkChange" />
</RadioGroup>
<RadioGroup
    android:id="@+id/groupSymChi"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:layout_marginStart="20dp"
    android:layout_marginEnd="20dp"
    android:checkedButton="@+id/radio2"
    android:gravity="center"
    android:orientation="horizontal">
    <Switch
        android:id="@+id/symSwitch"
        android:layout_width="140dp"
        android:layout_height="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_height="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" />
    </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:textColor="@color/colorText"
    android:textSize="18sp"
    android:typeface="serif"
    android:layout_marginStart="30dp"
    android:backgroundTint="@color/gray"
    android:digits=""
    android:layout_marginEnd="50dp">
</EditText>

<RadioGroup
    android:id="@+id/groupMotPasse"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="40dp"
    android:layout_marginLeft="30dp"
    android:layout_marginTop="-40dp"
    android:layout_marginEnd="20dp"
    android:layout_marginRight="30dp"
    android:checkedButton="@+id/radio2"
    android:gravity="right"
    android:orientation="horizontal">

```

```

        <Button
            android:id="@+id/copierButton"
            android:layout_width="35dp"
            android:layout_height="35dp"
            android:background="@drawable/clipboard"
            android:onClick="copierChange" />
    </RadioGroup>
    <RadioGroup
        android:id="@+id/groupSecurite"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:checkedButton="@+id/radio2"
        android:gravity="center"
        android:orientation="horizontal">
        <TextView
            android:id="@+id/securiteSimple"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="S curite :"
            android:textColor="@color/colorText"
            android:textSize="20sp" />
        <TextView
            android:id="@+id/securiteTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text=""
            android:textSize="20sp" />
    </RadioGroup>
    <SeekBar
        android:id="@+id/securiteSeekBar"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="50dp"
        android:layout_marginTop="10dp"
        android:layout_marginRight="50dp"
        android:thumb="@drawable/thumb"
        android:clickable="true"
        android:gravity="center"
        android:max="94"
        android:progress="94" />
</LinearLayout>
</ScrollView>

```