Droit et Conduite de Projet . 2020-2021

Faculté des Sciences et Techniques
Master 1 - Maths. CRYPTIS

COMPTE-RENDU

À l'attention de :
M. CRESPIN
M. DUSART
M. CONCHON

Rédigé par :
PIARD A.
JACQUET R.

Table des matières

1 Introduction et organisation

1.1 Introduction

Le but de notre projet était d'entraîner et de tester les compétences, en terme de programmation, de nos collègues du Master 1 Maths CRYPTIS. Comme énoncé dans notre fiche d'avant-projet, deux *Competitive Programming Test* étaient prévus, avec les quatre mêmes participants pour chacun d'eux, et nous avions deux protocoles organisationnelles qui s'offraient à nous en vue de la condition sanitaire actuelle. Le protocole ayant été retenu est celui du distanciel.

Le premier sujet a été traité le 1er février 2021 et le second le 8 février 2021.

1.2 Organisation

Le « recrutement » des participants s'est fait à distance. Depuis le début de l'année nous avons un groupe Messenger et un serveur Discord pour la promo master 1 maths CRYPTIS 2020/2021. Nous nous sommes d'ailleurs orientés à quasiment 100% sur le discord, ce qui est bien plus pratique. C'est d'ailleurs Arthur qui avait pris l'initiative de créer ce serveur discord. C'est donc par cet intermédiaire que nous avons fait une première requête de participants aux alentours du 8 octobre 2020. Nous avions lors de cette requête transmis notre fiche d'avant-projet ainsi que ce message :
Bonjour à tous,
Nous allons organiser deux épreuves (en distanciel à cause des circonstances actuelles) de programmation mélangeant la programmation "brutale" et la cryptographie. Pour ce faire nous avons besoin de huit étudiants désireux et intéressé par notre projet. Ces huit étudiants formeront deux groupes de quatre étudiants. Chaque groupe traitera un sujet. La durée de chaque épreuve est une heure. Ces épreuves seront organisées avant la fin de l'année. A l'issue de l'épreuve, nous récolterons :
- code/programme (même incomplet)
- notes divers, tous ce qui concerne vous et l'épreuve
Les données que nous récolterons seront utilisées, a posteriori, pour la rédaction du rapport.
Pour participer, et/ou si vous avez des questions merci de nous contacter à l'une de ces adresses :
arthur.piard@etu.unilim.fr ou raphael.jacquet@etu.unilim.fr ou directement sur Discord
Merci à tous ceux qui participeront, nous vous attendons nombreux.

Dans les faits, nous n'avons pas eux énormément de retours mais tout de même quelques intéressés. Pour ceux-là et pour d'autres personnes désireuses mais ne s'étant pas prononcé, nous avons organisé un vocal, toujours sur le discord, pour répondre à leurs éventuels questions et notamment pour les rassurer car certaines personnes n'étaient pas confiantes vis à vis de leurs compétences en programmation. Lorsque nous avons connus les noms de certains participants, Raphaël a créé un

autre serveur discord dans lequel il y a deux groupes de personnes : un groupe pour les participants et un groupe pour les administrateurs, c'est à dire nous. Sur ce discord il y avait différents salons. Un salon pour rappeler le but de notre projet, un salon pour prévoir d'un jour et d'un créneau horaire satisfaisant le plus de personnes possibles, un salon pour poster les sujets, un salon pour qu'ils postent leur travail en fin de session, un salon pour qu'ils nous posent des questions, etc, ainsi que deux salons vocaux, présents pour les débloquer s'ils avaient un souci lors de l'épreuve.
...

2 Premières réactions

À l'issue du premier *Competitive Programming Test*, nous nous sommes rendus compte que les étudiants ont mis beaucoup plus de temps que prévu sur les exercices, ce qui fait qu'aucun d'eux n'a terminé le premier sujet. Il se trouve que les exercices n'étaient pas triés par ordre de difficulté mais qu'ils ont tous choisis de commencer par le premier exercice.

Nous les avons légèrement questionnés et aidés à comprendre ce qui n'allait pas, en vue du second *Competitive Programming Test* qui était prévu la semaine suivante. Celui-ci s'est bien mieux passé. Les étudiants n'ont pas terminés le sujet mais ils ont tous traités au moins un exercice. La plupart des étudiants s'étant intéressés à deux exercices.

3 Analyse des sujets

3.1 Premier sujet

3.1.1 Premier exercice

L'objectif principal de ce premier exercice était de mettre l'étudiant face à un chiffré dont la méthode de chiffrement est inconnue. Il a été utilisé deux clés pour chiffrer le message. L'exercice en lui-même n'était pas difficile mais il fallait que l'étudiant tente diverses choses pour comprendre comment le message avait été chiffré. Le chiffrement en question est un chiffrement par substitution (de type César) pour lequel les clés étaient 3 et 13. Malgré les indications, la majorité des étudiants a perdu beaucoup de temps sur cet exercice, ce qui a influé sur l'ensemble du sujet. En effet un étudiant n'a pas pu effectuer l'exercice 2 et n'a pas trouvé comment résoudre l'exercice 1.

3.1.2 Second exercice

L'objectif de ce second exercice était de développer la méthode de déchiffrement d'un chiffrement précisément décrit. On rappelle que le choix du langage de programmation n'était pas imposé et que suivant le langage utilisé cela est plus ou moins rapide à écrire. Les étapes de l'algorithme étaient bien décomposées mais malgré tout nous n'avons pas eu beaucoup de retours, en raison du temps mal géré par les étudiants lors de l'exercice 1.

3.2 Second sujet

3.2.1 Premier exercice

L'objectif de cet exercice était de programmer l'algorithme de chiffrement écrit en français dans l'énoncé. Les seules difficultés étaient la maîtrise des opérateurs logiques et la conversion de type. Contrairement à nos attentes, cet exercice n'a pas été aussi bien réussi qu'attendu. Étant donné que l'épreuve a été conduite en distanciel, les étudiants avaient accès à toutes les ressources possibles. Malgré tout, certains n'ont pas su trouver comment effectuer un ET Logique.

3.2.2 Second exercice

L'objectif de cet exercice était d'écrire la méthode de déchiffrement à partir de celle qui a permit le chiffrement. Les opérations étaient donc à inverser. Si les étudiants possédaient la bonne fonction, ils leur étaient alors possible de déchiffrer le message. Ici aussi de nombreuses difficultés se sont présentées aux étudiants, comme le passage de tableaux de bits à une chaîne de caractères.

3.2.3 Troisième exercice

L'objectif de cet exercice était de déchiffrer un fichier binaire donné. Afin de déchiffrer ce fichier, il fallait trouver la clef de chiffrement qui était écrite sur l'image dont l'extension n'est pas la bonne. Il fallait donc au préalable trouver le bon format d'image avec la commande type xxd ou exiftool ou autres. Une fois la clef récupérée il faut la concaténer autant de fois que nécessaire pour obtenir une taille de clef similaire à la taille du fichier binaire. Enfin il suffisait d'effectuer un XOR entre le fichier binaire et la clef pour obtenir l'image finale. Cet exercice était volontairement légèrement plus difficile et comme attendu il n'a pas été traité.

4 Comparaisons entre étudiants

4.1 Sujet n°1

4.1.1 Premier exercice

```
import re

chiffre = "IPLU'QV\Ð 'Ç'[VP3'SL'TLZZHNL'JHJOÐ'LZ['A'HGT1GXJ'("

n = len(chiffre)
print(n)
```

Ici il est clair que l'étudiant n'a eu aucune idée de comment traiter l'exercice. Du moins il n'en a pas laissé de traces.

```
#!/usr/bin/python3

#EXERCICE 1
alph="abcdefghijklmnopqrstuvwxyzàçéèù"
échantillonfr="Quand, ..."
chif="IPLU'QVnÐ 'Ç'[VP3'SL'TLZZHNL'JHJOÐ'LZ['A'HGT1GXJ'("

def minuscul(texte):
        tmin=texte.lower()
        return [c for c in tmin if c in alph]

def anfreq(chaine):
        dicoccur={}
        for c in chaine:
                if c in dicoccur:
                        dicoccur[c]=dicoccur[c]+1
                else:
                        dicoccur[c]=1
        listocc=list(dicoccur.items())
        listocc=sorted(listocc,key=lambda t:t[1])
        listocc.reverse()
        return listocc

def replaUN(chaine,dicodéco):
        ecart=ord(dicodéco[0][1]-ord(dicodéco[0][0])
        for x in (0,len(chaine)):
                chaine[x]=chr(ord(chaine[x])+ecart)
        return chaine

```

```
# Appels des fonctions (main())
statsfr=anfreq(minuscul(échantillonfr))
statsch=anfreq(chif)
llcl=[t[0] for t in statsfr]
llch=[t[0] for t in statsch]
dicodage=list(zip(llch,llcl))
#print(dicodage)
chclUN=replaUN(chif,dicodage)
```

Cet étudiant a eu la bonne idée de faire de l'analyse fréquentielle depuis un texte écrit en français. Grâce à cette méthode il peut trouver une clef et reconstituer une partie du message. Ensuite il peut essayer de trouver la seconde clef, compte-tenu du fait qu'il connaît déjà une partie du message, en faisant le déchiffrement à la main. Cet étudiant utilise de nombreuses fonctions python élémentaires, il maîtrise la gestion de liste, les dictionnaires et les boucles.

```
#!/usr/bin/python3
alphabet = 'abcdefghijklmnopqrstuvwxyz'
# La fréquence d'apparution des lettres varie bien évidemment en fonction
# de la langue et du type de texte
# Pour un texte rédigé en français, on a :
liste_frequence=['e',' ','a','s','i','t','u','r','n'] #quelques lettres
#par ordre décroisant

chaine=input('chiffré : ')
chaine=chaine.lower()
chaine=[c for c in chaine if c in alphabet]
dico_occurences = {}

def occurence(chaine):
        for c in chaine:
            if c in dico_occurences:
                dico_occurences[c] = dico_occurences[c] + 1
            else:
                dico_occurences[c] = 1
        print(dico_occurences)

        liste_elements = list(dico_occurences.items())
        liste_elements = sorted(liste_elements,key=lambda t:t[1])
        print(liste_elements)
        dico=dict(liste_elements)
        liste_key=list(dico.keys())

        n=alphabet.index(liste_key[len(liste_key)-1])
        m=alphabet.index('e')
```

```
        if n >= m :
                decalage=n-m
        else:
                decalage=m-n
        return decalage

texte_clair=''
decalage=occurence(chaine)
decalage=26-decalage #pour déchiffrer
alphabet_dechiffrement = alphabet[decalage:] + alphabet[:decalage]
associations = dict(zip(alphabet,alphabet_dechiffrement.upper()))
for c in chaine:
    if c in associations :
        texte_clair += associations[c]
print("#############")
print("le message clair : ")
print(texte_clair.lower())
# je parviens pas à continuer !!!!!!
```

Cet étudiant a suivi le même raisonnement que le précédent. Son raisonnement arrive au même résultat et en allant un peu plus loin il aurait pu arriver au résultat final. Il maîtrise quand même lui aussi les listes, dictionnaires et les fonctions élémentaires de python.
On voit que seulement deux personnes sur quatre ont traités cet exercice ce qui est moins que ce que nous attendions.

4.1.2 Second exercice

```
#!usr/bin/python3
import re
A = """155c04970e1747172e4e070e044e17776e076526
0417474e27330474074e2767072e4e070e9726044f2f776e0407e704272f
0f042e76074e970e4f27047623"""
n = len(A)

def hexad(n): #fonction qui prend transforme un nombre en haxa et
 # le passe en entier
        h = 0
        for i in range(len(n)):
                if n[i] == 'a':
                        h += 10 * 16**(1-i)
                elif n[i] == 'b':
                        h += 11 * 16**(1-i)
                elif n[i] == 'c':
```

```
                        h += 12 * 16**(1-i)
                elif n[i] == 'd':
                        h += 13 * 16**(1-i)
                elif n[i] == 'e':
                        h += 14 * 16**(1-i)
                elif n[i] == 'f':
                        h += 15 * 16**(1-i)
                else:
                        h += int(n[i]) * 16**(1-i)
        return h

def bbinaire(x):
        b = str(bin(x))
        b = b[2:]
        n = len(b)
        if n<8:
                b = (8-n)*'0' + b
        return b

def echange(n):
        if n< 10000000 and n%10 == 1:
                n= n + 10000000 -1
        elif n>1000000 and n%10 == 0:
                n = n - 1000000 +1
        return n

B = []
m = len(B)

for i in range(0,n-1,2):
        B.append(bbinaire(hexad(A[i:i+2])))

#for j in range(m):
#mon but était de faire le chemin inverse avec mes fonctions

print(B)
```

Ici notre étudiant a créé des fonctions qui n'ont pas réellement d'intérêt puisqu'il existe des fonctions élémentaires en python qui permettent, par exemple, de passer d'un entier à sa représentation hexadécimale et vice-versa. La fonction bbinaire permet d'utiliser des octets. Quant à la fonction echange, elle, n'a aucun intérêt. L'étudiant a quand même essayé de faire quelque chose mais ça reste très loin du résultat final.

```
#!/usr/bin/python3
import sys , re
liste1 = []
liste2 = []
message = """155C04970E1747172E4E070E044E17776E0
765260417474E27330474074E2767072E4E070E9726044F2F776e040
7e704272f0f042e76074e970e4f27047623"""

p = re.compile('\w\w')
liste1 = p.findall(message)

for lettre in liste1 :
        liste2 += bin(int(lettre, 16))

print (liste2)
```

Cet étudiant a utilisé les expressions régulières pour récupérer deux à deux les caractères du chiffrés. Il les convertit ensuite en binaire.

```
#!/usr/bin/python3
message = input("Entrez le message à envoyer? : ")
l=len(message)
liste=[]
def gest_caractere(message):
        blocs=[[message[2*i:2*i+2]] for i in range(0,l//2)]
        #on construit des blocs de 2 elements
        print(blocs)

        for i in range(0,len(blocs)):
                liste.append(blocs[i][0])
        print(liste)
        liste_bin=[bin(int(liste[i], 16)) for i in range(0,len(liste))]
        print(liste_bin)
        #le temps ??????
gest_caractere(message)
```

Cet étudiant a eu la bonne idée de découper le message en blocs de deux caractères, il les stocke dans une liste qu'il convertit ensuite en une liste de nombre binaires. Les lignes 10 à 12 n'ont aucun intérêt. Il pouvait totalement travailler avec la liste blocs. Toutefois il n'aura effectuer que deux opérations sur la dizaine demandée.

L'objectif de cet exercice était pourtant clair, mais il a dû susciter beaucoup d'incompréhensions vis à vis de nos étudiants qui ont utilisés beaucoup de fonctions

inutiles.

4.2 Sujet n°2

4.2.1 Premier exercice

```
#!usr/bin/python3
import os, sys
phrase = 'Il fait beau ce matin.'
phrase = phrase.upper()
PH = []
for u in phrase:
        PH.append(ord(u))
def premf(A):
        for i in range(len(A)):
                if A[i]%3 == 0:
                        A[i] += (i+13)%i
        return A
premf(PH)
ch = ""
for i in range(len(PH)):
        ch += chr(PH[i])

CH = bytes(ch, 'utf-8')
ME = []
for i in CH:
        if i%2 ==0:
                ME.append(i^5)
        elif i%3==0:
                ME.append(i|6)
        elif i%5==0:
                ME.append(i&2)
        else:
                ME.append(i)

for i in range(len(ME)):
        ME[i] = chr(ME[i])

mes_fin = ''.join(ME)
print(mes_fin)

# j'obtiens II%CI%GI%CW%MaI[+
```

Cet étudiant a bien comprit le but de cet exercice. Il utilise dans le bon ordre les différentes fonctions et a comprit de manière générale les étapes à suivre. Toutefois

il a fait des erreurs. Il n'a pas utilisé les bons opérateurs logiques pour obtenir le bon chiffré.

```
#!/usr/bin/python3
#Exercice 1
message = 'Il fait beau ce matin.'
msg = message.upper()
L = []
for lettre in msg:
    L.append(ord(lettre))
print(L)
L_prime = []
for nombre in L:
    if int(nombre)%3 == 0:
        L_prime.append((int(nombre)+13)%int(nombre))
    else:
        L_prime.append(nombre)
print(L_prime)
#subsidiaire
#chaine = ''
#for nombre in L_prime:
#    c = chr(nombre)
#    print(c)
#    chaine += c
#subsidiaire
L_bin = []
print(bin(67))

for nombre in L_prime:
    b = bin(nombre)
    if int(nombre)%2 == 0:
        L_bin.append(b[2:len(b)]^101)
    elif int(nombre)%3 == 0:
        L_bin.append(b[2:len(b)] or 110)
    elif int(nombre)%5 == 0:
        L_bin.append(b[2:len(b)] and 10)
    else:
        L_bin.append(b[2:len(b)])

print(L_bin)
# NON FINI
```

Cet étudiant a lui aussi bien suivit l'ordre de l'algorithme défini, mais il n'a pas terminé son code. Il avait bien comprit le but de l'exercice et connaissait ses opérateurs logiques. Toutefois il a fait des erreurs comme la modification des octets

suivant la parité de l'index du tableau.

```
#!/usr/bin/python3
import sys
message = "Il fait beau ce matin."
message = message.upper()
tableau=[]

for i in range(0, len(message)) :
        tableau.append(ord(message[i]))
        if(tableau[i]%3==0):
                tableau[i]+= (13+i)%i
for i in range(0, len(message)-1) :
        if(i%2==0):
                tableau[i] = tableau[i]^5
        if(i%3==0):
                newl=""
                l = bin(tableau[i])
                for j in range(0, len(l)-3):
                        newl += l[j]
                newl +="11"
                newl += l[len(l)-1]
                tableau[i] = int(newl, 2)
        if(i%5==0):
                newl=""
                l = bin(tableau[i])
                for j in range(0, len(l)-3):
                        newl += l[j]
                newl +="0"
                newl += l[len(l)-2]
                newl +="0"
                tableau[i] = int(newl, 2)
l = ""
for i in range(0, len(message)-1) :
        l += chr(tableau[i])

print(l) #JL%FDHV BO@U'CW"HAfIZ
```

Cet étudiant a presque le bon résultat. Sa méthodologie est bonne mais ses opérations logiques sont mal effectuées. En effet, il faut les faire avec des bits. La chaîne "110" n'est pas le bit "110". Avec un peu plus de rigueur et d'attention cet étudiant aurait obtenu le bon chiffré. Il est quand même important de souligner que tout le reste est correct.

```python
#!/usr/bin/python3

message='il fait beau ce matin'
message=message.upper()
#print(message)
tableau=[ord(i) for i in message]
print(tableau)
for i in range(0,len(tableau)):
        if tableau[i]%3==0:
                tableau[i]=tableau[i]+(13+i)%i
#print(tableau)
tableau_caract=[chr(i) for i in tableau]
print(tableau_caract)
chaine=''.join(tableau_caract)
print(chaine)
tableau_oct=[bin(ord(i))[2:] for i in chaine]
print(tableau_oct[0])
for i in range(0,len(tableau_oct)):
        if tableau[i]%2==0:
                tableau[i]=tableau[i]^int('101',2)
        if tableau[i]%3==0:
                tableau[i]=tableau[i]|int('110',2)
        if tableau[i]%5==0:
                tableau[i]=tableau[i]&int('010',2)
#print(tableau)
chaine_f=[chr(i) for i in tableau]
chaine_f=''.join(chaine_f)
print(chaine_f)
```

Cet étudiant a une excellente méthodologie, il a bien suivit l'énoncé. La seule erreur notable est la mauvaise écriture des opérateurs logiques en python. En dehors de cette erreur, le code est totalement correct.

```python
#!/usr/bin/python3
#EXERCICE 1
txt="Il fait beau ce matin."
#1
TXT=txt.upper()
#2 et 3
tab=[]

for i in range(0,len(txt)):
        tab.append(ord(txt[i]))
        if tab[i]%3==0:
```

```
                tab[i]=tab[i]+((13+i)%i)
#4
rst=""

for i in range(0,len(tab)):
        rst=rst+chr(tab[i])
#5
loct=list(map(bin,rst.encode('utf-8')))
#6
def tabmodif(ta):é
        for i in range(0,len(ta)):
                if i%2==0:
                        ta[i]=ta[i]^101
                if i%3==0:
                        ta[i]=ta[i] (ou inclusif) 110
                if i%5==0:
                        ta[i]=ta[i] (et) 010
        return ta
#7
#Si fait avec la liste du 3)
rslt=""

for i in range(0,len(tab))
        rslt=rslt+chr(tab[i])

#Si fait avec liste du 5)
rsltat=""
for i in range(0,len(loct)):
        rsltat=rsltat+bytes([int(loct[i],2)]).decode('utf-8')
```

Cet étudiant était sur la bonne piste mais il n'a pas été au bout de la réflexion et a manqué de rigueur sur la rédaction du code. On remarque qu'il ne connaît ni les opérateurs logiques ni la conversion chaîne de caractères vers bits. Même après "correction" du code, le résultat n'est pas du tout celui attendu. Les légères corrections apportées étaient nécessaires pour pouvoir exécuter le programme. Autrement, les lignes de code de l'étudiant n'ont pas été modifiées.

4.2.2 Second exercice

```python
#!usr/bin/python3
import os, sys
ph="FC?E^FK?YZ^J?XZ?B^KFM-"
S = bytes(ph,'utf-8')
A = []
for u in S:
        A.append(chr((u^21)-10))
phrase = ''.join(A)
print(phrase)
# résultat = IL FAIT BEAU CE MATIN.
```

Cet étudiant a réussi l'exercice. Il a même prit l'initiative d'utiliser le xor entre entiers, ce qui fonctionne avec python. Il obtient le bon résultat ce qui est satisfaisant.

```python
#!/usr/bin/python3
#Exercice 2
message = 'FC?E^FK?YZ^J?XZ?B^KFM-'
L = []
for lettre in message:
    L.append(ord(lettre) + 10)
print(L)
msg = ''
for i in L:
    msg += chr(i)
print(msg)
# NON FINI
```

Cet étudiant n'a effectué que les trois premières étapes, sûrement par faute de temps. Il était toutefois sur la bonne voie. Les premières lignes de codes sont justes.

```python
#!/usr/bin/python3
import sys
message = "FC?E^FK?YZ^J?XZ?B^KFM"
dechiffre = ""
for i in range(0, len(message)) :
        dechiffre += chr((ord(message[i])^21)-10)
print(dechiffre) #IL FAIT BEAU CE MATIN
```

Cet étudiant a également réussi l'exercice et de façon encore plus compacte. Il n'y a pas grand chose à redire, sa maîtrise du XOR entre entiers est intéressante.

```
1  #Chiffrement:
2  chif='FC?E^FK?YZ^J?XZ?B^KFM-'
3  tabl=[]
4  chaine=""
5  for i in range(0,len(chif)):
6          tabl.append(ord(chif[i])+10)
7          chaine=chaine+chr(tabl[i])
8  lisoct=list(map(bin,chaine.encode('utf-8')))
9  for i in range(0,len(lisoct)):
10         lisoct[i]=lisoct[i]^10101
11 retout=""
12 for i in range(0,len(lisoct)):
13         retour=retour+bytes([int(lisoct[i],2)]).decode('utf-8')
14 #Déchiffrement
15 byt=list(map(bin,chif.encode('utf-8')))
16 inter=""
17 clair=""
18 for i in range(0,len(byt)):
19         byt[i]=byt[i]^10101
20         inter=inter+bytes([int(byt[i],2)]).decode('utf-8')
21 t=[]
22 for i in range(0,len(inter)):
23         t.append(ord(inter[i]))
24         t[i]=t[i]-10
25         clair=clair+chr(t[i])
```

Cet étudiant a essayé de coder la méthode de chiffrement, ce qui était inutile. Peut-être espérait-il que ça l'aiderait. Ceci étant, sa méthode de chiffrement ne marche pas. En effet, il a tenté d'effectuer un XOR entre deux types différents à la ligne 10.

Pour le déchiffrement il a effectué la même erreur, c'est à dire un XOR entre deux types différents (à la ligne 19 donc sa méthode ne marche pas. Cela dit, hormis cette erreur importante, le reste du programme semble correct.

4.2.3 Troisième exercice

A notre plus grand regret cet exercice n'a pas été traité. Cela aurait été un bon moyen de départager les étudiants et c'est celui qui ressemblait le plus à un CTF (Capture The Flag).

4.3 Conclusion

Les prémices de ce projet ont été de choisir des étudiants du Master Maths. CRYPTIS exclusivement. Notre choix s'est porté sur des étudiants avec des profils très différents et chacun avec leurs spécificités. Il aurait été trop facile de choisir des étudiants de la filière informatique du Master CRYPTIS.

Il est clair que les étudiants ont été moins productif que ce nous attendions. Mais, a posteriori, lorsque l'on compare ce qu'ils ont fait en ce début de semestre 2 et aujourd'hui avec les travaux pratiques auxquels nous assistons, il y a une différence notable.

En effet, leur niveau pour la majorité est à la hausse. Cela s'explique par un conditionnement et une pratique plus régulière de l'algorithmique générale. La matière Sécurité TIC qui utilise les notions de **Python** vues au premier semestre et l'utilisation de nouvelles bibliothèques comme **OpenSSL** n'est pas un problème pour eux. Ils arrivent beaucoup mieux à suivre les étapes et à traduire les énoncés en lignes de code. Nous avons beaucoup discutés avec les étudiants testés après les **CPT**.

En leur expliquant les subtilités et les cheminements d'idées nécessaire à la bonne résolution de tel ou tel exercice nous pensons fortement avoir augmenté leur niveau en programmation ce qui était le but recherché.

Nous avons apprécié ce type de projet en autonomie. Avec la situation sanitaire plus que complexe durant ce second semestre nous avons dû nous organiser autrement et faire les épreuves en distanciel. Cela n'a pas été idéal pour nous comme pour les étudiants mais nous sommes satisfait du résultat. Comme conclusion plus globale nous pouvons tirer comme enseignement que les étudiants ayant fait peu voir jamais d'informatique (orientée programmation) pendant leur cycle de Licence seront nécessairement plus en difficulté dans le cursus Math. CRYPTIS. Nous recommandons donc aux futurs promotions de travailler, seul, ou bien avec toutes les ressources qu'Internet propose, leur base en programmation.

Nous vous remercions pour votre œil attentif et espérons que vous aurez autant apprécié que nous le récit de notre projet.

4.3.1 Conclusion de Raphaël

4.3.2 Conclusion d'Arthur