

# Rapport TP1 - Sécurité Informatique

## INF4420A



POLYTECHNIQUE  
MONTRÉAL

Quentin COURREAU – 1973362

Rafael BOBAN – 1973338

## PARTIE A

### Question 1 - Entropie

A)

```
[raboba@l4712-27 Source - Entropie - Chiffrement] $ ./texte 200 > entropie/q1
[raboba@l4712-27 Source - Entropie - Chiffrement] $ cat entropie/q1
ROBERT OF ARTOYS BANISH T THOUGH THOU BE FROM FRAUNCE THY NATIUE COUNTRY YET WITH VS THOU SHALT RET
AYNE AS GREAT A SEIGNIORIE FOR WE CREATE THEE EARLE OF RICHMOND HEERE AND NOW GOE FORWARDS WITH OUR
P[raboba@l4712-27 Source - Entropie - Chiffrement] $ ./texte 200 > entr
[raboba@l4712-27 Source - Entropie - Chiffrement] $ ./h-lettre < entropie/q1
(space) = 37
A = 13
B = 3
C = 4
D = 3
E = 21
F = 6
G = 4
H = 12
I = 8
J = 0
K = 0
L = 2
M = 2
N = 9
O = 16
P = 1
Q = 0
R = 17
S = 7
T = 17
U = 7
V = 1
W = 5
X = 0
Y = 5
Z = 0
Nombre total de caracteres : 200
Entropie de l'entree : 3.948063
```

b) **1er Théorème de Shannon**

1. Efficacité maximum d'un code compresseur est approximativement égale à  $H(S)$
2. Il existe un code compresseur (sans erreur) avec efficacité  $H(S) + 1$

Chaque symbole émis par  $S$  peut être codé individuellement avec en moyenne  $H(S)$  bits.

Le calcul de l'entropie d'une source de messages donne une mesure de l'information minimale que l'on doit conserver afin de représenter ces données sans perte. On appelle cela compression.

- c) L'entropie d'une lettre (en moyenne) d'un fichier quelconque qui aurait été généré de la même façon mais avec les mêmes probabilités ( $1/27$  donc **caractères équiprobables**) pour chacun des 27 symboles (majuscules et espaces) est d'environ :  $\log_2(27) = 4.75$  ...En d'autres termes, **la redondance est maximale** lorsque tous les caractères ont la même probabilité d'apparition. L'entropie exprimant le **principe d'incertitude**, ceci confirme donc nos dires.

- d) On a  **$3.94/4.75 = 0.831$**

Plus le quotient des deux entropies est proche de 1, plus les deux entropies sont proches. D'après la définition du 1er théorème de Shannon donné plus haut, on en déduit que ce rapport entre deux entropies correspond à un taux de compression.

La **redondance** peut être définie comme étant le nombre de contraintes imposées sur un texte. Dans une langue naturelle telle que l'anglais, cela cause une diminution globale de l'entropie. Par exemple, la règle qui dit qu'un "q" doit toujours être suivi d'un "u" est une des dépendances qui fait de l'anglais une langue plus redondante. Les règles de grammaire, les parties d'un discours et le fait que nous ne pouvons pas inventer des mots de manière arbitraire rendent également l'anglais redondant. De plus, comme chaque lettre de l'anglais possède une fréquence qui lui est propre et non équiprobable, l'entropie se voit significativement réduite. En fait, l'entropie est encore plus faible, car il existe des corrélations entre un caractère et celui, voire ceux qui le précèdent ( $N$  et  $N-1$ ). On appelle cela des  $n$ -grammes et il a été observé, que plus  $n$  est grand plus l'entropie diminue et tend vers l'entropie moyenne de la langue étudiée. Tous ces éléments expliquent pourquoi à la question **a)** l'entropie calculée est plus faible que l'entropie calculée en **c)**.

e) En générant un texte avec la source lettre, on obtient une entropie de **4.08**.

La différence entre les deux entropies est donc de **4.08 - 3.94 = 0.14** qui est donc non significative car inférieure à **0.4**.

```
[raboba@l4712-27 Source - Entropie - Chiffrement] $ ./lettre 200 > entropie/qle
[raboba@l4712-27 Source - Entropie - Chiffrement] $ ./h-lettre < entropie/qle
(space) = 18
A = 18
B = 3
C = 6
D = 7
E = 21
F = 3
G = 4
H = 12
I = 11
J = 1
K = 0
L = 8
M = 3
N = 22
O = 12
P = 3
Q = 0
R = 12
S = 10
T = 17
U = 1
V = 0
W = 4
X = 0
Y = 4
Z = 0
Nombre total de caracteres : 200
Entropie de l'entree : 4.081542
```

f) L'entropie ne tient pas compte des mots anglais dans le texte mais seulement de la fréquence des caractères dans le texte. La fréquence des caractères pour les deux sources étant relativement proches, l'entropie des deux textes se retrouve également proche. A noter que plus le texte est long, plus la fréquence des caractères qui le compose se rapproche des fréquences moyennes de chaque caractère de la langue anglaise. Les deux entropies résultantes tendront donc vers un résultat infinitésimalement proche.

## Question 2 - Histogramme

a)

```
[raboba@l4712-27 Source - Entropie - Chiffrement] $ ./lettre 200 > histogramme/qa
[raboba@l4712-27 Source - Entropie - Chiffrement] $ ./cesar < histogramme/qa > histogramme/lettreCe
sarChiffree
[raboba@l4712-27 Source - Entropie - Chiffrement] $ cat histogramme/lettreCesarChiffree
VRXL QP OG ZUP U H GU DXH BDXF KIFH RL RWWVRQRWVFW GDRRHWWHDGKHR RL RKFWMLAFDVLDRDXUWFLVODDUKDXFD
HDGLRUQDRVOPHHXU VOGKZ XRGZKDKRE VHE GH HXUZJQWVLQV KGH HU QWKVHHHH HMQWUXKR HWXROZJ RJGHE E QHV
[raboba@l4712-27 Source - Entropie - Chiffrement] $ ./cesar-d < histogramme/lettreCesarChiffree
SQUI NM LD WRM R E DR AUE YAUC HFCE OI OTTSNOTSCT DA00ETTEADHEO OI OHCTJRIXCASIOAURTCISNAARHAUCA
EADIORNAOSLMEEUR SLDHW UODDWHAHOB SEB DE EURWGNTSINS HDE ER NTHSEEEE EJNTRUHO ETUOLWG OGDED B NES
```

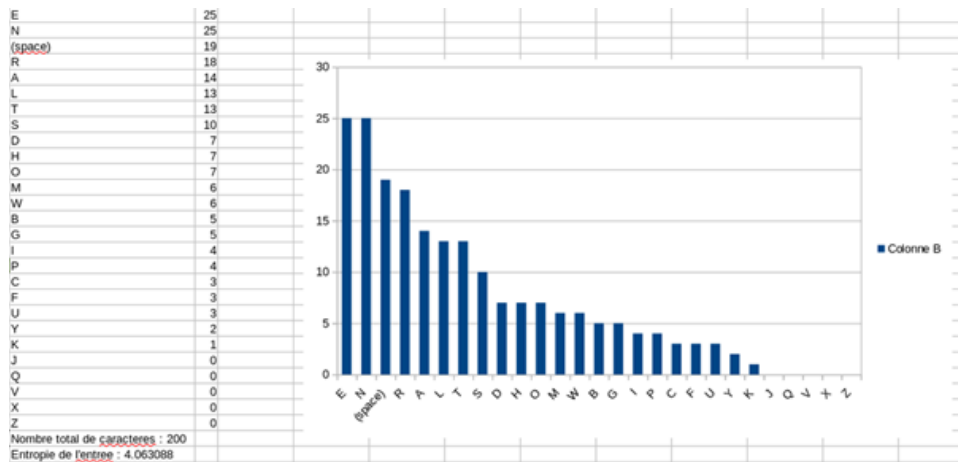
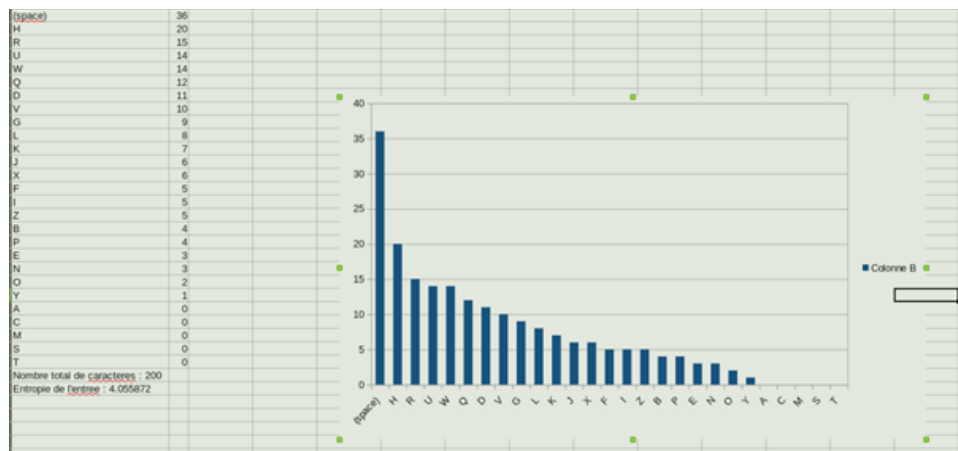
```

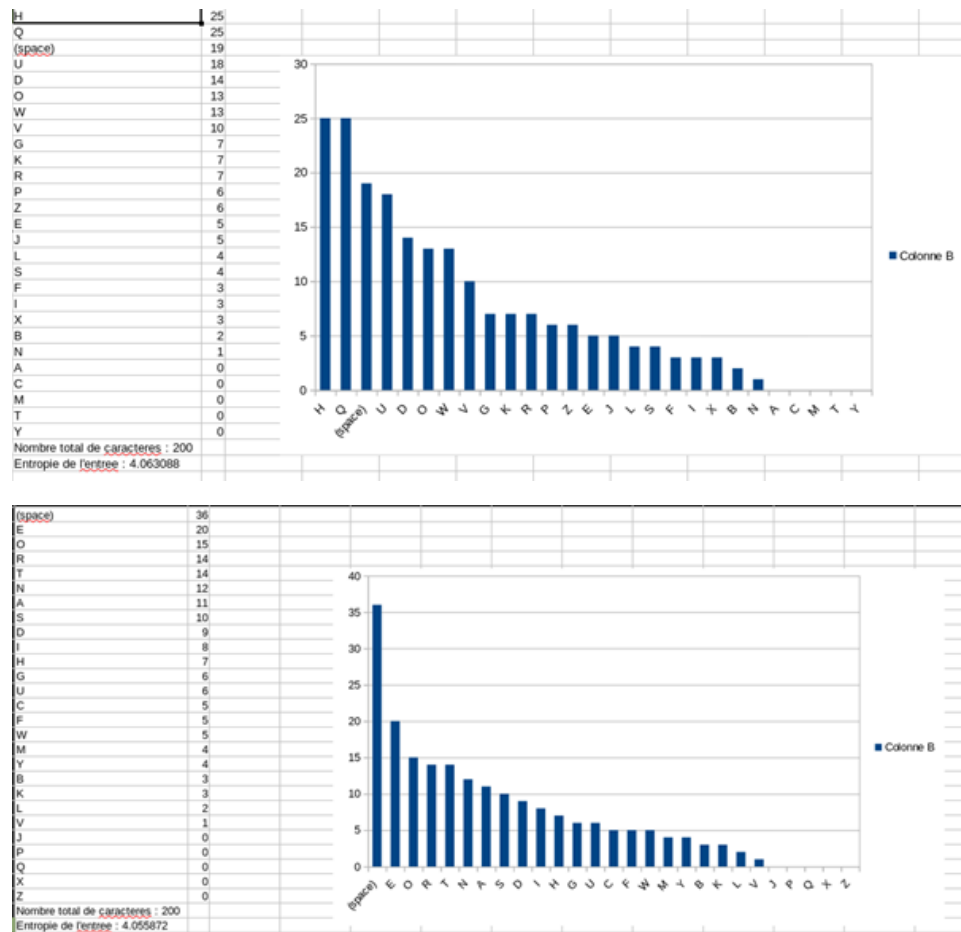
[raboba@l4712-27 Source - Entropie - Chiffrement] $ ./texte 200 > histogramme/qat
[raboba@l4712-27 Source - Entropie - Chiffrement] $ ./cesar < histogramme/qat > histogramme/texteCe
sarChiffree
[raboba@l4712-27 Source - Entropie - Chiffrement] $ cat histogramme/qat
TANDS THE LEAGUE BETWEEN THE SCOT AND VS MO CRACKT AND DISSEUERED MY RENOWNED LORD THE TREACHEROU
S KING NO SOONER WAS INFORMDE OF YOUR WITH DRAWING OF YOUR ARMY BACKE BUT STRAIGHT FORGETTING OF HI
[raboba@l4712-27 Source - Entropie - Chiffrement] $ cat histogramme/texteCesarChiffree
WDQGV WKH OHDJXH EHWZHHQH WKH VFRW DQG YV PR FUDFNW DQG GLVVHXHUHG PB UHQRZQHG ORUG WKH WUHDFKHURX
V NLQJ QR VRRQHU ZDV LQIRUPGH RI BRXU ZLWK GUDZLQJ RI BRXU DUPB EDFNH EXW VWUDLJKW IRUJHWWLQJ RI KL
[raboba@l4712-27 Source - Entropie - Chiffrement] $ ./cesar-d < histogramme/texteCesarChiffree
TANDS THE LEAGUE BETWEEN THE SCOT AND VS MO CRACKT AND DISSEUERED MY RENOWNED LORD THE TREACHEROU
S KING NO SOONER WAS INFORMDE OF YOUR WITH DRAWING OF YOUR ARMY BACKE BUT STRAIGHT FORGETTING OF HI

```

On voit bien que pour la source **texte**, le texte après chiffrement est différent du texte original et après déchiffrement on retrouve le texte d'origine.

b)





Ce chiffrement est donc rudimentaire et est très loin des chiffrements modernes qui respectent bien mieux le principe d'incertitude et de confusion.

c) Nous remarquons que les quatre histogrammes sont pratiquement identiques. La raison est que l'on a appliqué un chiffrement **mono-alphabétique** sur un authentique texte anglais (./texte) et sur un texte généré à l'aide des fréquences de lettres de la langue anglaise. Ce chiffrement, très simple, substitue une lettre donnée par une autre lettre (toujours la même). Les fréquences sont donc identiques mais simplement décalées dans l'histogramme. On remarque d'ailleurs que le décalage appliqué vaut trois.

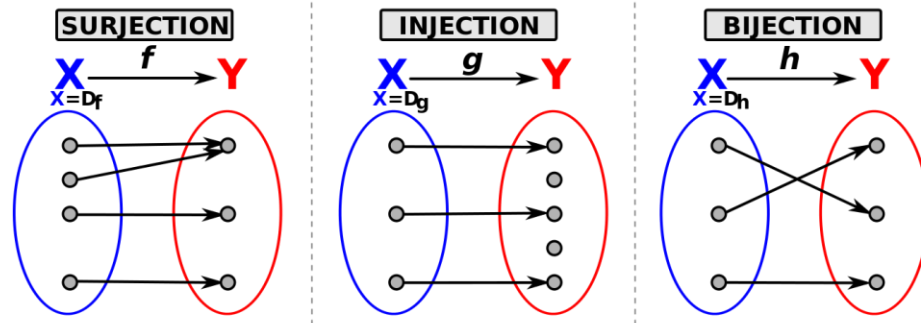


Figure 1.1 - Schema d'une bijection

En mathématiques, il s'agit d'une **application bijective**.

*“Une application est **bijective** si et seulement si tout élément de son ensemble d'arrivée a un et un seul antécédent, c'est-à-dire est image d'exactly un élément (de son domaine de définition).”*

Un digramme ou bigramme est une association de 2 caractères, généralement 2 lettres, leur fréquence d'apparition permet d'obtenir des informations sur un message. Si les fréquences individuelles ne suffisent pas, il est parfois judicieux de compter le nombre de digrammes.

L'analyse résultante montrerait des fréquences bien plus faibles par rapport à une analyse de fréquences individuelles. Pour une analyse de trigrammes, les fréquences seraient encore plus faibles. (car fréquence d'apparition plus rare dépendante de l'**ordre** des lettres du digramme ou trigramme). De manière générale pour **n-gramme**, plus n est grand, plus la probabilité d'apparition de ce n-gramme est faible. De plus, les fréquences des digrammes sont plus équitablement réparties.

Dans le cas de **./texte** et si les fréquences sont comptabilisées sur deux lettres à la fois, l'histogramme résultant montrerait de hautes fréquences pour les digrammes présents dans langue anglaise. (th, he, etc.).

En revanche, dans le cas de lettre, les lettres étant tirées aléatoirement selon leur fréquence d'apparition dans la langue anglaise, elle ne conserve par conséquent pas l'ordre, propriété intrinsèque d'un digramme et plus généralement d'un n-gramme. Par conséquent, il est plus fortement probable de retrouver deux lettres aléatoires dans les regroupements et donc, les lettres avec une forte probabilité d'apparition comme 't', 'o' ou 'e', seront plus fréquentes. Ainsi, le digramme 'ee' serait nettement plus fréquent dans **./lettre** que dans **./texte**, car il y a peu de mots dans la langue anglaise ayant deux fois le caractère 'e' de suite. Dans l'autre cas, le digramme 'th' sera plus fréquent dans texte que dans lettre, car 'th' revient très souvent dans la langue anglaise en raison de sa nature grammaticale. En conclusion, ce qu'il

faut retenir ici est que l'utilitaire **./lettre** produit des lettres qui sont tirées "aléatoirement" selon les fréquences de lettres de la langue anglaise, et ne conserve donc pas l'**ordre** des lettres qui est une propriété intrinsèque d'un digramme véritable, ou plus généralement d'un n-gramme.

d) Cette méthode facilite effectivement le processus de déchiffrement d'un texte **véritable** car les digrammes donnent des informations précieuses sur la langue d'un texte. Par exemple, on sait que les digrammes les plus présents dans la langue française sont respectivement : **ES, LE, DE**. Source :

<http://www.bibmath.net/crypto/index.php?action=affiche&quoi=chasseur/frequences/francais>

En anglais, ce sont **TH, HE, IN**. Les digrammes permettent donc de donner des informations supplémentaires sur un texte et de potentiellement en déduire sa langue.

De plus, si les espaces entre les mots sont conservés (ce qui est le cas présentement) on peut faciliter le processus de cryptanalyse, notamment en regardant :

- Dans un mot, les doublets les plus fréquents
- Les mots de deux lettres les plus fréquents
- Les mots de trois lettres les plus fréquents (si analyse par trigramme)

Enfin, il nous semble important de préciser que plus le texte est **long**, plus la cryptanalyse est précise et donc le processus de déchiffrement facilité.

Selon nous, l'analyse des digrammes sur le texte généré par l'utilitaire **./lettre** ne donne aucune information tangible puisque les lettres sont simplement tirées "aléatoirement" selon les fréquences de lettres de la langue anglaise, et ne conserve donc pas l'**ordre** des lettres qui est une propriété intrinsèque d'un digramme, ou plus généralement d'un n-gramme.

### Question 3 – Masque jetable

a)

```

quentin@quentin-Lenovo-ideapad-510S-13IKB: ~/Documents/inf4420a/Utilitaires TP2/Utilitaires TP1/
quentin@quentin-Lenovo-ideapad-510S-13IKB:~/Documents/inf4420a/Utilitaires TP2/Utilitaires
TP1/Source - Entropie - Chiffrement$ ./monnaie 1024 > masqueJetable/monnaie1024.bin
quentin@quentin-Lenovo-ideapad-510S-13IKB:~/Documents/inf4420a/Utilitaires TP2/Utilitaires
TP1/Source - Entropie - Chiffrement$ ./h-bit < masqueJetable/monnaie1024.bin
0 = 4072
1 = 4120
Nombre total de bits : 8192
Entropie du texte entre : 0.999975
quentin@quentin-Lenovo-ideapad-510S-13IKB:~/Documents/inf4420a/Utilitaires TP2/Utilitaires
TP1/Source - Entropie - Chiffrement$ ./h-ascii < masqueJetable/monnaie1024.bin
Nombre total d'octets : 1024
Entropie de l'entree : 7.768644
quentin@quentin-Lenovo-ideapad-510S-13IKB:~/Documents/inf4420a/Utilitaires TP2/Utilitaires
TP1/Source - Entropie - Chiffrement$ ./binaire 1024 > masqueJetable/binaire1024.bin
quentin@quentin-Lenovo-ideapad-510S-13IKB:~/Documents/inf4420a/Utilitaires TP2/Utilitaires
TP1/Source - Entropie - Chiffrement$ ./h-bit < masqueJetable/binaire1024.bin
0 = 5076
1 = 3116
Nombre total de bits : 8192
Entropie du texte entre : 0.958304
quentin@quentin-Lenovo-ideapad-510S-13IKB:~/Documents/inf4420a/Utilitaires TP2/Utilitaires
TP1/Source - Entropie - Chiffrement$ ./h-ascii < masqueJetable/binaire1024.bin
Nombre total d'octets : 1024
Entropie de l'entree : 0.827864

quentin@quentin-Lenovo-ideapad-510S-13IKB:~/Documents/inf4420a/Utilitaires TP2/Utilitaires
TP1/Source - Entropie - Chiffrement$ ./monnaie 1024 > masqueJetable/cle1024.bin
quentin@quentin-Lenovo-ideapad-510S-13IKB:~/Documents/inf4420a/Utilitaires TP2/Utilitaires
TP1/Source - Entropie - Chiffrement$ ./masque masqueJetable/cle1024.bin 1024 masqueJetabl
e/monnaie1024.bin masqueJetable/monnaie1024Chiffre.bin
quentin@quentin-Lenovo-ideapad-510S-13IKB:~/Documents/inf4420a/Utilitaires TP2/Utilitaires
TP1/Source - Entropie - Chiffrement$ ./h-bit < masqueJetable/monnaie1024Chiffre.bin
0 = 4049
1 = 4143
Nombre total de bits : 8192
Entropie du texte entre : 0.999905
quentin@quentin-Lenovo-ideapad-510S-13IKB:~/Documents/inf4420a/Utilitaires TP2/Utilitaires
TP1/Source - Entropie - Chiffrement$ ./h-ascii < masqueJetable/monnaie1024Chiffre.bin
Nombre total d'octets : 1024
Entropie de l'entree : 7.805379

quentin@quentin-Lenovo-ideapad-510S-13IKB:~/Documents/inf4420a/Utilitaires TP2/Utilitaires
TP1/Source - Entropie - Chiffrement$ ./masque masqueJetable/cle1024.bin 1024 masqueJetabl
e/binaire1024.bin masqueJetable/binaire1024Chiffre.bin
quentin@quentin-Lenovo-ideapad-510S-13IKB:~/Documents/inf4420a/Utilitaires TP2/Utilitaires
TP1/Source - Entropie - Chiffrement$ ./h-bit < masqueJetable/binaire1024Chiffre.bin
0 = 4189
1 = 4003
Nombre total de bits : 8192
Entropie du texte entre : 0.999628
quentin@quentin-Lenovo-ideapad-510S-13IKB:~/Documents/inf4420a/Utilitaires TP2/Utilitaires
TP1/Source - Entropie - Chiffrement$ ./h-ascii < masqueJetable/binaire1024Chiffre.bin
Nombre total d'octets : 1024
Entropie de l'entree : 7.833862

```

b) On observe que l'entropie (calculée par blocs de 8 bits avec h-ascii) de binaire a fortement augmenté, et est pratiquement égale de celle de monnaie.

c) On remarque que l'entropie pour les deux fichiers générés est sensiblement la même. La différence entre les deux entropies est effectivement inférieure à 0.4 donc **non significative**. En conclusion, même si l'entropie de la source est imparfaite/faible, le chiffrement **ONE TIME PAD assure** une entropie maximale/parfaite. Et ceci semble tout à fait logique, puisque la clé utilisée est aléatoire et de taille égale au message à chiffrer. Peu importe que la source soit parfaitement déterministe avec une entropie très faible puisque chaque caractère sera à terme chiffré de manière aléatoire. Sans conteste, il s'agit dans les deux cas d'une méthode sécuritaire de chiffrement !



## Question 4 – Analyse de risque

a) Il n'y a pas de réponse exacte car elle dépend du contexte et d'une variable particulièrement importante dans ce cas précis qui est : le **temps**. Pour répondre à cette question il nous faut donc la reformuler. Combien de temps faut-il rester sur le site – que ce soit le site A ou le site B – afin de rentabiliser au mieux l'achat de ce dernier ? On calcule pour cela l'espérance de perte égale à :  $E = P * I$  où  $P$  = Probabilité d'incident et  $I$  = Impact d'incident. On a :  $E = 0.25 * 100000 = 25000$ . Par rapport au site A, on établit le ratio-années :  $500\ 000 / 25\ 000 = 20$  **années**. On privilégiera le site B sur une période de 20 années pour maximiser les profits de l'entreprise, et avant que le risque d'un ouragan croît.

b)

i) Principalement intégrité,

ii) Disponibilité,

iii) Confidentialité

c) Pour chacun des scénarios, nous utiliserons la méthodologie **MEHARI**. Dans ce type de méthodologie, il nous faut considérer probabilité et impact ensemble. En effet, la perception du risque peut être affectée, notamment parce que le cerveau humain a tendance à surévaluer l'entité impact (biais psychologique). D'après le cours, le calcul de la probabilité s'écrit comme suit : **Probabilité = (Capacité \* Opportunité \* Motivation) / 3**. Ce calcul décrit une fonction monotone. Le risque quant à lui est calculé de la façon suivante : **Risque = Impact \* Probabilité**. En découlent les tableaux suivants :

Scénario 1 :

Acteur	Capacité	Opportunité	Motivation	Probabilité	Impact	Risque
Tricheur	4	4	4	4	2	8
C.O	1	4	1	2	2	4
Concurrents	2	4	2	2.66	2	5.32

Pour le premier contexte, l'agent tricheur présente le plus gros risque pour l'entreprise.

Scénario 2 :

Acteur	Capacité	Opportunité	Motivation	Probabilité	Impact	Risque
Tricheur	1	4	1	2	4	8
C.O	4	4	1	3	4	12
Concurrents	2	4	4	3.3	4	13.2

Pour le second contexte, l'agent "Concurrents" présente le plus gros risque pour l'entreprise.

### Scénario 3 :

Acteur	Capacité	Opportunité	Motivation	Probabilité	Impact	Risque
Tricheur	1	3	1	1.66	3	4.97
C.O	4	3	4	3.66	3	10.98
Concurrents	1	3	2	2	3	6

Enfin, pour le troisième et dernier contexte, l'agent "Crime Organisé" présente le plus gros risque pour l'entreprise.

Ce qu'il faut retenir ici, est que même si les agents restent constants, le scénario lui, change constamment. Il faut donc sans cesse évaluer/réévaluer les menaces selon un contexte précis.

d)

1) En réponse immédiate à notre succès, le paramètre **motivation** de l'agent **concurrent** augmenterait de manière inéluctable. Une réaction en chaîne s'en suivrait et par conséquent, la probabilité d'une attaque à l'encontre de notre entreprise augmenterait, ce qui augmenterait également le risque. (Car Risque = Probabilité \* Impact).

2) En réponse au refus de payer des pots-de-vin réclamés par la mafia locale, cette dernière aurait une motivation d'autant plus grande. Plusieurs scénarios peuvent-être pensés :

- En premier lieu, la mafia locale pourrait faire pression sur le patron notamment grâce aux milliers d'ordinateurs compromis, ce qui pourrait nuire à la réputation de l'entreprise.
- Également, la motivation du crime organisé augmenterait, d'une part pour assurer que leur crédibilité reste intacte (notamment aux yeux des autres mafias) mais également par fierté (biais psychologique). De plus, le refus de payer des pots-de-vin de la part du patron peut entraîner une diminution des capacités de la mafia locale sur le **long terme**. Néanmoins, sur une **courte période et en réponse immédiate**, dans une tentative désespérée, le crime organisé pourrait augmenter ses **capacités** d'attaques et de pression afin d'inciter et contraindre le patron à continuer de payer des pots-de-vin.

3) Globalement, les trois composantes, capacité, opportunité et motivation serait affectée côté tricheur. La composante "capacité" diminuerait car le tricheur se fera prendre plus facilement et devra mettre en œuvre de nouveaux moyens afin de contourner le système de détection. La motivation sera évidemment affectée et l'opportunité de tricher réduite.

Cependant, sur le long terme, le scénario peut différer car de nouvelles méthodes de triche – outils plus efficaces – peuvent émerger. Les composantes “capacité” et “opportunité” augmenteraient alors et la motivation d’antan retrouvée.

e)

Acteur	Capacité	Opportunité	Motivation	Probabilité	Impact	Risque
Tricheur	1	1	1	1	3	3
C.O	4	1	4	3	3	9
Concurrents	1	1	2	1.33	3	4

Baisse du risque globale mais à quel prix ? Nous ne parlons pas du prix monétaire ici, mais bel et bien de l’action de déléguer une tâche aussi importante à une entreprise située dans un pays étranger. Les concurrents et le C.O pourraient en profiter pour manipuler nos nouveaux partenaires qui pourraient donner des renseignements et faire de l’espionnage industriel sur notre système, nos activités etc. Les mentalités, les lois n’étant pas les mêmes, notre capacité à mettre en place une telle grille n’est-elle pas biaisée ? Y’a t-il trop de paramètres à prendre en compte ? La délégation mais à quel prix ? Au final, réduit-on vraiment le risque ? Ou est-ce notre perception et le sentiment d’avoir fait une bonne affaire qui nous induit en erreur ? J’ai le sentiment qu’avec cette offre nous prenons plus de risques qu’on ne les réduit, et ce avec le sentiment de bien faire, de TROP BIEN faire. Le jeu n’en vaut clairement pas la chandelle.

## Partie B

### Question 1 - Codage [/1.25]

#### Cas où le codage est inchangé :

a) Les différents alphabets sont :  $s = \{0-9\}$  ;  $t = \{0-9\}$  ;  $t' = \{0,1\}$ . Ce sont des ensembles.

b) Langage L1 : Arrangement de quatre symboles de l’alphabet  $s$ .

Langage L2 :  $m.m$  avec  $m$  mots du premier langage.

Langage L3 : Bloc de taille 64 bits composé de 0 et 1,  $t'$ .

#### c) Interception d’un message chiffré sur le réseau :

- Attaque par force brute de type “texte chiffré seulement” sans pouvoir chiffrer des messages. On teste toutes les possibilités de clés soient  $2^{56}$  pour DES.
- Attaque texte connu, accès au texte en clair en plus du texte chiffré (Rappel que 10 000 combinaisons pour un NIP)

### Accès à une boîte noire :

- Attaque par texte clair choisi : capacité de choisir le texte en clair, et accès à une boîte noire qui chiffre chaque message avec la clé qu'on recherche.
- Attaque par texte chiffré choisi : Accès à plusieurs texte chiffrés et une boîte noire qui déchiffre chaque message avec la clé qui a permis de chiffrer (Cas symétrique -> DES)

-> Attaque par rejeu

De plus, l'énoncé nous dit qu'un attaquant peut intercepter et même modifier les messages chiffrés. Dans une telle situation un attaquant en profitera pour introduire son propre NIP afin d'accéder au compte de la victime, sans jamais avoir à déchiffrer un quelconque message.

On note également l'absence de l'application d'une fonction de hachage sur le chiffré.

d)

La boîte noire nous permettrait d'effectuer les attaques décrites plus haut en exploitant un grand nombre d'entrées, et en observant les sorties résultantes. Par tâtonnement et analyse, on arriverait à terme à déceler des motifs/patterns utiles pour reconstituer la clé.

```
troisO@qucou:/mnt/c/Users/Quentin/Documents/es  
/Utilitaires TP1/Codage$ ./transBase.py 1234  
!S4yEtroisO@qucou:/mnt/c/Users/Quentin/Documen  
sUtilitaires TP1/Codage$ ./transBase.py 1234  
!S4yEtroisO@qucou:/mnt/c/Users/Quentin/Documen  
  
troisO@qucou:/mnt/c/Users/Quentin/Docume  
TP1/Codage$ ./transBase.py 1234 > nipB  
troisO@qucou:/mnt/c/Users/Quentin/Docume  
TP1/Codage$ ./recepBase.py < nipB  
1234troisO@qucou:/mnt/c/Users/Quentin/Do
```

### Cas où le codage change :

e)

```

troisO@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF442
/tps/tp1/Utilitaires TP2/Utilitaires TP1/Codage$ ./trans1.py 1234 > nip1
troisO@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF442
/tps/tp1/Utilitaires TP2/Utilitaires TP1/Codage$ ./recep1.py < nip1
1234troisO@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/IN
/tps/tp1/Utilitaires TP2/Utilitaires TP1/Codage$ ./trans2.py 1234 > nip2
troisO@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF442
/tps/tp1/Utilitaires TP2/Utilitaires TP1/Codage$ ./recep2.py < nip2
Delaï de transmission suspect, operation annulee
troisO@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF442
/tps/tp1/Utilitaires TP2/Utilitaires TP1/Codage$ ./trans3.py 1234 > nip3
troisO@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF442
/tps/tp1/Utilitaires TP2/Utilitaires TP1/Codage$ ./recep3.py < nip3
Delaï de transmission suspect, operation annulee

```

f) Premièrement et commun aux trois codages ; Après analyse des programmes trans1.py, trans2.py et trans3.py, les bits de padding sont générés pour chacun des trois codages selon un algorithme qui est **déterministe** et s'avère être potentiellement **réversible**. Explication : nous avons en fait connaissance des deux bits de padding. Pour les deux premiers codages, cela représente 2 bits sur 64 bits. Ratio théorique de  $2/64 = 3.125\%$ . Dans le cas du codage 3, nous connaissons donc 4 bits sur les 64, ce qui représente un ratio théorique de  $4/64 = 6.25\%$ . Or, cela présente déjà une faille de sécurité permettant des attaques fréquentielles et/ou des attaques à texte clair connu. Il aurait fallu générer ces bits de padding de façon **aléatoire**.

L'algorithme commence par convertir le NIP décimal en format binaire de taille quatorze. Ensuite il convertit en chaîne de caractère et la découpe en deux parties de taille égale (=7). Il compte alors le nombre de '1' pour chacune de ces deux parties et effectue un modulo 2. Si le nombre de '1' est pair, alors un bit à '0' sera ajouté, sinon si le nombre de '1' est impair, un bit à '1' sera ajouté. On réalise très vite qu'avec la connaissance de la boîte noire et de l'algorithme utilisé qu'il est très facile de générer 10 000 combinaisons, d'appliquer le même algorithme de padding, d'observer quels nombres vérifient ces critères, et de filtrer en conséquence afin de réduire l'espace de nombres candidats. Tout ceci nous donne des informations précieuses et utiles pour notre cryptanalyse. En effet, ces deux bits de padding nous permettent d'isoler le NIP à un ensemble de nombres plus petit inférieur à 10 000. Nous savons donc maintenant qu'un algorithme déterministe donne par essence une certaine quantité d'information, ce qui, en d'autres termes signifie que l'on réduit l'entropie du système, et donc sa sécurité globale. L'algorithme employé ici possède une certaine logique dans l'ajout et la création de ces bits de padding, et c'est cette logique sous-jacente qui produit intrinsèquement une quantité d'informations, ce qui réduit l'entropie et est du pain bénît pour un attaquant.

Dans le cas du codage 2, selon le nombre de clients qui modifient leur code NIP, les chances de collisions ne sont en fait pas si faibles. En effet, en considérant le nombre de bits alloués aux deux premiers champs, la concaténation de ces deux éléments produit une chaîne binaire de taille théorique **32 bits**. Donc, au mieux, les chances de collisions sont de l'ordre de  $2^{32}$  pour deux clients actifs. Comme l'unicité est uniquement basé sur un nombre de 32 bits, somme du nouveau NIP avec un nombre aléatoire (le timestamp étant

déterministe), si deux messages sont envoyés dans la même milliseconde, alors les chances d'obtenir une collision pendant cette milliseconde sont de  **$2^{32}$**  (c'est-à-dire qu'elles sont identiques ou presque deux fois plus élevées que le risque total d'obtenir une collision si l'on envoie  $2^{32}$  messages avec des informations aléatoires de 96 bits sans timestamps). En fait, les chances de collisions sont en pratique plus élevées car comme indiqué dans l'énoncé, un attaquant peut connaître parfaitement le fonctionnement des boîtes de codage. Il peut connaître les bits de parité puisqu'ils ne sont à priori pas aléatoires. Bien sûr, si la probabilité que deux messages soient envoyés au cours de la même milliseconde est suffisamment faible (et qu'on peut le garantir, même en cas d'attaque), l'ajout d'un timestamp rendrait effectivement la tâche encore plus ardue pour un attaquant. Je le conçois, mais je ne compterais généralement pas dessus. Un scénario peut être pensé : Un attaquant peut forcer les clients à utiliser le même horodatage (timestamp) de sorte que le risque de collision augmente, pour en tirer un avantage. Dans ce cas précis, un nonce purement aléatoire serait en effet le meilleur choix.

De plus, mon impression est que le codage 2 est en fait même plus enclin à des attaques par rejeu que le codage 1 qui utilise un nonce aléatoire de 48 bits. En effet, le codage 2 utilise seulement 16 bits pour générer un nombre aléatoire ce qui représente deux caractères. En prenant en compte la table ASCII comme source d'entrée, cela représente  $2^{16} = 256^2 = \mathbf{65536}$  combinaisons. La génération d'un NIP est très simple à calculer et est de l'ordre de :  $10^4 = \mathbf{10\ 000}$  **combinaisons**. Nous avons donc  $65536 + 10000 = \mathbf{75536}$  **combinaisons** au total. Or, d'après l'énoncé et nos analyses, les 2 bits de parités ne sont pas aléatoires et peuvent donc être potentiellement connus à l'avance, tout comme le timestamp qui est prédictible. De plus, d'après l'énoncé, *le « timestamp » est utilisé par la banque pour invalider les messages dont le timestamp est trop vieux*. Mais qu'est-ce que trop vieux ? Est-il possible pour un attaquant de générer ces 75536 combinaisons avant que le délai fixé par la banque ne soit épuisé ? Tout dépend du délai !

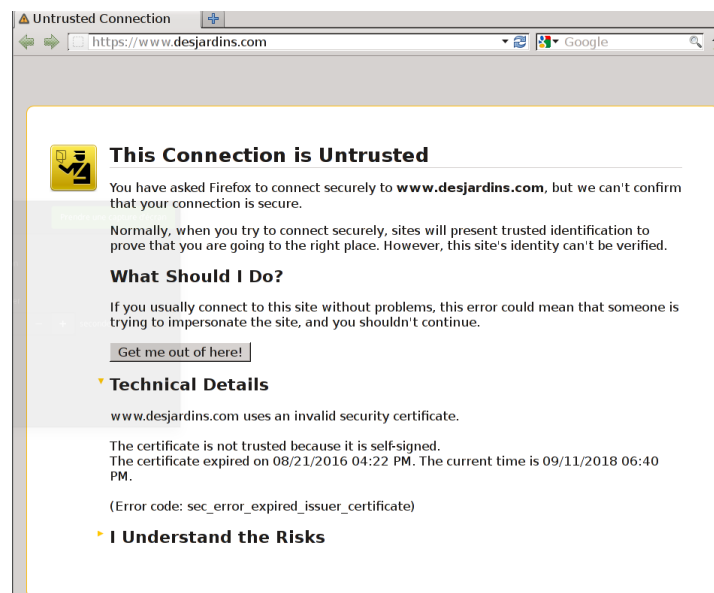
Le codage 1 possède un nombre aléatoire de 48 bits. On calcule le nombre de combinaisons :  $2^{48} = \mathbf{281474976710656}$  **combinaisons** (les 10000 combinaisons NIP sont négligeables ici), ce qui représente un nombre bien plus grand de combinaisons que dans le cas du codage 2 précédent et offre selon moi la meilleure des sécurités.

Le codage 3 ne semble pas fiable car si le système de transmission est originellement vulnérable (d'après l'énoncé) et donc potentiellement compromis, il est alors possible qu'un attaquant connaisse l'ancien NIP utilisé. De plus, étant donné qu'il a potentiellement connaissance du fonctionnement de la boîte de codage, il sait qu'un timestamp de 32 bits est utilisé (déterministe), que l'ancien NIP+2 bits de parité est utilisé (potentiellement connu donc déterministe) et que le nouveau NIP+2 bits est utilisé (10000 combinaisons

seulement). Le codage 3 présente selon moi un caractère trop déterministe qui est aux antipodes du principe de confusion et de chaos recherché en cryptographie.

## Question 2 - Certificats à clé publique, HTTPS et SSL [/1]

a) Nous ne pouvons nous y connecter car une alerte de sécurité est générée par Firefox. En effet, le serveur utilise un certificat auto-signé et non pas un certificat émis et vérifié par une Autorité de Certification de confiance. En réponse, la plupart des navigateurs recommandent aux visiteurs de quitter la page pour des raisons de sécurité.



b) Le site d'une compagnie bancaire telle que **Desjardins** se doit d'utiliser des **certificats émis et vérifiés par une Autorité de Certification de confiance** (tiers). Il est donc paradoxal que le site d'une telle institution use d'un certificat auto-signé qui nuit à sa réputation et affecte la confiance de ses clients.

c) Firefox affiche maintenant le site sans alerte de sécurité car nous avons ajouté une exception. Le certificat est donc maintenant dans la liste des certificats de confiance.



d)

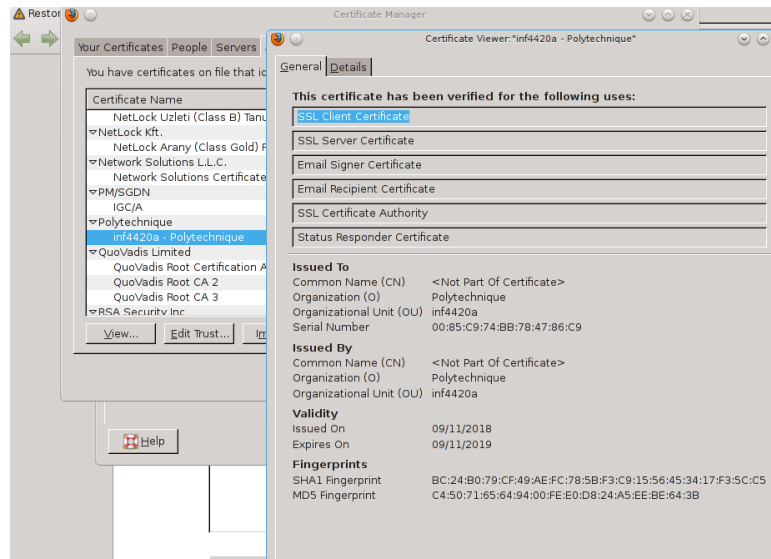
## Création du certificat de l'autorité de certification

On génère une clé d'une taille de 4096 bits en utilisant le chiffrement triple DES. Cette dernière nécessite une **passphrase** qui sera demandée à chaque utilisation de notre clé privée et qui signera tous les certificats que l'on émettra. Ensuite, on génère le certificat auto-signé de type x509 en usant de la clé précédemment construite. On attribue au certificat une durée de 365 jours, soit une année. Le certificat n'étant pas signé par une autorité de certificat de confiance, le navigateur affichera une anomalie comme celle vu plus haut. C'est notre certificat d'autorité de certification qui va permettre de signer les certificats créés.

```
openssl genrsa -des3 -out ca.key 4096
```

```
openssl req -new -x509 -days 365 -key ca.key -out ca.crt
```





Pour vérifier le bon usage du certificat :

```

~ : bash
File Edit View Bookmarks Settings Help
admin_web@certificates ~ $ openssl x509 -in ca.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            85:c9:74:bb:78:47:86:c9
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=CA, ST=Quebec, L=Montreal, O=Polytechnique, OU=inf4420a/emailAddress=quentin.courrea
u@polymtl.ca
        Validity
            Not Before: Sep 11 23:06:36 2018 GMT
            Not After : Sep 11 23:06:36 2019 GMT
        Subject: C=CA, ST=Quebec, L=Montreal, O=Polytechnique, OU=inf4420a/emailAddress=quentin.courrea
u@polymtl.ca
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            Public-Key: (4096 bit)
            Modulus:
                00:b7:65:1d:5f:a4:ee:6c:a0:b6:34:2d:de:d0:a6:
                42:aa:7c:68:2f:17:af:66:c8:56:07:8b:3c:10:71:
                e1:74:bd:13:03:a9:f8:de:fe:7a:64:d8:47:6c:00:
                ab:3f:81:03:52:f1:76:52:d8:c8:23:b8:da:71:5f:
                d2:d7:61:b3:9e:df:e5:8d:3d:c5:08:98:71:b1:f8:
                09:f8:25:75:0b:a0:1d:50:e3:1c:98:62:2d:4f:a3:
                e0:96:18:fd:58:57:c1:38:f4:79:7b:4a:10:8f:25:
                b5:aa:29:3d:00:4e:c2:7d:14:8e:35:6e:00:41:28:
                67:20:a7:c9:4d:71:38:49:9a:e5:71:c6:98:40:e4:
                c0:3d:bd:a8:8c:ed:b4:1d:3b:5f:c3:ff:ea:7c:f6:

```

## Création d'un fichier de demande de signature de certificat (CSR Certificate Signing Request)

***openssl genrsa -des3 -out desjardins.key 4096***

***openssl req -new -key desjardins.key -out desjardins.csr***

Ceci a pour effet de créer le formulaire de demande de certificat (fichier desjardins.csr) à partir de notre clé privée préalablement créée.

## La signature du certificat serveur par le CA (Certificate Authority)

Signature de la requête :

***openssl x509 -req -days 365 -in desjardins.csr -CA ca.crt -CAkey ca.key -set\_serial 01 -out desjardins.crt***

Le certificat signé est le fichier "desjardins.crt".

La sortie de la commande est la suivante :

```
admin@webcertificates ~$ openssl x509 -req -days 365 -in desjardins.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out desjardins.crt
Signature ok
subject=/C=CA/ST=Quebec/L=Montreal/O=desjardins/OU=bank/CN=www.desjardins.com/emailAddress=dj@dj.com
Getting CA Private Key
Enter pass phrase for ca.key:
unable to load CA Private Key:
140253973636776:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt:evp_enc.c:467:
140253973636776:error:0906A065:PEM routines:PEM_do_header:bad decrypt:pem_lib.c:476:
admin@webcertificates ~$ openssl x509 -req -days 365 -in desjardins.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out desjardins.crt
Signature ok
subject=/C=CA/ST=Quebec/L=Montreal/O=desjardins/OU=bank/CN=www.desjardins.com/emailAddress=dj@dj.com
Getting CA Private Key
Enter pass phrase for ca.key:
admin@webcertificates ~$ ls
Desktop  ca.crt  ca.key  desjardins.crt  desjardins.csr  desjardins.key
admin@webcertificates ~$
```

Voici le contenu de notre certificat desjardins.crt ;

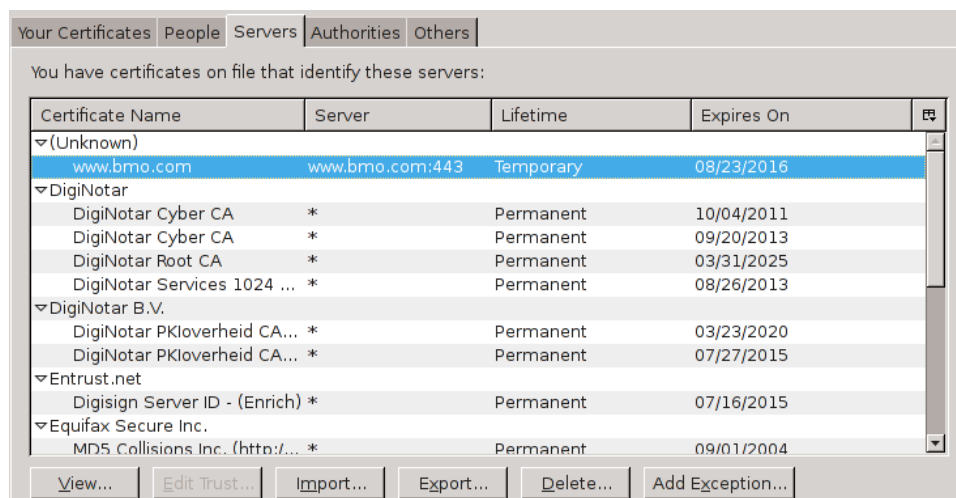
```
-----BEGIN CERTIFICATE-----
MIIFijCCA3ICAQEwDQYJKoZIhvcNAQEFBQAwYgxCzAJBgNVBAYTAkNBMQ8wDQYD
VQIQIDAZRdWViZWmXETAPBgNVBACMCElbnRyZWZfMRyWFAyDQVQKDA1Qb2x5dGVj
aG5pcXVlMREwDwYDVQQLDAhpbmYONDIwYTEqMCGCSqGSIb3DQEJARYbcXVlbnRp
bi5jb3VycmVhdUBwb2x5bXRSLmNhMB4XDTE4MDkxODE5MzcwM1oXDTE5MDkxODE5
MzcwM1owYwxCzAJBgNVBAYTAkNBMQ8wDQYDVQIQIDAZRdWViZWmXETAPBgNVBACM
CElbnRyZWZfMRMwEQYDVQKDApKZXNqYXJkaW5zMzQwYwYDVQQLDARiYW5rMRsw
GQYDVQKDBDJ3d3cuZGVzZmFyZGlucy5jb20xGDAWBgkqhkiG9w0BCQEWcWRqGRq
LmNvbTCCAiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBAKTs/fgjV4sTczAy
2hgVLpokXDtEPNqf3EakEsybLSqm8idVgjTz2MF86g6iUMCHQTZYzP3VTotHdPok
vimdUT6fTWuVotPHGy/E+UqVy1cJOM36+FhfkadCsQYe8zj5hY+tCLNoG3z+emIm
fXf2+WM/2R2aYqYaJvVof2dhNW966Hln3loDnZ6avccqVv79UVvQeT00QEmKHlO
UALUvIob9b3ypus8l6lvvKiAAioZKYnZJlGwUVOXMcZUAjYL8xslJl1B/ZUPKaeG
wWJ8DBzr9AFncLVmCgNkKm6HUHJldlvXuhClm9BawGPT6qtVR4r4Yh6D0xMhN6Q
L5oGp9xCH6kYVdlqidYWJ6LntzF0oUVerC5DZpCHFFe3cgoeBbNMNwXoR5248ggy
r/7UlpZYL5IJRMjmYowtthyJ38nY3VsCtPmm4tfBzNld9PY8GADUoeIqnuTpSeb7
46x9R7pVrGR9ZphPWIgc3zsuRIF9N454GbNQ8oAonxPQLrIX8jVmffZthkunJv05
khyZ2oBgBXnYhQecMAFnFVn0bAw8wim6aDCLjTkXRlXzmcZbivNtMjmsPnz95WpO
PLc0Jw8nvloGrQgceukdgpydW6Go0s0BWRx00L/tSlaWNPvn+GhFiS4kXuFGaGr6
LQju/7MyNsB3oPZUJlrg40dF8hL5AgMBAEwDQYJKoZIhvcNAQEFBQADggIBAGxk
4q/vOkcnjSc8hEJGlPKJw7m0oqa2FCDesc3aTZJufEHp38cCnREST4882J4JNVli
3epBORhdutDL2sHAoz6p83c+mQ+TuXICpfQgbcrdNiHecUFR/4NzzEzVaJ6rhh+J
s3u+IGeXLsa0fLIfoRyis0r8EAf9v00GpipLHo8ATcp12JCs7SZig77oj9ANysxq
mLhNWM/nDVMxh2jpTxzK7+0trZPCudpamZEWf79PF9IvQvJNiEsmgoaxS10Ifkcv
t+TBndklmjFwqzSCE5kZ7ksdcCsFNNwr4+K4up3eRh+u8wBkD0Vwno0fNlKZnd7x
5el6tuZuXecPXmNpQaXJvDRTRZzTlWQX5woscTmFYQcMV3c2C+B95ah0Lk2Nb9oM
0aEy+2jv59oDGJS6/0UhtTKNzFAUpn3FDvsYavWmQQNngVDj0RHkxSVHfGfunbZD
jp4WlhSiD+aKsIvgefzTeGzHdcC5gIOZgQW9Fiyfqvb2fPn2ocdqovmwHteSq/Xp
AkM/Luq1NQ0+mWG0TJc2X4rwTer7dMsU3lgIik+ReWry20lNSZYrWT5mom4f6737
JEWsI2fjkSB/QOSY8laAE4aocago3xK/zLUCzFLJlxsJbP8GXMzXWnA/vwcJpDiI
4wDVN2bMxs+WcSxDSaYRH2xaz4AM/WGwa+Nm4Xcd
-----END CERTIFICATE-----
desjardins.crt lines 1-32/32 (END)
```

Firefox affiche la page sans alerte. Car nous avons à la question précédente indiqué *www.desjardins.com* au champ « Common Name ».

e) Firefox bloque la page et affiche une alerte pour les deux sites. En scrutant plus en détail le certificat, nous apprenons des informations intéressantes telles que l'organisation (**Hacker Inc.**) et Organizational Unit (**Phishing Dept**). En somme, rien qui ne présage de bonnes choses.

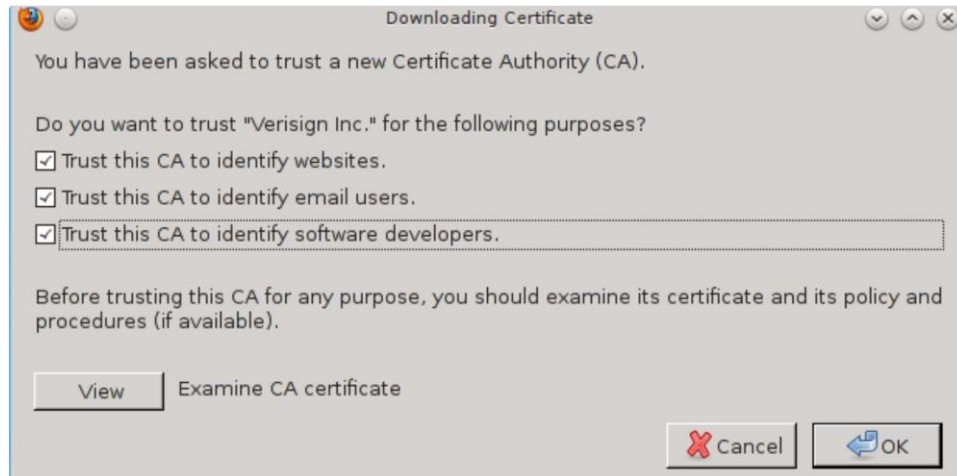


f) Nous observons qu'une exception temporaire a bien été ajoutée au sein du navigateur.



Après suppression du cache, l'exception précédemment ajoutée est supprimée et n'apparaît plus dans la liste des exceptions de la fenêtre **Certificate Manager**. En effet, l'exception est temporaire et non permanente. De fait, nous n'avons plus accès au site et Firefox affiche une alerte.

g)



L'ajout de l'exception temporaire a permis l'accès au site web. Après avoir accepté le CA du site web de rbc, nous avons maintenant un certificat permanent. Ainsi si nous vidons le cache, cela n'aura aucun d'effet, car le certificat est permanent et non temporaire.

h) Le site **bmo.com** possède également un certificat édité par la même autorité que **rbc.com**, "Verisign inc." Donc, on peut aussi y accéder, car on a par le passé fait le choix de faire confiance à cette autorité.

i) Accepter un certificat "self-signed" est dangereux car on n'a aucun de moyen de vérifier l'intégrité et la bonne foi du site. Ils sont généralement utilisés en entreprise et les employés ont pour consigne d'ignorer les avertissements. D'un point de vu "éducation des utilisateurs à une informatique" saine, une telle pratique est déplorable et entraîne de mauvaises habitudes.

Quant aux CA, ils sont dangereux car d'autres certificats ayant la même autorité peuvent ensuite être aussi automatiquement acceptés par le navigateur. Ils sont en quelques sortes sur une liste blanche à confiance aveugle. Ils augmentent la prolifération d'attaques.

### Question 3 - Chiffrement par bloc et modes d'opération [/0.5]

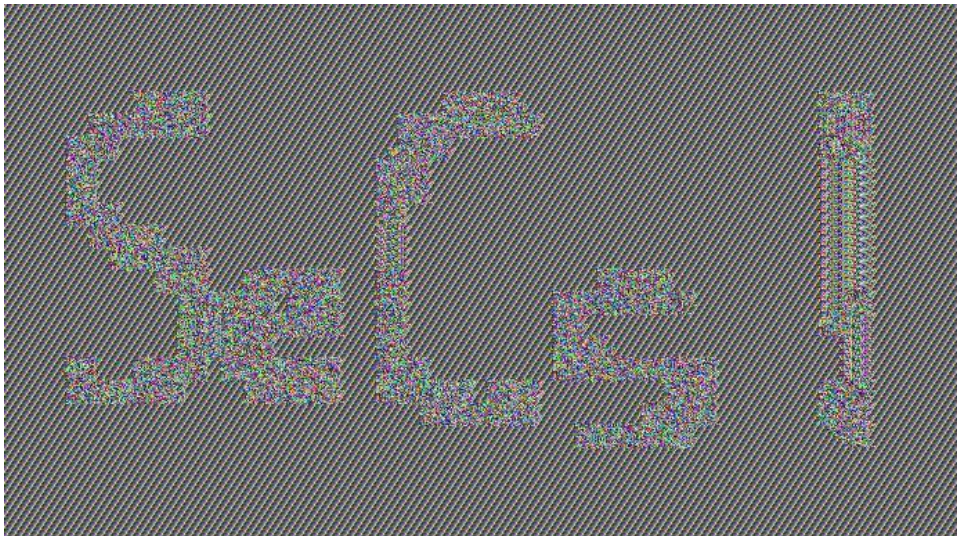
a)

```
troisO@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF4420A
TP1/Chiffrement par bloc$ ./AES.py
Erreur de syntaxe

Ce script chiffre un fichier jpeg avec AES-256 en mode ECB ou CBC
Options :
-i, --input      fichier jpeg
-m, --mode       ECB ou CBC
-o, --out        fichier chiffre (facultatif)

troisO@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF4420A
TP1/Chiffrement par bloc$ ./AES.py -i mdp.jpg -m ECB -o mdpChiffre.jpg
troisO@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF4420A
TP1/Chiffrement par bloc$ ls
AES.py  mdpChiffre.jpg  mdp.jpg
troisO@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF4420A
TP1/Chiffrement par bloc$ ./AES.py -i mdp.jpg -m CBC -o mdpChiffreCBC.jpg
```

Lancement du programme python AES.py

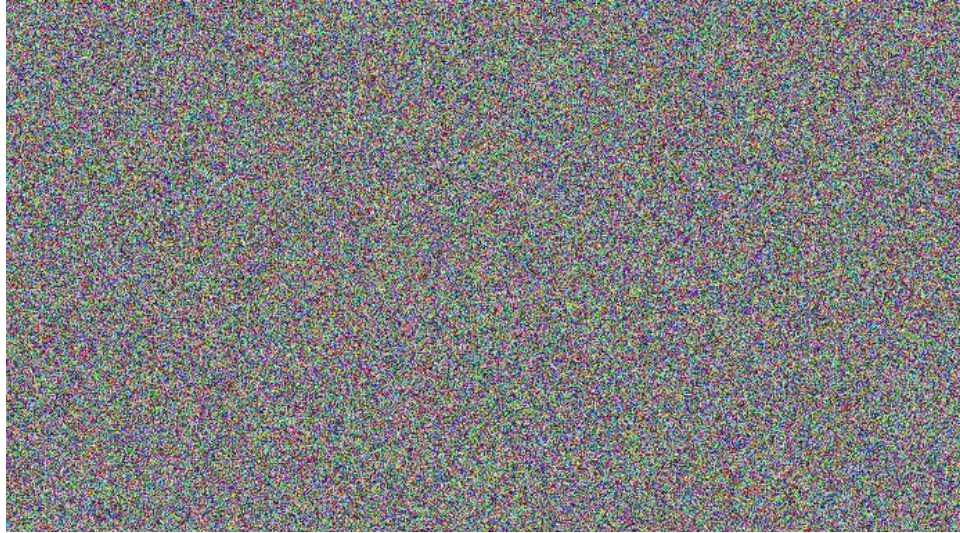


Chiffrement AES mode ECB

On distingue très nettement les formes des lettres qui composent le mot de passe ainsi que les séparations entre les blocs.

b)





Chiffrement AES mode CBC

Le fichier généré ici présente du bruit aléatoire et ne permet aucune reconnaissance visuelle de l'image originale. Le mode CBC respecte bien mieux le **principe d'incertitude et de chaos** que le mode ECB et offre donc une plus grande sécurité. Il est à privilégier.

c)

### Chiffrement AES avec mode ECB

Le principe du mode ECB repose sur la découpe des données à chiffrer en bloc de **N bits**, en effectuant ensuite un "ou exclusif" ou XOR sur chaque bloc avec une clef de chiffrement de **taille N bits** :

$$C_i = K \oplus P_i$$

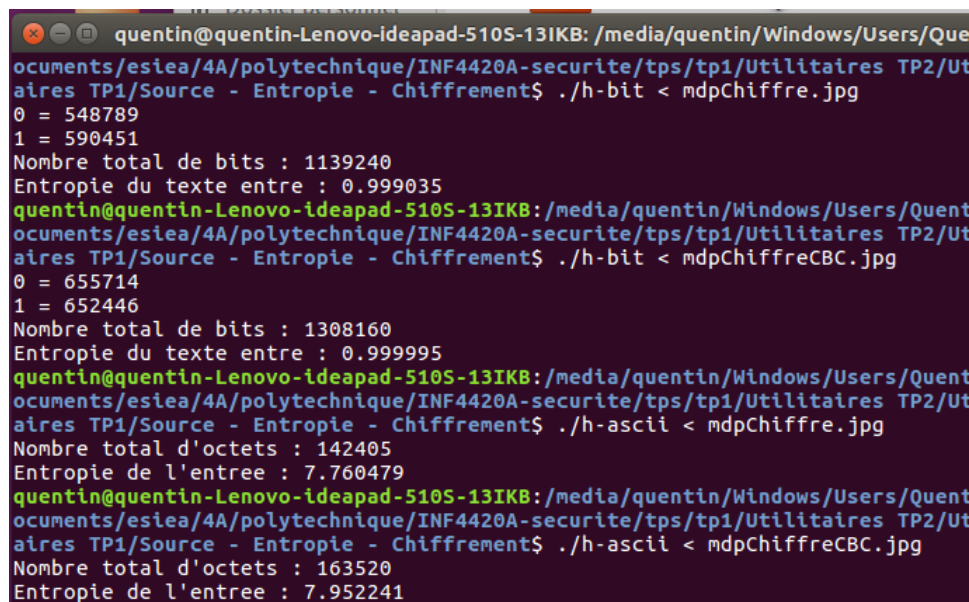
Où C représente le Chiffré (Cipher), K la clef et P le clair (Plain)

Le problème est que si plusieurs blocs contiennent les mêmes données, alors les blocs de sortie seront chiffrés de la même manière. On comprend tout de suite que l'entropie calculée sur le fichier précédemment générée serait plus **faible** par rapport au fichier généré avec le mode CBC qui, visuellement nous indique mieux respecter le principe d'incertitude. Le mode ECB présente donc deux problèmes majeurs :

- Le premier étant que deux blocs identiques c'est-à-dire avec le même contenu seront chiffrés de la même manière, cela facilitant les attaques par statistique. On appelle cette science la cryptanalyse. On peut alors, par tâtonnement, en tirer des informations précieuses, notamment **reconstituer des séquences claires à partir du**

**texte chiffré.** On obtient dès lors un « dictionnaire de codes » avec les correspondances entre le clair et le chiffré d'où le terme **codebook**.

- Le second étant que le mode ECB ne respecte pas l'intégrité des données. Un attaquant peut tout à fait remplacer un bloc chiffré par d'autres blocs chiffrés du message, permuter deux blocs, et ce sans que le destinataire ne s'aperçoive de rien. Il n'existe pas de "**chaînage**" entre les différents blocs et permet donc des attaques telles que la suivante : Imaginons une transaction bancaire d'un montant de **1999** euros. Un attaquant, adorateur de l'argent facile, permutera alors le premier et le dernier chiffre : **9991** euros.



```
quentin@quentin-Lenovo-ideapad-510S-13IKB: /media/quentin/Windows/Users/Quent
ocuments/esiea/4A/polytechnique/INF4420A-securite/tps/tp1/Utilitaires TP2/Ut
aires TP1/Source - Entropie - Chiffrement$ ./h-bit < mdpChiffre.jpg
0 = 548789
1 = 590451
Nombre total de bits : 1139240
Entropie du texte entre : 0.999035
quentin@quentin-Lenovo-ideapad-510S-13IKB: /media/quentin/Windows/Users/Quent
ocuments/esiea/4A/polytechnique/INF4420A-securite/tps/tp1/Utilitaires TP2/Ut
aires TP1/Source - Entropie - Chiffrement$ ./h-bit < mdpChiffreCBC.jpg
0 = 655714
1 = 652446
Nombre total de bits : 1308160
Entropie du texte entre : 0.999995
quentin@quentin-Lenovo-ideapad-510S-13IKB: /media/quentin/Windows/Users/Quent
ocuments/esiea/4A/polytechnique/INF4420A-securite/tps/tp1/Utilitaires TP2/Ut
aires TP1/Source - Entropie - Chiffrement$ ./h-ascii < mdpChiffre.jpg
Nombre total d'octets : 142405
Entropie de l'entree : 7.760479
quentin@quentin-Lenovo-ideapad-510S-13IKB: /media/quentin/Windows/Users/Quent
ocuments/esiea/4A/polytechnique/INF4420A-securite/tps/tp1/Utilitaires TP2/Ut
aires TP1/Source - Entropie - Chiffrement$ ./h-ascii < mdpChiffreCBC.jpg
Nombre total d'octets : 163520
Entropie de l'entree : 7.952241
```

En utilisant l'utilitaire **./h-bit** pour chacun des fichiers générés, on observe que l'image générée avec le mode CBC présente une meilleure répartition des 0 et des 1 et offre donc une meilleure entropie, comme présupposé alors. L'utilitaire **./h-ascii** confirme qu'il existe une plus grande incertitude pour l'image chiffrée avec le mode CBC.

#### Conclusion :

Bien que AES offre une sécurité reconnue, son mode lui, doit être choisi avec précaution. Et même si CBC présente également des défauts - qui ne seront pas abordés ici car cela dépasserait le cadre de ce TP - il est à préférer au mode ECB pour des raisons évidentes évoquées ci-dessus.

## Question 4 - Organisation des mots de passe en UNIX/Linux [/1]

a)

```

Mdp etc # cat passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/false
daemon:x:2:2:daemon:/sbin:/bin/false
adm:x:3:4:adm:/var/adm:/bin/false
lp:x:4:7:lp:/var/spool/lpd:/bin/false
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
news:x:9:13:news:/var/spool/news:/bin/false
uucp:x:10:14:uucp:/var/spool/uucp:/bin/false
operator:x:11:0:operator:/root:/bin/bash
portage:x:250:250:portage:/var/tmp/portage:/bin/false
nobody:x:65534:65534:nobody:/var/empty:/bin/false
man:x:13:15:added by portage for man:/usr/share/man:/sbin/nologin
sshd:x:22:22:added by portage for openssh:/var/empty:/sbin/nologin
Mdp etc # ls -l passwd
-rw-r--r-- 1 root root 648 Aug 27 2012 passwd
Mdp etc #

```

Sous Linux, toutes les données de connexion des utilisateurs sont stockées dans le fichier **/etc/passwd**. C'est un simple fichier texte utilisé lors de la connexion d'un utilisateur. Les informations utilisateur telles que le nom d'utilisateur, l'UID, le GID, le répertoire de base, shell, etc. sont stockées dans le fichier **/etc/passwd**. Ce fichier est lisible par tous les utilisateurs mais ne peut être modifié que par l'utilisateur **root**.

Les permissions d'accès du fichier **/etc/passwd** sont :

- **rw-** : indique que le propriétaire du fichier, root en l'occurrence, peut lire et modifier (et donc supprimer) le fichier. En revanche, il ne peut pas l'exécuter car il n'a pas de x à la fin.
- **r--** : tous les utilisateurs qui font partie du groupe root mais qui ne sont pas root peuvent seulement lire le fichier. Ils ne peuvent ni le modifier, ni l'exécuter.
- **r--** : tous les autres utilisateurs peuvent seulement lire le fichier

Le fichier ne contient pas de mots de passe car il est historiquement lisible par tous. Seul le fichier **/etc/shadow** contient des mots de passe et n'est accessible que par root pour des raisons de sécurité évidentes. Le champ qui contenait autrefois un mot de passe dans **/etc/passwd** contient désormais le caractère "x", comme on peut le voir sur l'image ci-dessus.

b)



```

mdp etc # cat shadow
root:$6$mq.JEjcm$7S1IKCSLXahz.Am1u6kug1Fj4V13jdg1CDNBIRIi0jrjmmv9UFQLasRjIqGvP0P7KbgLWMxi.8XycKotUUMM0:15580:0:0:0:
halt:!:9797:0:0:0:
operator:!:9797:0:0:0:
shutdown:!:9797:0:0:0:
sync:!:9797:0:0:0:
bin:!:9797:0:0:0:
daemon:!:9797:0:0:0:
adm:!:9797:0:0:0:
lp:!:9797:0:0:0:
news:!:9797:0:0:0:
uucp:!:9797:0:0:0:
nobody:!:9797:0:0:0:
man:!:15513:0:0:0:
sshd:!:15513:0:0:0:

```

Contenu du fichier `/etc/shadow` avant ajout d'un utilisateur

```

mdp etc # useradd -g users -s/bin/bash -m qucou
mdp etc # passwd qucou
New password:
Retype new password:
passwd: password updated successfully
mdp etc # cat passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/false
daemon:x:2:2:daemon:/sbin:/bin/false
adm:x:3:4:adm:/var/adm:/bin/false
lp:x:4:7:lp:/var/spool/lpd:/bin/false
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
news:x:9:13:news:/var/spool/news:/bin/false
uucp:x:10:14:uucp:/var/spool/uucp:/bin/false
operator:x:11:0:operator:/root:/bin/bash
portage:x:250:250:portage:/var/tmp/portage:/bin/false
nobody:x:65534:65534:nobody:/var/empty:/bin/false
man:x:13:15:added by portage for man:/usr/share/man:/sbin/nologin
sshd:x:22:22:added by portage for openssh:/var/empty:/sbin/nologin
quentin:x:1000:1000:/home/quentin:/bin/bash
qucou:x:1001:1000:/home/qucou:/bin/bash
mdp etc # cat shadow
root:$6$mq.JEjcm$7S1IKCSLXahz.Am1u6kug1Fj4V13jdg1CDNBIRIi0jrjmmv9UFQLasRjIqGvP0P7KbgLWMxi.8XycKotUUMM0:15580:0:0:0:
halt:!:9797:0:0:0:
operator:!:9797:0:0:0:
shutdown:!:9797:0:0:0:
sync:!:9797:0:0:0:
bin:!:9797:0:0:0:
daemon:!:9797:0:0:0:
adm:!:9797:0:0:0:
lp:!:9797:0:0:0:
news:!:9797:0:0:0:
uucp:!:9797:0:0:0:
nobody:!:9797:0:0:0:
man:!:15513:0:0:0:
sshd:!:15513:0:0:0:
quentin:!:17784:0:99999:7:::
qucou:$6$.gEbo1FR$6utg1n5XjL/TbuLF8JZbrggUPbcBy/iF0qp7iF2WmNQhbb0Zi0tX0D0RgY7I0oc5KfmsCq/D3Tu/0bf12U2io/:17784:0:99999:7:::

```

Contenu des fichiers `passwd` et `shadow` après création des utilisateurs `quentin` et `qucou`

Une ligne a été ajoutée au contenu du fichier `passwd` après création de l'utilisateur **quentin**. On note néanmoins, dans le fichier `shadow`, qu'aucun mot de passe n'a été associé à ce compte. Un autre utilisateur – **qucou** - s'est vu attribué un mot de passe comme on peut le voir très clairement dans le contenu du fichier `shadow`. Celui-ci est d'ailleurs **haché**.

c) On remarque que seul le fichier `shadow` est modifié. En effet, on observe très clairement que le mot de passe haché a changé. Ce qui semble logique puisque nous venons de changer le mot de passe manuellement. On en déduit/confirme que les mots de passe se trouvent et sont gérés par le fichier `shadow`, et non le fichier `passwd` comme son nom pourrait le laisser à penser.

```

qucou:$6$.gEbo1FR$6utg1n5XjL/TbuLF8JZbrggUPbcBy/iF0qp7iF2WmNQhbb0Zi0tX0D0RgY7I0oc5KfmsCq/D3Tu/0bf12U2io/:17784:0:99999:7:::

```

```

Mdp etc # ls -l shadow
-rw-r----- 1 root root 524 Sep 10 03:35 shadow

```

Comme l'indique l'image ci-dessus, seuls l'utilisateur root peut lire et modifier le fichier shadow. Le groupe root peut seulement lire. Tous les autres utilisateurs n'ont aucun droit sur ledit fichier. Des permissions aussi restrictives sont indispensables pour la gestion de données sensibles telles que des mots de passe.

d) Le mot de passe chiffré diffère du message chiffré d'antan. Pour comprendre pourquoi les informations du mot de passe ont changé, il faut comprendre comment ce dernier est généré. Premièrement les mots de passe ne sont pas chiffrés, mais hachés. Ces mots de passe sont stockés selon le format suivant : **\$ID\$SALT\$HASH**. L'ID indique la fonction de hachage utilisée, HASH le haché du mot de passe. Enfin le SALT ou sel permet d'ajouter une complexité entropique et est utilisée pour la génération du mot de passe.

Type de Hachage	ID	Taille du Hash
SHA-512	\$6\$	86 caractères

Comme le sel est généré aléatoirement et change donc à chaque nouvelle utilisation, le mot de passe généré diffère également. Ci-après un code python qui illustre le processus de création d'un mot de passe sous un système d'exploitation UNIX.

```

GNU nano 2.5.3                                     File: mdp.py
python -c "import random,string, crypt;
randomSalt = ''.join(random.sample(string.ascii_letters,8));
print 'Sel : ' + randomSalt;
print crypt.crypt('monMotDePasseSecret', '\$6\$%s\$' % randomSalt)"

```

Code python

```

troisO@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF4420A-securite/tps/tp1$ ./mdp.py
Sel : QYgyNXtm
$6$QYgyNXtm$WPFU3as0/v.hYwKR5yCn6K.9ek8DXqm.Do8OnzT1oWiIlaVqwOdcSV7nYgK8aCmqAF7C4J7x22xjVK1Hm.TRH0
troisO@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF4420A-securite/tps/tp1$ ./mdp.py
Sel : BSEXGQLx
$6$BSEXGQLx$hrFCZ4te4D839Ig34KtGBcrVu2xq530JFGMXX7YSI965IFHb1aeIH.XqSU7jywrnGn2/LPJvskrUQppEdqEV8/
troisO@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF4420A-securite/tps/tp1$ ./mdp.py
Sel : yFltWrvo
$6$yFltWrvo$KxC4pHu/jIowZb/Kj5hW3rWGPYghvhuFjIvdph1BiNgzY1fIfBPykEC0j3fF1PxBpCCPQWwsmBxGQE/3dP5i1

```

Génération de trois mots de passe haché différents

On remarque que pour un même mot de passe, le haché de ce dernier diffère car **le sel change** à chaque appel du programme.

e) Il est tout à fait possible de se connecter avec le compte du second utilisateur en utilisant le haché du mot de passe du premier utilisateur. Ce que nous venons de réaliser est à

**proscrire absolument** car cela rend apparent que deux utilisateurs ou plus partagent le même mot de passe. Si un des deux comptes est compromis, l'autre le sera aussi. Avec des SALT distincts, ce qui est normalement le cas, il serait impossible de dire que deux comptes possèdent le même mot de passe puisque la probabilité que deux hachés soit identique est extrêmement faible, si ce n'est quasi-nulle.

f) Les entrées dans les deux fichiers - shadow et passwd - ont été effacées. Ce qui est tout à normal suite à la suppression de l'utilisateur **quentin**. A présent, il ne reste plus que l'utilisateur **qucou**.

```
rdp etc # cat shadow
root:$6$mq.JEJcm$7S11KCSLXahz.Am1u6kug1F.j4V13.jdg1CDNBIRIi0.jr.jmno9UFQLasR.jIqGYp0P7KbgLWMxi.i.BXycKotUUMM0:15580:0:0:0:
halt:!:9797:0:0:0:
operator:!:9797:0:0:0:
shutdown:!:9797:0:0:0:
sync:!:9797:0:0:0:
bin:!:9797:0:0:0:
daemon:!:9797:0:0:0:
adm:!:9797:0:0:0:
lp:!:9797:0:0:0:
news:!:9797:0:0:0:
uucp:!:9797:0:0:0:
nobody:!:9797:0:0:0:
man:!:15513:0:0:0:
sshd:!:15513:0:0:0:
qucou:$6$5FfqYtJ0$e21kJGJksGHPnL1omSQtfuTFR9AoCTTLa8/48NxbIeohgP.Y3Ru89cK2y1s2E11L6QrXoNfALz2.2s/uuJbJYYu0:17784:0:99999:7:0:
rdp etc # cat passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/false
daemon:x:2:2:daemon:/sbin:/bin/false
adm:x:3:4:adm:/var/adm:/bin/false
lp:x:4:7:lp:/var/spool/lpd:/bin/false
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
news:x:9:13:news:/var/spool/news:/bin/false
uucp:x:10:14:uucp:/var/spool/uucp:/bin/false
operator:x:11:0:operator:/root:/bin/bash
portage:x:250:250:portage:/var/tmp/portage:/bin/false
nobody:x:65534:65534:nobody:/var/empty:/bin/false
man:x:13:15:added by portage for man:/usr/share/man:/sbin/nologin
sshd:x:22:22:added by portage for openssh:/var/empty:/sbin/nologin
qucou:x:1001:100:/home/qucou:/bin/bash
```

## Question 5 - Contrôle de qualité de choix de mot de passe [/1]

a) Lors de la première phase d'essai, 7 mots de passe ont été trouvés. On notera que le compte inf4420 possède le même haché et donc le même mot de passe dans les deux fichiers password1 et password2.

```

Mdp john # john /root/password1 /root/password2
Loaded 14 password hashes with 12 different salts (FreeBSD MD5 [32/64 X2])
0244fni      (inf4420)
0244fni      (inf4420)
john1        (john)
claudia      (david)
security     (admin)
niemtel      (lola)
Tigers5      (andre)
3sunshine    (morning)
guesses: 8   time: 0:00:10:04 (3)  c/s: 11737   trying: 47252812 - 47252815
guesses: 8   time: 0:00:12:52 (3)  c/s: 11617   trying: myrg - myb!
guesses: 8   time: 0:00:17:01 (3)  c/s: 10945   trying: dinko - dine9
guesses: 8   time: 0:00:18:13 (3)  c/s: 10916   trying: drk2fs - drk2m0
Session aborted

```

```

Mdp john # cat john.pot
$1$Wila6SGN$LPLfCWuikEzKOb7CPT01p.:0244fni
$1$N/P09Tgu$CAsOZntIFmZk3tAfrZY2B0:john1
$1$Aw/cHolc$laW8KVkQeJAernWEITL3B/:claudia
$1$arMaK13M$PMYzT2poiPR4pdGW26rlw0:security
$1$S2uBDM/D$C8dXktTJAjxUndXThMboX/:niemtel
$1$fU99GiZo$UAg3oILYbUuYsdiahaBMf1:Tigers5
$1$hLGaa7.R$FbMLS3T/XJIrSkUcWnHu.1:3sunshine

```

b) Soit un alphabet  $X$  de  $N$  lettres, l'entropie de cet alphabet, noté  $H(X)$ , est l'opposé de la somme du produit des probabilités multipliées par leur logarithme en base 2. L'entropie est maximale quand la probabilité d'apparition de chaque élément est équiprobable.

$$H(X) = - \sum_{n=1}^N p(x) \cdot \log_2 p(x) = -\log_2(p) = \log_2(N) = \log_2(52) = 5.70$$

On vérifie  $2^{5.70} = 52$

$H(x) = \log_2(62) = 5.95$  On vérifie :  $2^{5.95} = 62$

$H(x) = \log_2(256) = 8$  On vérifie :  $2^8 = 256$  pour une table ASCII étendu.

c) Un mot de passe sera d'autant plus fort qu'il y a de caractères dans l'alphabet utilisé. (en supposant que les caractères aient une fréquences d'apparition équiprobable). Plus de choix dans l'alphabet (source) entraîne une plus grande incertitude sur les caractères de sortie.

d) Les critères sont :

- Les mots de passe ne doivent pas être des mots du dictionnaire,
- Les mots de passe ne doivent pas être sémantiquement proche du login. Ex : john -> john1 ou inf4420 -> 0244fni,

- Les mots de passe ne devraient pas utiliser d'alphabet tel que le leetSpeak qui consiste à faire une correspondance des caractères de l'alphabet usuelle vers l'alphabet leetspeak. Ex : Soleil -> 50l31l. On augmente un peu l'entropie, mais un outil tel que JohnTheRipper bien configuré viendra à bout de ce genre de mot de passe,
- Les longs mots de passe sont à privilégier comme les passphrase,
- Ne jamais utiliser son nom, sa date de naissance ou tout ce qui peut être relié directement ou indirectement à soi et qui est donc par conséquent facilement devinable,
- Le mot de passe **Tr0ub4dor&3** possède 28 bits d'entropie, donc  $2^{28}$  combinaisons. Tandis que le mot de passe : **correct horse battery staple** possède 44 bits d'entropie, donc  $2^{44}$  combinaisons. Le premier est de prime abord sophistiqué et pourtant il n'en n'est rien, ce mot de passe est facilement crackable comparé au second.

## Partie C

### Question 1 - Échec du protocole RSA [/0.75]

a) Il s'agit d'appliquer une attaque sur RSA communément appelée **textbook RSA ou raw RSA** où le chiffrement est directement appliqué au message. Cela rend le chiffrement déterministe et permet à un attaquant - étant donné un texte chiffré - de rechercher le texte en clair correspondant et d'établir une table des correspondances. Généralement, afin de chiffrer proprement un message en utilisant RSA, il faut au préalable lui appliquer un schéma de remplissage (**OAEP/PKCS**) qui permet d'ajouter un caractère aléatoire au schéma déterministe qu'est intrinsèquement RSA. Sans cela, RSA n'est pas sécurisé et permet de nombreuses attaques comme celle décrite plus en détail à la question suivante.

b) **Matricule = 1973362** ;  $e = 449$  ;  $n = 103009$  ; Texte Chiffré = {61038, 100123, 21665, 12728, 87116}

A l'aide d'un simple script python, on génère pour chaque lettre son chiffré correspondant. On peut ainsi établir une table de correspondance en utilisant un dictionnaire python.

```
trois0@qucou: /mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF4420A-securite/tps/tp1$ python3 rsaTextbookAttack.py
A = 0 0
B = 1 1
C = 2 46740
D = 3 86345
E = 4 12728
F = 5 45056
G = 6 78698
H = 7 10962
I = 8 29745
J = 9 79641
K = 10 1444
L = 11 89708
M = 12 99148
N = 13 32495
O = 14 100123
P = 15 19417
Q = 16 71836
R = 17 61038
S = 18 87116
T = 19 35174
U = 20 21665
V = 21 67198
W = 22 73584
X = 23 38378
Y = 24 8628
Z = 25 44773
{0: 'A', 1: 'B', 45056: 'F', 86345: 'D', 99148: 'M', 87116: 'S', 10962: 'H', 46740: 'C', 79641: 'J', 19417: 'P', 100123: 'O',
71836: 'Q', 21665: 'U', 1444: 'K', 44773: 'Z', 35174: 'T', 78698: 'G', 89708: 'L', 61038: 'R', 32495: 'N', 73584: 'W', 297
45: 'I', 8628: 'Y', 12728: 'E', 38378: 'X', 67198: 'V'}
Texte clair = ROUES
```

Le texte déchiffré est : “**ROUES**”.

c) La fonction de chiffrement code les lettres A et B pour 0 et 1, car les valeurs non chiffrées de A et B sont tout simplement 0 et 1. Or la fonction de chiffrement est du type  $(i ** e) \% n$ .

C’est pour cela que généralement, afin de chiffrer proprement un message en utilisant RSA, il faut au préalable lui appliquer un schéma de remplissage (**OAEP/PKCS**) qui permet d’ajouter un caractère aléatoire au schéma déterministe qu’est intrinsèquement RSA. Sans cela, RSA n’est pas sécurisé et permet de nombreuses attaques comme celle décrite plus en détail à la question suivante.

## Question 2 - Déchiffrement "simple" [/0.75]

Le chiffrement utilisé ici est un chiffrement de type substitution **mono-alphabétique** utilisant un **alphabet désordonné**, ce qui rend le déchiffrement plus complexe. En effet, l’alphabet comptabilisant 26 caractères, nous avons **26!** combinaisons. Cette clé a donc une entropie de  $\log_2(26!) = \log_2(4.03 * 10^{26}) = \log_2(4) + \log_2(10^{26}) = 2 + 26 * \log_2(10) = 2 + 78 = 80 \text{ bits}$ . Le résultat exact est **88 bits**. Ce qui est une taille de clé relativement grande. Le caractère @ (espace) est volontairement omis.

### Algorithme de substitution utilisé

- **Source**
  - Texte en caractères latin
- **Codage**
  - aucun

- **Chiffrement**
  - $x \rightarrow \pi(x)$
- **Clé**
  - $\pi$  (une table de substitution)

L'énoncé nous dit que le texte clair est en anglais, mais pour nous en assurer nous avons calculé l'indice de coïncidence du texte chiffré.

```
trois0@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF4420A-securite/tps/tp1$ ./computeIC.py cipherQ2.txt
0.069 (0.0685144124169)
```

L'indice de coïncidence correspond à l'IC moyen de la langue anglaise (0.0667). On confirme donc que le texte est en anglais et nous pouvons commencer notre cryptanalyse.

- L'analyse fréquentielle des lettres.
- L'analyse fréquentielle des digrammes.
- L'analyse fréquentielle des trigrammes.

Lorsqu'un texte est relativement court comme celui-ci, il est risqué de se concentrer uniquement sur les mots de grande taille et/ou sur les fréquences individuelles des lettres car la correspondance avec les lettres claires et les lettres du texte chiffré peut être extrêmement hasardeuse. Nous nous sommes donc concentrés sur de petits mots - les mots de deux ou trois lettres - car plus facile à cryptanalyser. L'entropie d'un digramme ou trigramme est effectivement plus basse que l'entropie d'une lettre seule car nous effectuons le calcul suivant :  $H(S)/n$ -gramme. On en déduit que plus  $n$  est grand, plus on se rapproche de l'entropie du langage. Enfin, une fois ces mots déchiffrés, nous pouvons plus facilement attaquer les mots de plus grande taille. Le reste est trivial et machinale.

Un rapide coup d'œil nous indique par exemple que plusieurs occurrences du mot **CAE** apparaissent ce qui nous fait immédiatement penser au trigramme **THE**. On trouve donc 3 correspondances de lettre en un seul coup d'œil, ce qui est un bon début et facilitera grandement notre travail par la suite.

L'alphabet utilisé est le suivant :

HYTFEZ\_\_LNMKSCUIGRAXOPWVBD

ABCDEFGHIJKLMNOPQRSTUVWXYZ

En découle le texte déchiffré :

***He has constrained our fellow citizens taken captive on high seas to bear arms against their country, to become the executioners of their friends and brethren, or to fall themselves by their hands***

Matricule = 1973362

## **Sources :**

Q1 :

- [https://fr.wikipedia.org/wiki/Entropie\\_de\\_Shannon](https://fr.wikipedia.org/wiki/Entropie_de_Shannon)
- <http://benhur.telug.ca/SPIP/inf6460/spip.php?article110>
- <http://nomis80.org/cryptographie/node23.html>

Q2 :

- <http://math.pc.vh.free.fr/divers/crypto/cryptanalyse.htm>
- [http://www.bibmath.net/crypto/index.php?action=affiche&quoi=chasseur/frequences\\_english](http://www.bibmath.net/crypto/index.php?action=affiche&quoi=chasseur/frequences_english)

Q3 :

- [https://fr.wikipedia.org/wiki/Sécurité\\_inconditionnelle](https://fr.wikipedia.org/wiki/Sécurité_inconditionnelle)

<https://www.shellhacks.com/linux-generate-password-hash/>

<https://unix.stackexchange.com/questions/209231/does-the-shadow-file-have-encrypted-passwords>

<https://unix.stackexchange.com/questions/219933/same-salt-hash-value-in-etc-shadow>

<https://www.tutorialsandyou.com/linux/linux-passwd-file-5.html>

<https://crypto.stackexchange.com/questions/41170/what-advantage-is-there-for-using-a-nonce-and-a-timestamp>