



INF4420A - Sécurité informatique

Automne 2021

TP1 - Cryptographie

Groupe 05

Par



Équipe 13

Soumis à

95/100

Excellent travail, cependant pour les questions du NIP j'ai remarqué que certaines réponses sont identiques à celles de Pier-Luc et Ming !! Comme ils ont mis leurs matricules sur les captures et pas vous j'en déduis que vous les avez récupéré... Je laisse passer car la question en particulier n'était pas très claire, cependant je voudrais que ça ne se reproduise pas.

6 octobre 2021

Partie A

Question 1 - Entropie (0.75) 0.75/0.75

a) Calculez l'entropie par lettre (h-lettre) d'une chaîne générée avec texte d'une longueur de 200 caractères. ok

The terminal window shows the following command and its output:

```
(inf4420a@inf4420a)[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./texte 200 > texte.bin

(inf4420a@inf4420a)[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-lettre < texte.bin
(space) = 39
A = 12
B = 4
C = 3
D = 6
E = 18
F = 3
G = 2
H = 13
I = 11
J = 0
K = 2
L = 8
M = 6
N = 5
O = 11
P = 3
Q = 2
R = 8
S = 14
T = 18
U = 7
V = 2
W = 2
X = 0
Y = 1
Z = 0
Nombre total de caracteres : 200
Entropie de l'entree : 4.047211

(inf4420a@inf4420a)[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ echo 1956576 2021-09-07
1956576 2021-09-07
```

Figure 1: Calcul de l'entropie par lettre d'une chaîne générée avec texte de 200 caractères

L'entropie d'une chaîne de 200 caractères générée avec texte est de 4.047211.

b) En vous servant du premier théorème de Shannon, expliquez ce que signifie cette valeur.

Elle indique que pour cette source, chaque symbole émis peut être codé individuellement avec en moyenne 4.047221 bits, soit la valeur de l'entropie. Cette valeur montre donc à quel point la source (dans ce cas-ci, texte) émet des valeurs avec une grande incertitude. Si les valeurs retournées varient beaucoup, l'entropie sera grande. Plus l'entropie est grande, plus il faudra d'informations pour décoder ce que la source a transmis sans aucune ambiguïté. ok

Il s'agit donc d'un indicateur de la quantité d'information minimale contenue dans la source sans toutefois perdre d'information. Par exemple, si nous avons une entropie proche de 1, cela signifie que

pour chaque caractère envoyé, la source devra fournir 1 bit d'information pour que le récepteur puisse décoder celui-ci sans ambiguïté. **oui**

c) Quelle serait l'entropie par lettre (en moyenne) d'un fichier qui aurait été généré de la même façon, mais avec les mêmes probabilités (1/27) pour chacun des 27 symboles (lettres majuscules et espace)?

Comme toutes les lettres auraient autant de chance d'être envoyées, nous nous retrouverons avec une entropie maximale, soit ici de $\log_2(27) = 4.75$. Cela indique donc que nous avons besoin entre 4 et 5 bits pour indiquer au récepteur sans ambiguïté quelle lettre a été envoyée. **ok**

d) Que représente le quotient de la valeur en a) sur la valeur en c) ?

$4.047211 / 4.75 = 0.852$. Ce ratio démontre à quel point on est proche ou non de l'entropie maximale (et donc que les lettres apparaissent de façon plutôt équitable et complètement aléatoirement). Il s'agit de ce qu'on appelle le taux de compression. **exact**

Dans notre cas, avec le programme texte, bien qu'on voit que les lettres n'apparaissent pas complètement aléatoirement (car nous n'avons pas un ratio de 1), on est loin de pouvoir prédire sans ambiguïté la prochaine lettre. Donc, avec le générateur de lettre texte, nous nous trouvons plus près d'une situation aléatoire (quoique pas complètement) que d'une situation prévisible.

e) Refaites la même chose qu'en a) avec la source *lettre*. Comparez la valeur obtenue avec celle en a). Est-ce que la différence est significative (supérieure à 0.4) ?

The terminal window displays the following commands and output:

```
(inf4420a@inf4420a)[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./lettre 200 > lettre.bin
(inf4420a@inf4420a)[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-lettre < lettre.bin
(space) = 24
A = 12
B = 0
C = 4
D = 6
E = 25
F = 3
G = 3
H = 10
I = 15
J = 0
K = 3
L = 6
M = 4
N = 13
O = 11
P = 3
Q = 0
R = 13
S = 13
T = 15
U = 5
V = 3
W = 6
X = 0
Y = 3
Z = 0
Nombre total de caractères : 200
Entropie de l'entrée : 4.120755

(inf4420a@inf4420a)[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ echo 1956576 2021-09-07
1956576 2021-09-07
```

In the background, a Moodle interface shows a course titled "INF4420A" and a sub-page titled "Utilitaire TP1".

Figure 2: L'entropie pour une chaîne générée avec *lettre* contenant 200 caractères

L'entropie pour une chaîne de 200 caractères générée avec *lettre* est de 4.120755. La différence par rapport à la valeur obtenue en a) (4.047211) n'est pas significative (0.073544) étant donné que l'écart est inférieur à 0.4. **Oui**

Toutefois, cela indique tout de même qu'il y a une tendance à ce que l'entropie soit plus grande pour *lettre* et donc que les lettres soient générées plus aléatoirement avec *lettre* qu'avec *texte* (il faut donc légèrement plus de bits pour que le récepteur puisse déterminer sans ambiguïté la lettre, on s'approche un peu de l'entropie maximale et donc d'une meilleure sécurité). C'est normal, étant donné que l'ordre des lettres dans *texte* est plus logique (puisque c'est tiré d'un texte en anglais et donc que l'ordre correspond aux normes linguistiques) et que l'ordre des lettres dans *lettre* ne suit pas de norme.

f) On sait qu'un texte anglais est constitué de mots et de phrases qu'il est nécessaire d'interpréter en fonction d'un langage et d'une grammaire. Un texte anglais est donc très redondant (et donc facile à compresser). Les chaînes générées par *lettre* ne sont pas de l'anglais malgré l'utilisation des mêmes fréquences. Le résultat obtenu en e) peut donc surprendre. Expliquez cette contradiction apparente (le fait que les deux entropies soient proches).

Même si pour *lettre* les lettres ne sont pas tirées d'un texte de la langue anglaise comme pour *texte*, les fréquences utilisées pour *lettre* proviennent tout de même de la langue anglaise. Donc, comme dans les 2 cas (*texte* et *lettre*) les fréquences des lettres se ressemblent, il est normal que l'entropie se ressemble aussi, malgré le fait qu'avec *lettre*, celles-ci n'apparaissent pas dans un ordre conforme aux normes linguistiques de l'anglais. **oui !**

Question 2 - Histogrammes (0.75) 0.75/0.75

a) Utilisez les programmes *cesar* et *cesar-d* avec les sources *texte* et *lettre*, pour chiffrer et déchiffrer des chaînes de 200 caractères. **ok**

Chiffrement et déchiffrement avec *texte* :

```

(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./texte 200 > texteStart.bin
(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./cesar < texteStart.bin > texteCesar.bin
(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./cesar-d < texteCesar.bin > texteCesarD.bin
(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ cat texteStart.bin
LEAU OF THE LYONS CASE HE WEARES LEAST MEETING WITH THE LYON IN THE FEELD HE CHAUNCE TO TEARE HIM PEECEMEALE FOR HIS PR
IDE ART THE SOUNDEST COUNSELL I CAN GIUE HIS GRACE IS TO SURRENDER ERE HE BE CON
(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ cat texteCesar.bin
Télécharger le dossier
OHDXH RI WKH OBRQV FDVH KH ZHDUHV OHDVW PHH WLQJ ZLWK WKH OBRQ LQ WKH IHHOG KH FKDXQFH WR WHDUH KLP SHHFPHDOH IRU KLV SU
LGH DUW WKH VRXQGHW FRXQVHOO L FDQ JLXH KLV JUDFH LV WR VXUUHQGHU UH KH EH FRQ
(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ cat texteCesarD.bin
Énoncé TP1
LEAU OF THE LYONS CASE HE WEARES LEAST MEETING WITH THE LYON IN THE FEELD HE CHAUNCE TO TEARE HIM PEECEMEALE FOR HIS PR
IDE ART THE SOUNDEST COUNSELL I CAN GIUE HIS GRACE IS TO SURRENDER ERE HE BE CON
(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ echo 1956576 2021-09-08
1956576 2021-09-08

```

Figure 3: Chiffrement et déchiffrement avec *texte*

```

(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./lettre 200 > lettreStart.bin
Tableau de bord / Mes documents
(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./cesar < lettreStart.bin > lettreCesar.bin
binaire
(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./cesar-d < lettreCesar.bin > lettreCesarD.bin
Documents
(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ cat lettreStart.bin
Frequency
h-ascii h-bit
EI OIOOINUY LNEMNNDDEMEEE TOUSWR ETICEEIRFGSARI IEIBVTAB RSOOBEE USHBSEEGRUAANDNCERNIMALASARAANRTP P SHC TVD EVET EE SIMHEDHN EC
BTTOEM G AUOITIA RTROE AJUHCI OIUSNT ENTEHNWP SRA EWGYIB CSWNE
(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ cat lettreCesar.bin
Télécharger le dossier
File System
HL RLRLQXB OOHPQQGGHGPHHH WRXVZU HWLFHHLUI3VDUL LHLEYWDE UVRRREHH XVKEVHHJUXDDQGFQHUQLPDODVDUDDQWS S VKF WYG HYHW HH VLPKHWGKQ HF
EWWRHP J DXRLWLD UWURH DMXKFL RLXVQW HQWHQZS VUD HZBBLF FVZQH
(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ cat lettreCesarD.bin
Frequency
h-ascii h-bit
EI OIOOINUY LNEMNNDDEMEEE TOUSWR ETICEEIRFGSARI IEIBVTAB RSOOBEE USHBSEEGRUAANDNCERNIMALASARAANRTP P SHC TVD EVET EE SIMHEDHN EC
BTTOEM G AUOITIA RTROE AJUHCI OIUSNT ENTEHNWP SRA EWGYIB CSWNE
(inf4420a@inf4420a:[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ echo 1956576 2021-09-08
1956576 2021-09-08

```

Figure 4: Chiffrement et déchiffrement avec *lettre*

b) Utilisez le programme h-lettre pour obtenir les fréquences des lettres. Construisez des histogrammes de fréquences ordonnées du plus grand au plus petit pour la sortie de chacune des sources ainsi que pour les versions codées. (Note : Vous pouvez facilement générer ces histogrammes en redirigeant la sortie de h-lettre dans un fichier que vous pouvez importer et traiter dans Excel, par exemple). ok

Entropie par lettre pour texte :

```
└─(inf4420a㉿inf4420a)-[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-lettre < texteStart.bin > texte-h-lettre.bin
└─ Enoncé TP1
└─ Network

└─(inf4420a㉿inf4420a)-[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-lettre < texteCesar.bin > texteCesar-h-lettre.bin

└─(inf4420a㉿inf4420a)-[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ echo 1956576 2021-09-08
1956576 2021-09-08
```

Figure 5: Utilisation du programme *h-lettre* afin d'obtenir les fréquences des lettres pour *texte*

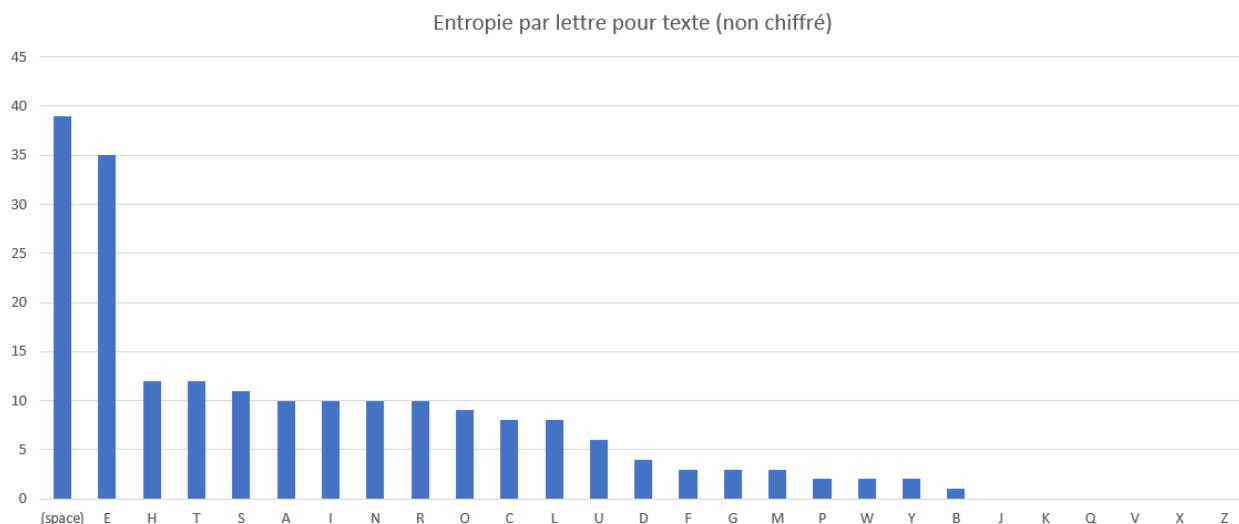


Figure 6: Fréquences des lettres pour un fichier *texte* non chiffré

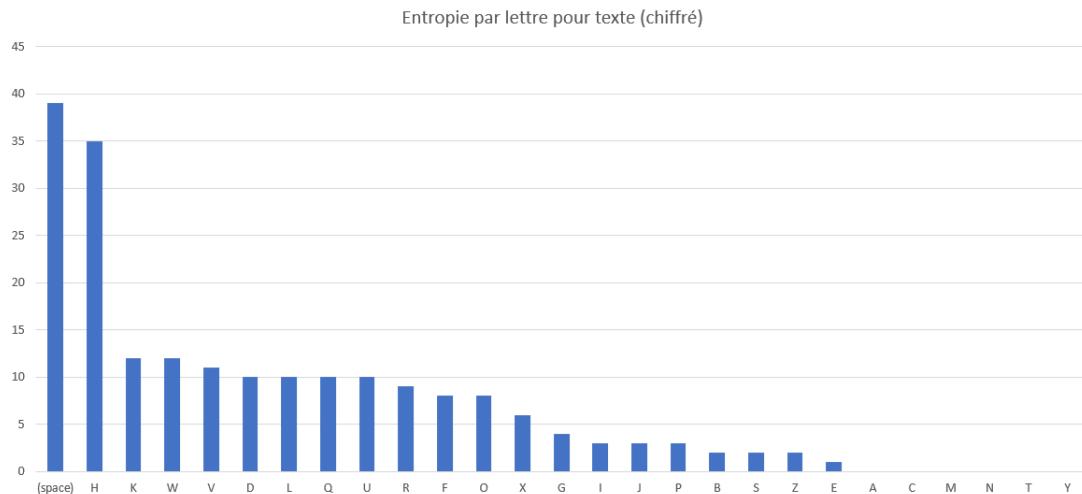


Figure 7: Fréquence des lettres d'un fichier *texte* chiffré

```

└─(inf4420a㉿inf4420a)-[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
  $ ./h-lettre < lettreStart.bin > lettre-h-lettre.bin
                                              ↳ Enoncé TP1
└─(inf4420a㉿inf4420a)-[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
  $ ./h-lettre < lettreCesar.bin > lettreCesar-h-lettre.bin
                                              ↳ Network

└─(inf4420a㉿inf4420a)-[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
  $ echo 1956576 2021-09-08
1956576 2021-09-08

```

Figure 8: Utilisation de *h-lettre* pour obtenir la fréquences des lettres d'une *lettre*

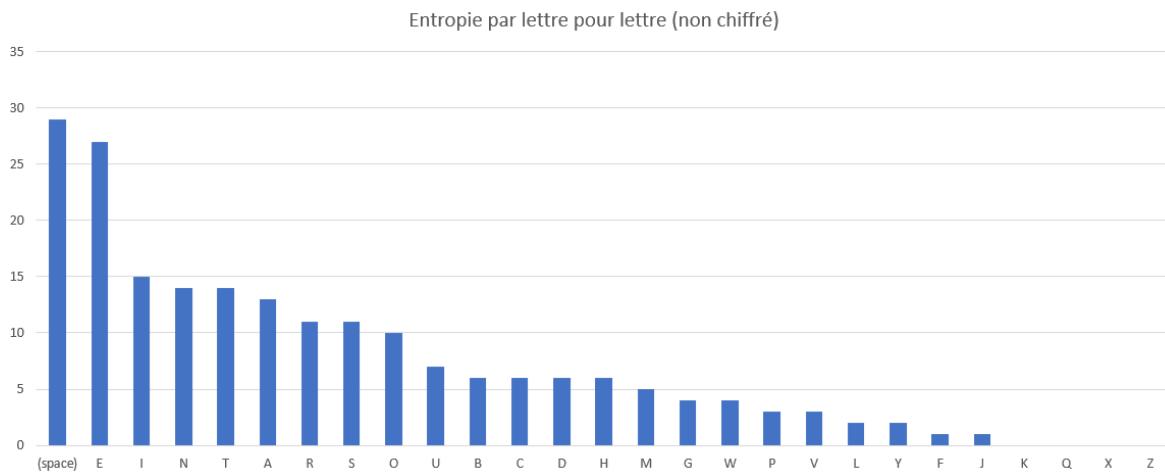


Figure 9: Fréquence des lettres pour un fichier *lettre* non chiffré



Figure 10: Fréquence des lettres pour un fichier *lettre* chiffré

c) Que remarquez-vous en comparant ces quatre histogrammes? Comment seraient les histogrammes des sources *lettre* et *texte* si les fréquences étaient comptabilisées sur deux lettres à la fois? Comment devrait être par exemple les fréquences du (ee) et du (th) dans le cas de *texte* et de *lettre*.

On remarque que les bandes (fréquences) sont identiques pour la version déchiffrée et chiffrée pour *texte* et pour *lettre* (pour les 2 indépendamment), mais la lettre associée à la fréquence change lorsqu'on chiffre. On a un décalage de 3 lettres. **ok**

On voit aussi que certaines lettres sont plus fréquentes que d'autres (comme l'espace et le E), et qu'entre *lettre* et *texte*, ça se ressemble, étant donné que *texte* est basé sur la langue anglaise et que bien que *lettre* ne le soit pas, le choix des lettres se base sur la fréquence d'apparition des lettres dans la langue anglaise. Donc, il est logique que les fréquences pour chaque lettre se ressemblent. **ok**

Maintenant, si on calculait la fréquence sur 2 lettres à la fois, bien évidemment, les fréquences pour chaque "paire" de lettres seraient plus petites que les fréquences observées présentement pour les lettres individuelles, car il est plus rare de voir 2 lettres une à la suite de l'autre que de voir la lettre tout simplement.

De plus, étant donné que *lettre* ne suit pas les normes de la langue anglaise quant à l'ordre des lettres, si on prend par exemple des paires de lettres fréquentes en anglais (comme ee ou th), la fréquence de ces paires pour *lettre* serait encore plus basse que pour *texte*, qui lui suit les normes de la langue anglaise.

oui

d) En vous référant au point précédent ainsi qu'à la question 1 f), est-ce que cette méthode (comptabiliser les fréquences sur deux lettres) facilite le déchiffrement du message dans le cas de la source texte ? Et dans le cas de lettre ? Expliquez la différence s'il y en a une. Pour chacune des deux sources, si cette méthode n'augmente pas la facilité de déchiffrement du message, quelle solution proposez-vous ?

Dans le cas de *texte*, oui la comptabilisation de la fréquence sur 2 lettres faciliterait le déchiffrement du message, étant donné que certaines paires de lettres se trouvent parfois plus fréquemment dans la langue anglaise (source non markovienne), comme ee ou th. Donc, on aurait un gain de compression et il serait plus facilement possible de le déchiffrer, surtout si on a de longs textes, où la précision des fréquences est meilleure. **oui !**

Pour ce qui est de *lettre*, non, le fait de calculer la fréquence sur 2 lettres ne faciliterait pas le déchiffrement du message, étant donné que comme mentionné, la fréquence de paires de lettres serait beaucoup plus basse que pour *texte*, puisqu'il n'y a pas de corrélation entre les symboles (source markovienne), donc on aurait moins de chances de retrouver les lettres ensemble. On a donc aucun gain de compression dans ce cas-ci. Une solution pour améliorer le déchiffrement serait de prendre un plus grand échantillon afin d'avoir plus de précision sur la lettre dont il pourrait s'agir grâce à de meilleures probabilités. **tout à fait**

Question 3 - Masque jetable (0.75) **0.75/0.75**

a) Générez un fichier de 1024 octets avec monnaie et un avec binaire. Calculer l'entropie par bit (h-bit) et l'entropie par octet (h-ascii) sur les deux fichiers créés. **ok**

Pour monnaie :

```
(inf4420a@inf4420a)[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./monnaie 1024 > monnaie.bin

(inf4420a@inf4420a)[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-bit < monnaie.bin
0 = 4079
1 = 4113
Nombre total de bits : 8192
Entropie du texte entre : 0.999988

(inf4420a@inf4420a)[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-ascii < monnaie.bin
Nombre total d'octets : 1024
Entropie de l'entrée : 7.814871

(inf4420a@inf4420a)[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ echo 1956576 2021-09-08
1956576 2021-09-08
```

Figure 11: Entropie h-bit et h-ascii pour *monnaie*

Entropie par bit pour *monnaie* : 0.999988

Entropie par octet pour *monnaie* : 7.814871

On observe que l'entropie par octet est beaucoup plus élevée que celle par bit. Il faut presque 8 bits pour déterminer la valeur d'un octet alors qu'il en faut 1 pour deviner un bit, ce qui est très logique. Nous sommes très proches de l'entropie maximale pour chacun d'entre eux (8 et 1).

```
└─(inf4420a㉿inf4420a)─[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./binaire 1024 > binaire.bin

└─(inf4420a㉿inf4420a)─[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-bit < binaire.bin
0 = 5080
1 = 3112
Nombre total de bits : 8192
Entropie du texte entre : 0.957959

└─(inf4420a㉿inf4420a)─[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-ascii < binaire.bin
Nombre total d'octets : 1024
Entropie de l'entrée : 0.826392

└─(inf4420a㉿inf4420a)─[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ echo 1956576 2021-09-08
1956576 2021-09-08
```

Figure 12: Entropie h-bit et h-ascii pour *binaire*

Entropie par bit pour *binaire* : 0.957959

Entropie par octet pour *binaire* : 0.826392

Ici, les entropies sont moins élevées. On peut donc en déduire que l'algorithme *binaire* qui génère les 0 et les 1 est moins aléatoire que pour *monnaie*, et donc qu'il est plus facile de prédire le prochain chiffre et cela fait en sorte que l'entropie diminue, puisque moins de bits sont nécessaires pour connaître la valeur sans ambiguïté. **oui**

b) Générez une clé de 1024 octets pour un masque jetable avec monnaie (tel que vu en classe, la taille de la clé doit être la même que la taille du message à chiffrer). Appliquez le masque jetable sur les deux fichiers générés au point précédent en utilisant la clé nouvellement créée. Calculer l'entropie par bit (h-bit) et l'entropie par octet (h-ascii) des nouveaux fichiers chiffrés. Qu'observez-vous? Quelles conclusions pouvez-vous en tirer?

```

└─(inf4420a㉿inf4420a)─[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./monnaie 1024 > cle.bin

└─(inf4420a㉿inf4420a)─[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./masque cle.bin 1024 monnaie.bin monnaieChiffre.bin

└─(inf4420a㉿inf4420a)─[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./masque cle.bin 1024 binaire.bin binaireChiffre.bin

└─(inf4420a㉿inf4420a)─[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-bit < monnaieChiffre.bin
0 = 4093
1 = 4099
Nombre total de bits : 8192
Entropie du texte entre : 1.000000

└─(inf4420a㉿inf4420a)─[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-ascii < monnaieChiffre.bin
Nombre total d'octets : 1024
Entropie de l'entrée : 7.825546

└─(inf4420a㉿inf4420a)─[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-bit < binaireChiffre.bin
0 = 4152
1 = 4040
Nombre total de bits : 8192
Entropie du texte entre : 0.999865

└─(inf4420a㉿inf4420a)─[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-ascii < binaireChiffre.bin
Nombre total d'octets : 1024
Entropie de l'entrée : 7.804915

└─(inf4420a㉿inf4420a)─[~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ echo 1956576 2021-09-08
1956576 2021-09-08

```

Figure 13: Calcul de l'entropie par bit (h-bit) et de l'entropie par octet (h-ascii) des nouveaux fichiers chiffrés.

Pour ce qui est de *monnaie*, on voit que l'entropie est similaire à avant le chiffrement, que ce soit par bit ou par octet. Elle n'a que très légèrement augmenté. C'est normal, car nous étions près de l'entropie maximale déjà (1 et 8). **exactement**

Par contre, pour *binaire*, l'entropie a augmenté par bit, mais surtout par octet. Donc, nous pouvons en conclure que la méthode de chiffrement a rendu la tâche de deviner le prochain bit ou octet plus difficile. Nous sommes maintenant en présence d'entropies similaires que pour *monnaie*, qui se rapprochent donc eux aussi de l'entropie maximale. Le chiffrement augmente donc la sécurité du message, puisqu'il est plus difficile de deviner son contenu. **oui !**

c) Pour les deux cas, s'agit-il d'une méthode sécuritaire de chiffrement?

Comme dans les 2 cas après l'application du chiffrement nous nous trouvons près d'une entropie maximale, on peut penser qu'il s'agit d'une méthode de chiffrement sécuritaire, malgré le fait que l'algorithme est simple et vieux. Donc, utiliser ce masque est très pertinent particulièrement lorsqu'on génère une chaîne avec *binaire*, qui avait au départ une entropie très faible, surtout par octet.

Cependant, il faut s'assurer que la clé (le masque) soit choisie correctement et que celle-ci soit uniquement accessible à la source et au récepteur, et pas à un potentiel malfaiteur.

oui !

Question 4 - Analyse de risque (1.5) 1.5/1.5

La compagnie PokerMaxProUltime, une nouvelle compagnie de poker sur Internet, vous offre un emploi de rêve en tant qu'analyste principal de sécurité pour leur nouvelle installation dans un paradis fiscal aux Caraïbes. Vous devez toutefois faire face à plusieurs défis.

a) Pour commencer, votre patron vous indique que deux sites potentiels sont retenus pour la nouvelle installation. Le site A se situe sur une île paisible, mais où le marché immobilier est gonflé par les étrangers. Il en coûterait donc 500 000 \$ pour s'installer à cet endroit. L'île B, pour attirer des capitaux étrangers, a fait une proposition à votre compagnie. Il en coûterait seulement 100 000 \$ pour s'installer sur le site B. Toutefois, selon les données météos que vous avez à votre disposition, l'île B est balayée chaque année par un ouragan qui a 25% de chance de détruire votre installation. Quelle serait votre recommandation et pourquoi ?

Il serait mieux d'aller avec l'île A, car bien que les installations coûtent moins cher sur l'île B, la probabilité est trop grande et l'impact ne peut pas être négligé, donc cela fait en sorte que le risque est grand et qu'il vaut mieux prendre l'autre option pour assurer la sûreté de fonctionnement.

En effet, l'espérance de la perte $E = P * I = 0,25 * 100\,000 \$ = 25\,000 \$$. Donc, comme nous avons une différence de coût de $500\,000 \$ - 100\,000 \$ = 400\,000 \$$, cela prendrait $400\,000 \$ / 25\,000 \$/\text{an} = 16 \text{ ans}$ avant que les coûts de la réparation des dommages causés par les ouragans atteignent les coûts supplémentaires engendrés par l'acquisition de l'île A par rapport à l'île B. Si nous supposons que l'entreprise a prévu de s'installer sur l'île pour plus de 16 ans, elle devrait choisir l'île A. **bon raisonnement**

b) Vous rencontrez les gestionnaires des diverses lignes d'affaires, et vous évaluez leurs processus d'affaire pour identifier les risques. Trois risques majeurs en ressortent : **ok**

Pour chacun de ces scénarios, précisez s'il s'agit principalement d'un scénario touchant l'intégrité, la confidentialité ou la disponibilité.

i. Un malfaiteur ignore les lignes de conduite prescrites (Terms of Service) et utilise les fonctions du logiciel pour tricher, diminuant l'intérêt du site pour les joueurs légitimes.

Intégrité, car on modifie les données.

ii. Un malfaiteur inonde le serveur de requête pour empêcher les autres joueurs de se connecter à votre site.

Disponibilité, car le site sera indisponible à cause de cette inondation.

iii. Un malfaiteur infiltre votre base de données pour obtenir certaines des informations que vous stockez sur vos clients (adresses courriel et postal, numéro de carte de crédit, habitudes de jeu, historique des achats).

Confidentialité, car les données des utilisateurs ne seront plus confidentielles.

c) Après de longs mois d'études, vous avez identifié trois agents de menace potentiels pour votre entreprise : ok

- *Tricheurs professionnels : gens qui s'y connaissent peu en informatique, mais beaucoup au jeu;*
- *Crime organisé : groupes criminalisés qui ont plusieurs experts à leur solde et qui possèdent une solide infrastructure avec des milliers d'ordinateurs compromis;*
- *Sites de poker concurrents : le jeu en ligne est un milieu lucratif et certains de vos concurrents sont prêts à tout pour connaître le secret de votre succès.*

Votre patron vous fournit le résultat de l'étude de risque qu'il a fait faire par un grand cabinet de conseil et vous demande de le compléter :

Tableau 1: Résultats de l'étude de risque du premier scénario

	Acteur	Capacité	Opportunité	Motivation	Probabilité	Impact	Risque
Scénario i	Tricheur	4	4	4	4	2	8
	C.O.	1	4	1	2	2	4
	Concurrents	2	4	2	2.67	2	5.33

Dans ce cas-ci, ce sont les tricheurs qui sont la plus grande menace pour notre entreprise étant donné que le risque leur étant associé pour ce scénario (8) est le plus élevé si on le compare aux 2 autres. C'est logique, car les tricheurs sont ceux qui auront le plus de chance de réaliser un scénario de tricherie.

Tableau 2: Résultats de l'étude de risque du second scénario

	Acteur	Capacité	Opportunité	Motivation	Probabilité	Impact	Risque
Scénario ii	Tricheur	1	4	1	2	4	8
	C.O.	4	4	1	3	4	12
	Concurrents	2	4	4	3.33	4	13.33

Dans ce cas-ci, ce sont les sites de poker concurrents qui sont la plus grande menace pour notre entreprise étant donné que le risque leur étant associé pour ce scénario (13.33) est le plus élevé si on le compare aux 2 autres. Cependant, ils sont suivis de près par le crime organisé (12). C'est logique, car en rendant la concurrence indisponible, les sites concurrents auront probablement plus de clients.

Tableau 3: Résultats de l'étude de risque du troisième scénario

	Acteur	Capacité	Opportunité	Motivation	Probabilité	Impact	Risque
Scénario iii	Tricheur	1	3	1	1.67	3	5
	C.O.	4	3	4	3.67	3	11
	Concurrents	1	3	2	2	3	6

Dans ce cas-ci, c'est le crime organisé qui est la plus grande menace pour notre entreprise étant donné que le risque leur étant associé pour ce scénario (11) est le plus élevé si on le compare aux 2 autres. C'est logique, car le but du crime organisé est très fort probablement d'obtenir des informations confidentielles sur les utilisateurs, comme le suggère ce scénario.

Note : Tel que vu en classe, peu importe la méthode utilisée pour calculer la probabilité, il suffit d'être consistant entre les divers scénarios. Ici, nous avons fait la moyenne des valeurs obtenues pour chaque facteur (capacité, opportunité et motivation) afin d'obtenir la probabilité, puis nous avons multiplié la probabilité par l'impact pour obtenir le risque. **bien de le préciser !**

d) Pour chacune des situations suivantes expliquez quel(s) paramètre(s) changerai(en)t et dans quel sens (plus grand, plus petit). Quelle(s) conséquence(s) pour la gestion du risque ?

1. Votre compagnie de poker remporte un très grand succès et dépasse tous vos concurrents.

Le paramètre de motivation augmente. Les malfaiteurs auraient plus tendance à vouloir s'en prendre à nous étant donné que nous sommes devenus une plus grosse cible et que nous attaquer pourrait rapporter plus gros. **oui**

Par conséquent, le risque augmente lui aussi. Il faudra donc augmenter nos contre-mesures pour gérer celui-ci adéquatement.

2. Votre patron a refusé de payer les pots-de-vin réclamés par la mafia locale

La motivation de la mafia augmente, étant donné que celle-ci aura quelque chose contre nous.

Par conséquent, le risque augmente lui aussi. Il faudra donc augmenter nos contre-mesures pour gérer celui-ci adéquatement. **oui**

3. Votre patron fait l'acquisition d'un tout nouveau système de détection des tricheurs très performant.

L'opportunité des tricheurs diminue, car ils auront moins de facilité à saisir une occasion de tricher.

Par conséquent, le risque diminue lui aussi. **oui**

e) Un vendeur vous propose un service de surveillance à distance pour faire de la détection d'intrusion sur vos serveurs. Il suffit d'installer un logiciel de surveillance et de contrôle à distance pour permettre à ce fournisseur de détecter et combattre les intrusions. Celui-ci vous offre le service à très bon marché (5 000 \$ par mois pour une surveillance 24h sur 24) puisqu'il vient d'ouvrir un nouveau centre d'opération dans un pays de l'ex-Union Soviétique où la main d'œuvre coûte une fraction de la main d'œuvre au Canada. Refaites la grille de la question c) pour le scénario iii) en prenant en compte la mesure proposée. Est-ce que vous croyez que cette offre en vaut la chandelle ? Est-ce que votre recommandation s'applique dans toutes les circonstances ?

Tableau 4: Résultats de l'étude de risque du troisième scénario en tenant compte de la mesure proposée

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

Cette offre ne semble pas en valoir la chandelle. En effet, même si l'opportunité pour tous les acteurs est descendue à 1 plutôt que 3, pour certains acteurs, entre autres pour le crime organisé, le risque demeure élevé. Par exemple, pour le crime organisé, nous sommes passés de 11 à 9, ce qui reste encore vraiment élevé.

Donc, le risque pour ce scénario n'a pas diminué de manière assez significative pour que ça vaille la peine de débourser de l'argent. En plus, cela demanderait beaucoup d'efforts à déployer et cela pourrait nous exposer à d'autres risques, surtout considérant qu'il faut faire affaire à un autre pays où les règles en matière de sécurité sont très différentes d'ici.

Par contre, dans d'autres circonstances où les acteurs n'auraient pas des capacités si élevées et que le risque avait vraiment diminué, cela en aurait probablement valu la peine. Aussi, si cela n'avait pas été dans un autre pays, cela aurait encore une fois pu être plus considérable.

ok pour le raisonnement (attention cependant de ne pas essayer d'interpréter les valeurs du risque, comme vous l'avez précisé au-dessus le calcul de probabilité peut varier, par conséquent la valeur de risque doit être uniquement comparative)

Partie B

Question 1 - Codage (1.25)

0.8/1.25 Attention j'ai trouvé cette partie très similaire à celle de Pier-Luc et Ming !!

Cas où le codage est inchangé :

a) Expliciter les alphabets σ , T et T' qui sont respectivement les alphabets pour la sortie de la source, du codeur et du bloc de chiffrement.

- $\sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}; |\sigma| = 10$ ok
- $T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}; |T| = 10$ non, c'est un encodage ASCII donc on passe en binaire comme pour T'
- $T' = \{0, 1\}; |T'| = 2$ ok

b) Identifiez les langages provenant des alphabets σ , T et T'

- Langage produit par σ : Une chaîne composée de 4 chiffres de 0 à 9, pouvant se répéter. Il s'agit du NIP entré par l'utilisateur. ok
- Langage produit par T : Une chaîne composée de 8 chiffres de 0 à 9 où les 4 derniers chiffres sont une répétition des 4 premiers. Les chiffres peuvent apparaître plus d'une fois. Il s'agit du NIP entré par l'utilisateur, répété 2 fois. un bloc de 64 bits du coup après encodage
- Langage produit par T' : Une chaîne de 64 chiffres (0 ou 1). Les chiffres peuvent apparaître plusieurs fois. Proviennent du chiffrement DES du NIP répété 2 fois. ok

c) Ensuite, identifiez les attaques auxquelles le système est vulnérable. Pour identifier ces attaques, rappelez-vous qu'un attaquant peut connaître parfaitement le fonctionnement des boîtes de codage et chiffrement mais qu'il n'a bien sûr pas accès à la clé. Aussi, un attaquant peut intercepter tous les messages chiffrés et même les modifier.

Comme le système utilise le chiffrage avec l'algorithme DES, cela signifie que la clé ne comporte que 56 bits. Aussi, nous savons que l'attaquant a accès potentiellement au texte non chiffré et au texte chiffré, mais pas la clé de chiffrement (donc impossible de faire une attaque texte chiffré choisi). Cela donne donc la possibilité à 3 types d'attaques :

- Force brute (texte chiffré seulement) : On essaie toutes les possibilités de clés afin de déterminer quelle clé déchiffre le message (en regardant si le message semble correct, dans ce cas-ci si le NIP fonctionne). Comme la clé de l'algorithme DES ne contient que 56 caractères, nous avons un nombre relativement limité de possibilités (2^{56}). Par la suite, il sera possible de déchiffrer tous les messages en utilisant cette clé. ok
- Texte connu : On a accès au texte non chiffré ainsi qu'au texte chiffré. Avec ces informations, on peut tenter de déterminer la clé utilisée pour chiffrer le message et la réutiliser dans le futur. ok
- Attaque dictionnaire (texte clair choisi) : On a la possibilité de choisir un texte (en déterminant les symboles susceptibles d'être émis par la source) et d'obtenir sa valeur chiffrée. Ainsi, on peut essayer plusieurs valeurs de texte clair et tenter de déterminer laquelle nous permet d'avoir une correspondance avec le texte chiffré. ok

Aussi attaques par rejet ou modification de message, et possibilité de password reset (important pour les questions suivantes)

d) Pour chacune des attaques identifiées au c), montrez à l'aide de traces d'exécution comment vous les effectueriez. Pour cela, utilisez les scripts transBase et recepBase qui implémentent respectivement les blocs source+codeur+chiffrement et déchiffrement+décodeur+récepteur.

```
└─(virtualenv)─(inf4420a@inf4420a)─[~/Desktop/utilitaireTP1/Codage]
  └─$ python transBase.py 1234
    ??!S??4yE
```

Figure 14: Démonstration du texte du NIP chiffré

```
└─(virtualenv)─(inf4420a@inf4420a)─[~/Desktop/utilitaireTP1/Codage]
  └─$ python transBase.py 0000 > nip0
  └─(virtualenv)─(inf4420a@inf4420a)─[~/Desktop/utilitaireTP1/Codage]
    └─$ python recepBase.py < nip0
      0000
```

Figure 15: Démonstration de codage ainsi que de décodage d'un NIP

```
└─(virtualenv)─(inf4420a@inf4420a)─[~/Desktop/utilitaireTP1/Codage] ↵ sur C
  └─$ python transBase.py 0000 | xxd
    00000000: 91ce cde7 a380 7a01 .....z.
  └─(virtualenv)─(inf4420a@inf4420a)─[~/Desktop/utilitaireTP1/Codage]
    └─$ echo 1955913 22-09-2021
      1955913 22-09-2021 Barre verticale - Wikipédia
```

Figure 16: Démonstration du texte chiffré d'un NIP 0000 en éditeur hexadécimal

```
└─(virtualenv)─(inf4420a@inf4420a)─[~/Desktop/utilitaireTP1/Codage]
  └─$ python trans1.py 1234 > nip1
  └─(virtualenv)─(inf4420a@inf4420a)─[~/Desktop/utilitaireTP1/Codage]
    └─$ python recep1.py < nip1
      1234
```

Figure 17: Démonstration des traces d'exécution du premier codage

```
└─(virtualenv)─(inf4420a@inf4420a)─[~/Desktop/utilitaireTP1/Codage]
  └─$ python trans2.py 2222 > nip2
  └─(virtualenv)─(inf4420a@inf4420a)─[~/Desktop/utilitaireTP1/Codage]
    └─$ python recep2.py < nip2
      Delai de transmission suspect, operation annulee
```

Figure 18: Démonstration des traces d'exécution du second codage

```

└─(virtualenv)─(inf4420a㉿inf4420a)─[~/Desktop/utilitaireTP1/Codage]
$ python trans3.py 3333 > nip3

└─(virtualenv)─(inf4420a㉿inf4420a)─[~/Desktop/utilitaireTP1/Codage]
$ python recep3.py < nip3
Délai de transmission suspect, opération annulée

```

Figure 19: Démonstration des traces d'exécution du troisième codage

Cas où l'on change de codage : Les captures d'écrans et ce début de phrase incomplète sont exactement identiques au rendu de Pier-Luc et Ming !!!!!!

e) Pour chacun des trois codages, dites quelles attaques du c) ils permettent de bloquer en justifiant votre démarche

Tout d'abord, le premier codage ajoute un nombre de 48 bits aléatoirement pour la protection. Effectivement, il y a 2^{48} combinaisons possibles rendant ainsi la recherche de la clé plus difficile car les bits aléatoires rendent confus l'attaquant. Ainsi, les trois types d'attaques précédentes dépendent toutes de la clé, les rendant ainsi difficiles à effectuer. **ok, bloque les attaques par force brute**

Ensuite, le second codage a pour effet d'ajouter un horodatage (timestamp) de 16 bits et réduit le nombre de bits générés aléatoirement. Ainsi, il y a une diminution de la complexité si on le compare au premier codage. Ici nous avons 2^{16} (65536) combinaisons possibles plus les 10 000 combinaisons de NIP. Or, la banque se sert du horodatage pour valider les messages. Les messages ayant un vieux horodatage se font invalider. En effet, si la durée est très longue, l'attaquant possède plus de temps pour déchiffrer les combinaisons. En ayant un temps réduit, l'efficacité de l'attaque brute va diminuer. Effectivement, l'attaquant possède peu de temps avant que l'horodatage ne devienne invalide. **ok, évite les attaques par rejet et modification**

Pour terminer, le troisième codage ne génère aucun nombre aléatoire. Ce dernier fait l'utilisation de 16 bits de l'ancien NIP et l'utilisation de 2 bits de parité ainsi que l'utilisation de 16 bits pour le horodatage (timestamp). Ainsi, l'entropie du système diminue comparativement aux autres codages. De plus, en sachant qu'un ancien NIP peut être reconnu, il y a une diminution de la méconnaissance. Donc, les chances et les probabilités que les attaques réussissent sont plus élevées. Nous concluons que les trois attaques ne pourront pas être bloquées efficacement et que les chances que ces attaques soient bloquées par ce mode de codage sont faibles.

non, l'entropie reste quand même élevée, on conserve le timestamp pour éviter le rejet et en plus on évite un possible password reset car il faut connaître l'ancien mdp pour pouvoir changer de NIP

f) Selon vous quel est le meilleur codage ? Pourquoi ?

La meilleure option est la première, car en ajoutant une série de caractères aléatoires, on augmente l'entropie. Le message transmis devient donc plus difficile à déchiffrer. De plus, en comparaison avec les autres codages, le premier codage présente 2^{48} combinaisons possibles contrairement au codage 2 qui en présente 2^{16} . Aussi, l'ajout d'un horodatage est bien, mais un attaquant a toujours la possibilité de déchiffrer lors de la durée permise, ce qui rend l'utilité du horodatage pas très efficace. Pour ce qui est du codage 3, ce dernier n'est pas sécuritaire car il fait l'utilisation d'un ancien NIP et n'ajoute pas de nombres aléatoires. **en conséquence de mon commentaire précédent, le 3ème bloque plus d'attaques que les 2 précédents, il est donc à privilégier**

Question 2 - Certificats à clé publique, HTTPS et SSL (1) 1/1

Pour faire de l'hameçonnage, les pirates créent un site identique à un site d'une institution bancaire pour faire croire à la victime qu'elle se trouve sur le véritable site. Lorsque la victime entre ses informations confidentielles, son numéro de compte et son mot de passe par exemple, pour effectuer ses transactions, un message d'erreur tel que « le serveur est plein et ne peut traiter la requête » est retourné à la victime et ses informations confidentielles sont envoyées aux pirates.

Connectez-vous au site de la Caisse Desjardins à l'adresse https://www.desjardins.com.

a) Quelle est la différence entre le protocole http et https dans l'url ? ok

Les deux sont des protocoles de transfert hypertexte qui permettent à des données web d'être affichées dans un navigateur lors d'une requête. Par contre, contrairement à en HTTPS, en HTTP, les données ne sont pas chiffrées. Les requêtes et les réponses sont envoyées en texte clair. Ainsi, tout le monde peut accéder aux informations. [1]

HTTPS est pour sa part plus avancé et plus sécurisé. Le "s" signifie d'ailleurs "secure". Celui-ci permet de transmettre des informations encryptées avec le protocole TLS ou SSL. Ceci a pour effet de protéger les informations des attaquants. De plus, HTTPS permet l'authentification des serveurs web. Donc, les utilisateurs sont assurés que le serveur est le propriétaire du site web à l'aide d'un certificat. [2]

Quand vous cliquez sur le cadena dans la barre d'adresse (favicon), vous obtenez des informations sur le certificat à clé publique de la Caisse Desjardins.

b) Qu'est ce qu'un certificat à clé publique ? A quoi sert-il ? ok

Quand nous consultons un site Web qui utilise le protocole HTTPS (qui est destiné à assurer la sécurité de la connexion), le serveur du site Web présente un certificat pour prouver l'identité du site aux navigateurs tels que Chrome. Un certificat à clé publique est utilisé afin d'identifier ainsi que d'authentifier des personnes ou des machines pour assurer leur validité. Le certificat est délivré par un organisme de confiance et atteste la validité des informations se trouvant sur la page Web. Le certificat permet de lier la clé publique à une organisation tout en vérifiant l'identité de l'organisation. [3]

c) Dans un tableau, énumérez les principaux champs d'un certificat à clé publique. Pour chacun des champs, donnez la valeur correspondante du certificat de la Caisse Desjardins. ok

Tableau 5: Champs principaux d'un certificat à clé publique Desjardins

Champ du certificat	Valeur correspondante
Version	V3
Serial number	7df71e2198beda6169ebad0406d06d0a
Signature algorithm	sha256RSA

Signature hash algorithm	sha256
Issuer	CN = Entrust Certification Authority - L1K OU = (c) 2012 Entrust, Inc. - for authorized use only OU = See www.entrust.net/legal-terms O = Entrust, Inc. C = US
Valid from	August 16, 2021 3:54:28 PM
Valid to	August 16, 2022 3:54:28 PM
Subject	CN = www.desjardins.com O = Mouvement Desjardins L = Montreal S = Quebec C = CA
Public key	30 82 01 0a 02 82 01 01 00 d6 83 53 45 ea bf 8b f6 22 99 ae e5 b8 15 fc 50 8a b0 0b b2 6b 39 eb 98 e4 d3 c5 e8 f5 2d 2d 5b b1 d4 52 81 61 ae a5 85 75 e2 5d cf 34 14 35 be ca 76 27 78 05 36 36 56 b2 43 1d b1 a4 60 be fd 39 3c 39 35 25 10 1d dc 6f 42 f2 2c 39 db 37 f7 2b 3b a0 01 7c fb ee 38 5e ca ba 0f 50 7c 7a 85 af f3 bf a0 2d 59 11 66 49 88 b8 fd 6a e8 e1 b8 6d 0e 59 a1 15 a9 73 28 63 8f 90 7f 8c e0 9b 79 bc ba 36 f2 a9 e2 fa 4e 72 2b 99 0f cb db 8a 9d 01 93 ef b8 8f ed 80 47 59 8e 2c a1 71 fd 80 b6 8d 17 fc ce 08 30 97 3c 4c c2 d9 dd 95 fa 5a c8 8c 59 65 53 25 26 fc 95 22 20 0f 57 b2 45 36 bf 33 d4 d5 81 25 53 1c 99 ec 77 09 0b d2 1a 6f e3 60 72 37 e3 7d 27 4a 7c 2a d9 3b 80 a1 b9 6e 6b 97 1e 7b f8 c3 78 83 49 61 da 63 8f b5 27 78 bf 63 e8 e3 20 8c 41 e3 10 18 f1 62 c0 24 dd 75 87 02 03 01 00 01
Public key parameters	05 00
Key usage	Digital Signature, Key Encipherment (a0)
Thumbprint	274935a34e31312fb2df8a74466453213cb4cb96

d) Comment votre navigateur vérifie-t-il l'identité du propriétaire du site que vous avez visité?

Le navigateur va vérifier l'identité du propriétaire en plusieurs étapes. D'abord, le navigateur va faire la validation de l'intégrité du certificat en utilisant la clé publique et en confirmant la signature. Ensuite, le navigateur fera la vérification de la période de validité du certificat. Par la suite, le navigateur va faire la vérification du statut de révocation du certificat pour s'assurer que le Certificate Authority (CA) a révoqué le certificat. Enfin, le navigateur va vérifier l'organisation qui a attribué la clé publique ainsi que le propriétaire du site qui a obtenu le certificat. [4] **exact**

e) Qu'est-ce qu'un Certificate Authority (CA) ? A quoi sert-il ? Pourquoi observe-t-on deux CA lorsqu'on examine dans le navigateur le certificat de la Caisse Desjardins. Citez-les.

Un Certificate Authority est une organisation de confiance qui est approuvée pour émettre des certificats de sécurité. La caisse populaire Desjardins en possède deux puisque ces derniers suivent un modèle hiérarchique. Le premier CA émet le certificat, c'est-à-dire une certification parent qui certifie la CA et le second CA le reçoit (*intermediate certificate*). Les deux certificats sont Entrust Root Certification Authority -G2 ainsi que Entrust Certification Authority - L1K. [5] **ok**

f) Est-il risqué d'accepter dans votre navigateur un CA ? Pourquoi ?

Il est risqué d'accepter dans notre navigateur un CA tout dépendant de la confiance du CA. En effet, pour une autorité qui fait d'autres certificats, lorsqu'on accepte des CA dans le navigateur ceux-ci seront directement acceptés par le navigateur. Ainsi, nous allons faire confiance à des certificats produits par ce même CA sans les vérifier. Lorsqu'un certificat est auto-signé, ceci est risqué. [6] **oui**

g) Quelle est la différence entre un certificat auto-signé et un certificat TLS ? Pourquoi ne faut-il pas faire confiance à un site web dont le certificat est auto-signé?

Un certificat auto-signé est un certificat émis par des sociétés qui n'ont pas été validées par une CA. Un certificat auto-signé assure la confidentialité des échanges à l'intérieur d'une société. Des sites internet qui font l'utilisation de certificats auto-signés montrent des messages d'avertissemens qui sont parfois ignorés par les utilisateurs, ce qui les met en danger car la page internet est vulnérable aux menaces. Donc, il ne faut pas faire confiance à ce type de certificat car il ne garantit pas la sécurité des échanges. À l'opposé, les certificats TLS sont émis par des CA. C'est une version plus sécuritaire et signée par un organisme de certification. Les TLS permettent la sécurité des échanges avec des utilisateurs dans un site internet public. [7] **oui**

Question 3 - Chiffrement par bloc et modes d'opération (0.5) 0.5/0.5

L'administrateur réseau de la compagnie WebFacile se soucie des problèmes de vol de mot de passe par des malwares « sniffant » les fichiers textes. N'arrivant pas à se rappeler de tous les mots de passe de son réseau il décide de les enregistrer sous forme d'image puis de les chiffrer. Il décide d'utiliser un chiffrement AES 256 bits car il sait que celui-ci est sécuritaire. Cependant, ne comprenant pas bien à quoi correspondent les différents modes d'opération il utilise le premier : ECB (Electronic Code Book).

a) Le fichier mdp.jpg est un des mots de passe de l'administrateur enregistré sous forme d'image. À l'aide du script python AES.py, chiffrer ce fichier en mode ECB. (Exécutez le script sans argument pour connaître son fonctionnement). Observez le fichier de sortie et commentez.

```
(virtualenv)-(inf4420a@inf4420a)-[~/Downloads/utilitaireTP1/ChiffrementBLOC]
$ python AES.py
Erreurs 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 chiffré (facultatif)

(virtualenv)-(inf4420a@inf4420a)-[~/Downloads/utilitaireTP1/ChiffrementBLOC]
$ python AES.py -i mdp.jpg -m ECB -o resultECB.jpg

(virtualenv)-(inf4420a@inf4420a)-[~/Downloads/utilitaireTP1/ChiffrementBLOC]
$ echo 1956576 2021-09-22
1956576 2021-09-22
```

Figure 20: Chiffrement du fichier jpeg avec AES-256 en mode ECB



Figure 20: Fichier de sortie en mode ECB

Avec ECB, on voit très facilement les blocs et les lettres formées. C'est normal, car avec ce type de chiffrement, nous avons une grande correspondance entre le texte clair et le texte chiffré puisque tous les blocs sont chiffrés de la même façon. [ok](#)

b) Chiffrez maintenant le fichier en mode CBC. Observez le fichier puis commentez.

```
└─(virtualenv)─(inf4420a@inf4420a)─[~/Downloads/utilitaireTP1/ChiffrementBLOC]
  └─$ python AES.py -i mdp.jpg -m CBC -o resultCBC.jpg

└─(virtualenv)─(inf4420a@inf4420a)─[~/Downloads/utilitaireTP1/ChiffrementBLOC]
  └─$ echo 1956576 2021-09-22
  1956576 2021-09-22
```

Figure 21: Chiffrement du fichier jpeg avec AES en mode CBC



Figure 22: Fichier de sortie en mode CBC

Avec CBC, on ne voit pas bien les lettres affichées. C'est normal, car ce type de chiffrement par bloc ne donne pas une grande correspondance entre le texte clair et chiffré, il y a plus de chaos. Cela s'explique par le fait que le chiffrement de chaque bloc dépend du bloc précédent. C'est donc une meilleure méthode de chiffrement par bloc, étant donné qu'il n'est pas facile de déchiffrer le mot de passe. [oui](#)

c) Concluez sur l'importance des modes d'opération des algorithmes de chiffrement par bloc.

On voit que dans le cas présent, le mode de chiffrement CBC est meilleur, car grâce à une meilleure diffusion (un changement dans le message impacte le message ailleurs) et une meilleure confusion (il est difficile d'établir une relation évidente entre l'input et l'output), il devient plus difficile de déchiffrer le message qu'avec ECB qui n'a pas une bonne diffusion engendrant alors une mauvaise confusion.

Cela s'explique entre autres par le fait que ECB chiffre toujours chaque bloc de la même manière, alors que pour CBC, le chiffrement dépend du bloc chiffré précédent. Pour améliorer encore plus la sécurité avec CBC en diminuant les chances de pouvoir faire de l'analyse fréquentielle, il est aussi recommandé de changer le vecteur d'initialisation à chaque fois.

Bref, il est important de prêter une attention particulière au mode de chiffrement par bloc, car même si AES est une méthode sûre, le mode de chiffrement choisi peut faire en sorte que notre algorithme de chiffrement devient vulnérable. **exactement**

Question 4 - Organisation des mots de passe en UNIX/Linux (1) 0.95/1

a) Examinez le fichier /etc/passwd. Contient-il des mots de passe ? Pourquoi? Quelles sont ses permissions d'accès? Pourquoi ?

```
(inf4420a@inf4420a)-[~]
$ echo 1956576 2021-09-22
1956576 2021-09-22

(inf4420a@inf4420a)-[~]
$ cat /etc/passwd
root:x:0:0::root:/root:/usr/bin/zsh
daemon:x:1:1::daemon:/usr/sbin/nologin
bin:x:2:2::bin:/usr/sbin/nologin
sys:x:3:3::sys:/dev:/usr/sbin/nologin
sync:x:4:65534::sync:/bin:/bin/sync
games:x:5:60::games:/usr/games:/usr/sbin/nologin
man:x:6:12::man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7::lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8::mail:/var/mail:/usr/sbin/nologin
news:x:9:9::news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10::uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13::proxy:/bin:/usr/sbin/nologin
www-data:x:33:33::www-data:/var/www:/usr/sbin/nologin
backup:x:34:34::backup:/var/backups:/usr/sbin/nologin
list:x:38:38::Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39::ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41::Gnats Bug-Reporting System (admin)::/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:101::systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103::systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104::systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
mysql:x:104:110::MySQL Server,,,:/nonexistent:/bin/false
tss:x:105:111::TPM software stack,,,:/var/lib/tpm:/bin/false
strongswan:x:106:65534::/var/lib/strongswan:/usr/sbin/nologin
ntp:x:107:112::/nonexistent:/usr/sbin/nologin
messagebus:x:108:113::/nonexistent:/usr/sbin/nologin
redsocks:x:109:114::/var/run/redsocks:/usr/sbin/nologin
rwhod:x:110:65534::/var/spool/rwho:/usr/sbin/nologin
iodine:x:111:65534::/run/iodine:/usr/sbin/nologin
miredo:x:112:65534::/var/run/miredo:/usr/sbin/nologin
_rpc:x:113:65534::/run/rpcbind:/usr/sbin/nologin
usbmux:x:114:46::usbmux daemon...:/var/lib/usbmux:/usr/sbin/nologin
```

Figure 23: Examination du fichier /etc/passwd

Le fichier contient les données de connexion des utilisateurs. Par contre, il ne contient pas les mots de passe (ils sont remplacés par un x), étant donné que ce fichier est accessible par n'importe quel utilisateur. Cependant, les permissions d'accès sont limitées à la lecture pour tous les utilisateurs et c'est seulement le root qui peut modifier ce fichier, puisqu'on ne veut pas que n'importe qui puisse changer les informations. **ok**

b) Observez les fichiers passwd et shadow qui se trouvent sous le répertoire /etc/. Ajoutez un utilisateur avec la commande:

`$ useradd -g users -s/bin/bash -m NOM`

avec NOM= le nom de l'utilisateur que vous ajoutez Donnez un password à l'utilisateur que vous avez créé avec la commande:

`$ passwd NOM`

Observez ce qui se passe dans les fichiers passwd et shadow. Lequel ou lesquels de ces deux fichiers sont modifiés ? Pourquoi ?

```
(inf4420a@inf4420a)-[~]
$ useradd -g users -s/bin/bash -m LAURA
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.

(inf4420a@inf4420a)-[~]
$ sudo useradd -g users -s/bin/bash -m LAURA

(inf4420a@inf4420a)-[~]
$ passwd LAURA
passwd: You may not view or modify password information for LAURA.

(inf4420a@inf4420a)-[~]
$ sudo passwd LAURA
New password:
Retype new password:
passwd: password updated successfully
```

Figure 24: Ajout d'un utilisateur et nouveau password

Voici ce qui a été ajouté à la fin du fichier /etc/passwd :

```
Systemd-Coredump:x:999:999:Systemd-Coredump
LAURA:x:1001:100 :: /home/LAURA:/bin/bash
```

Figure 25: Affichage du fichier passwd

Voici ce qui a été ajouté à la fin du fichier /etc/shadow :

```
LAURA:$y$j9T$E.QaQM8jgkhNv/pFYTEEEx1$Sr04Q6SXB0SrSpE2RJOPpoMZ08WT/4fjVuocNt507LD:18892:0:99999:7:::
```

Figure 26: Affichage du fichier shadow

Bref, les 2 fichiers ont été modifiés, car le premier contient les utilisateurs et le 2e les mots de passe. Si on ajoute un utilisateur avec un mot de passe, il est donc normal que les 2 soient modifiés. **ok**

c) Changez le mot de passe de l'utilisateur que vous venez de créer avec la commande
\$ `passwd NOM`
Qu'est-ce que vous remarquez dans les fichiers `passwd` et `shadow`? Lequel de ces deux fichiers change? Pourquoi ? Où se trouve donc l'information du mot de passe? Quelles sont les permissions du fichier `shadow` et pourquoi ?

```
└─(inf4420a㉿inf4420a)-[~]
└─$ sudo passwd LAURA
New password:
Retype new password:
passwd: password updated successfully

└─(inf4420a㉿inf4420a)-[~]
└─$ echo 1956576 2021-09-22
1956576 2021-09-22
```

Figure 27: Changement du mot de passe

Contenu dans `/etc/passwd` : `LAURA:x:1001:100 :: /home/LAURA:/bin/bash`

Figure 28: Affichage du fichier `passwd` après le changement du mot de passe

Contenu dans `/etc/shadow` :

`LAURA:yj9T$h02v3vAjFPqAlXo4CRf1h/$gfExg718x/1WCcTKifD0EFIm6cmaylJSU1mv4yoGbz6:18892:0:99999:7:::`

Figure 29: Affichage du fichier `shadow` après le changement du mot de passe

Donc, on voit que c'est uniquement `/etc/shadow` qui change, car on a modifié un mot de passe. C'est parce que c'est ce fichier (et non `/etc/passwd`) qui contient l'information du mot de passe. Le fichier `/etc/shadow` est uniquement accessible par root (lecture/écriture), car il s'agit de données sensibles et on ne veut pas que n'importe qui y ait accès ou puisse les modifier. ok

d) Changez à nouveau le mot de passe du même utilisateur et donnez-lui le *même* mot de passe. Est-ce que les informations du mot de passe ont changé? Pourquoi?

```
└─(inf4420a㉿inf4420a)-[~]
└─$ sudo passwd LAURA
New password:
Retype new password:
passwd: password updated successfully

└─(inf4420a㉿inf4420a)-[~]
└─$ echo 1956576 2021-09-22
1956576 2021-09-22
```

Figure 30: Changement du mot de passe en lui donnant le même mot de passe

Contenu /etc/shadow :

```
LAURA:$y$j9T$9k9CoP.sNjqorHsfe0STG1$LNPLfcsEQ2gmyZ0n/QIK4lNxtzL2nl.1Rf56Ni/WWz0:18892:0:99999:7:::
```

Figure 31: Affichage du fichier shadow après le changement du mot de passe avec le même mot de passe

Oui, les informations du mot de passe ont changé. Cela signifie donc que le hash utilisé n'est pas toujours le même, ce qui est rassurant pour que les données restent confidentielles.

Cela s'explique par le fait que les informations du mot de passe contiennent 3 parties, soit le type, le salt et le mot de passe hashé (\$type\$salt\$hashed). Le *salt* change pour chaque mot de passe, il est aléatoire. Celui-ci influence le ~~chiffrement~~, ce qui va évidemment avoir un impact sur la valeur du mot de passe hashé.

hashage ok

Source : <https://linuxize.com/post/etc-shadow-file/>

e) Créez un deuxième utilisateur en suivant les mêmes étapes qu'au point b. Éditez ensuite le fichier shadow et remplacez la valeur par défaut (!!) du champ de mot de passe de l'utilisateur que vous venez de créer par la valeur du même champ pour l'utilisateur que vous avez créé en premier (les éditeurs de texte nano et vim sont disponibles). Sauvegardez le fichier et quittez votre session. Essayez de vous connecter sur le compte du deuxième utilisateur mais avec le mot de passe que vous venez de copier. Est-ce que ceci est possible? Expliquez pourquoi. Quel est le problème?

```
└─(inf4420a㉿inf4420a)-[~]
$ sudo useradd -g users -s/bin/bash -m ANDIP

└─(inf4420a㉿inf4420a)-[~]
$ sudo passwd ANDIP
New password:
Retype new password:
passwd: password updated successfully

└─(inf4420a㉿inf4420a)-[~]
$ echo 1956576 2021-09-22
1956576 2021-09-22
```

Figure 32: Création d'un second utilisateur

```

└─(inf4420a㉿inf4420a)-[~]
└─$ sudo nano /etc/shadow

└─(inf4420a㉿inf4420a)-[~]
└─$ su - ANDIP
Password:
(Message from Kali developers)

└─ We have kept /usr/bin/python pointing to Python 2 for backwards
    compatibility. Learn how to change this and avoid this message:
    ⇒ https://www.kali.org/docs/general-use/python3-transition/
    (Run: "touch ~/.hushlogin" to hide this message)
└─(ANDIP㉿inf4420a)-[~]
└─$ █

```

Figure 33: Affichage du message après le changement d'utilisateur

Effectivement, on voit qu'il est possible de changer d'utilisateur en utilisant le mot de passe changé manuellement. C'est possible, car en prenant les accès nécessaires (avec sudo), nous avons réussi à avoir accès à la modification du fichier /etc/shadow et ainsi modifier le mot de passe d'un utilisateur. Cela comporte cependant plusieurs risques et ce n'est donc pas recommandé. Si, par exemple, on fait une erreur, on ne pourra potentiellement plus jamais se connecter sur le compte sur lequel on a changé le mot de passe en passant par l'accès en écriture. *c'est surtout le risque que si quelqu'un a accès au fichier en écriture il peut remplacer le hash d'un mdp d'un autre utilisateur par celui de son propre mdp et donc se connecter à sa place (d'où la non permission de lecture et d'écriture pour les non admins)*

f) Effacez cet utilisateur avec la commande

\$ userdel -r NOM

Qu'est-ce qui se passe dans passwd et shadow ?

ok

```

└─(inf4420a㉿inf4420a)-[~]
└─$ sudo userdel -r ANDIP
[sudo] password for inf4420a:
userdel: ANDIP mail spool (/var/mail/ANDIP) not found

└─(inf4420a㉿inf4420a)-[~]
└─$ echo 1956576 2021-09-22
1956576 2021-09-22

```

Figure 34: Suppression de l'utilisateur

On voit que l'utilisateur ANDIP a disparu dans le fichier /etc/passwd :

```

systemd-coredump:x:999:999:systemd Core Dumper
LAURA:x:1001:100 :: /home/LAURA:/bin/bash

└─(inf4420a㉿inf4420a)-[~]

```

Figure 35: Affichage du résultat après suppression de l'utilisateur dans le fichier passwd

Ainsi que dans le fichier /etc/shadow :

```
systemd-coredump:!*:18877:::::  
LAURA:$y$j9T$9k9CoP.sNjqorHsfe0STG1$LNPLf  
└─(inf4420a㉿inf4420a)-[~]
```

Figure 36: Affichage du résultat de la suppression dans le fichier shadow

Question 5 - Contrôle de qualité de choix de mot de passe (1) 1/1

a) Utiliser john the ripper avec le dictionnaire rockyou, et identifier le mot de passe correspondant à chaque utilisateur. Utilisez la commande suivante pour connaître l'emplacement # Locate rockyou ok

```
└─(inf4420a㉿inf4420a)-[~/Downloads/utilitaireTP1] qZX74KxLY.:18346:0:99999::::::  
$ john passwd file /ML/110CC23CK0VE3YCKLAEVZ/Ryw/1Vu5lhgv.9eexVsjR0GxY  
Using default input encoding: UTF-8  
Loaded 4 password hashes with 4 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])  
Cost 1 (iteration count) is 5000 for all loaded hashes  
Proceeding with single, rules:Single  
Press 'q' or Ctrl-C to abort, almost any other key for status  
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for performance.  
Warning: Only 1 candidate buffered for the current salt, minimum 8 needed for performance.  
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.  
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.  
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.  
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.  
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.  
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.  
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.  
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.  
Further messages of this type will be suppressed.  
To see less of these warnings, enable 'RelaxKPCWarningCheck' in john.conf  
Almost done: Processing the remaining buffered candidate passwords, if any.  
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist  
123456789      (simple)  
sunshine        (brian)  
monkey          (action)  
liverpool       (vladimir)  
4g 0:00:00:30 DONE 2/3 (2021-09-15 17:31) 0.1292g/s 416.7p/s 427.2c/s 427.2C/s idontknow..me  
Use the "--show" option to display all of the cracked passwords reliably  
Session completed  
└─(inf4420a㉿inf4420a)-[~/Downloads/utilitaireTP1]  
$ echo 1956576 2021-09-15  
1956576 2021-09-15
```

Figure 37: Affichage des mots de passes

Le mot de passe de simple est 123456789.

Le mot de passe de brian est sunshine.

Le mot de passe d' action est monkey.

Le mot de passe de vladimir est liverpool.

b) Calculez l'entropie maximale pour les alphabets suivants : ok

Pour connaître l'entropie maximale, il suffit d'utiliser la formule $H(S) = \log_2(N)$ où N représente le nombre de caractères possibles. Pour obtenir cette entropie, on considère que chaque caractère a autant de chance de se retrouver dans le mot de passe.

i. •[a-zA-Z]

$$H(S) = \log_2(26 + 26) = \log_2(52) = 5.7 \text{ bits}$$

ii. •[a-zA-Z0-9]

$$H(S) = \log_2(26 + 26 + 10) = \log_2(62) = 5.95 \text{ bits}$$

iii. •L'ensemble de la table ascii

$$H(S) = \log_2(128) = 7 \text{ bits}$$

Note : Nous considérons qu'il s'agit de la table ascii simplement, et non de la table ascii étendue.

c) Déduisez des résultats de b) un critère important pour qu'un mot de passe soit fort ok

Afin de permettre aux utilisateurs de créer un mot de passe fort, il faut permettre l'utilisation du plus de caractères possible. En effet, plus il y en a, plus l'entropie maximale est élevée, et donc plus on peut choisir un mot de passe avec une grande entropie, le rendant plus difficile à déchiffrer et apportant donc une meilleure sécurité. Il faut alors s'assurer qu'on fait bien un mélange de tous ces types de caractère.

d) Au vu des résultats de John the Ripper, donner 3 autres critères pour qu'un mot de passe soit fort.

1. Il ne faut pas utiliser un mot existant provenant du dictionnaire, car cela est plus facile à déchiffrer. En effet, il faut que les caractères soient sélectionnés de manière équiprobable. C'est pourquoi prendre une suite de caractères aléatoires (comme certains générateurs de mots de passe le permettent) est plus efficace que de prendre des mots existants, car plus les caractères sont aléatoires, plus on se rapproche de l'entropie maximale, et plus le mot de passe est difficile à déchiffrer et apporte donc une meilleure sécurité. ok
2. Il faut prendre un mot de passe qui est long afin d'augmenter le nombre de possibilités de mot de passe, comme par exemple une phrase de passe. ok
3. Il ne faut pas utiliser d'informations personnelles (comme le nom, la date de naissance, etc.) car cela peut-être facile à deviner. ok

e) A votre avis pourquoi il est conseillé de ne pas utiliser le même mot de passe partout.

Bien évidemment, le fait d'utiliser le même mot de passe partout fait en sorte que si un attaquant réussit à obtenir le mot de passe d'un de nos comptes, il aura ensuite très facilement accès à tous nos autres comptes qui utilisent ce même mot de passe. oui

Partie C

Question 1 - Échec du protocole RSA (0.75) 0.75/0.75

Malgré le fait que RSA soit considéré sécuritaire, si n est assez grand pour ne pas être factorisable, il faut tout de même faire attention. Bob utilise le chiffrement RSA avec un n assez grand pour ne pas être factorisable. Alice envoie à Bob un message dans lequel chaque caractère alphabétique est chiffré séparément avec la clé publique de Bob, les caractères avant le chiffrement étant représentés par des nombres de 0 à 25 ('A'=0, 'B'=1,..., 'Z'=25).

a) Décrire comment Ève peut facilement déchiffrer ce message.

Ève peut facilement déchiffrer ce message car chaque caractère est chiffré séparément. Ceci ressemble à une substitution mono-alphabétique car les nombres chiffrés correspondent à des lettres. Aussi, nous savons qu'une lettre est représentée par un chiffre selon la logique où 'A'=0, 'B'=1... 'Z'=25. Lorsque le code est déchiffré, nous obtenons la valeur de la lettre. Si les lettres avaient été attribuées de manière aléatoire par exemple 'A' = 3, 'B' = 20... il aurait été beaucoup plus difficile de trouver le message. Afin de déchiffrer le message, nous partons avec le message codé ainsi que les valeurs fournies e et n . Nous pouvons trouver la valeur encodée des lettres en faisant l'utilisation de la formule:

$$\text{lettre_chiffrée} = m^e \pmod{n}$$

Dans cette formule m est la lettre codée en valeur numérique correspondant à la lettre [0,25]. De plus, e et n forment la paire de valeurs de la clé publique. Le *lettre_chiffrée* est la valeur chiffrée que nous obtenons.

Afin de trouver le contenu du message, nous pouvons comparer les nombres obtenus, c'est-à-dire *lettre_chiffrée*, avec les valeurs du message codé. c'est ça

b) Récupérer le texte à déchiffrer et la clé publique dans le document INF4420A_TP1_Q1_A21_GX du site Moodle. Utilisez l'attaque décrite précédemment pour déchiffrer le texte, sans factoriser n , selon la clé. Donnez votre réponse en texte, pas en chiffres. ok

$e : 599$ N: 618391 Texte : {216825, 351641, 480183, 443797, 523241}

En appliquant la formule $\text{lettre_chiffrée} = m^e \pmod{n}$, on obtient ces correspondances :

Tableau 6: Affichage des lettres codées et des lettres chiffrées selon les lettres de l'alphabet

Lettre	Lettre codée	Lettre chiffrée
A	0	$0^{599} \pmod{618391} = 0$
B	1	$1^{599} \pmod{618391} = 1$

C	2	$2^{599} \pmod{618391} = 216825$
D	3	$3^{599} \pmod{618391} = 475032$
E	4	$4^{599} \pmod{618391} = 523241$
F	5	$5^{599} \pmod{618391} = 420930$
G	6	$6^{599} \pmod{618391} = 226931$
H	7	$7^{599} \pmod{618391} = 520723$
I	8	$8^{599} \pmod{618391} = 480183$
J	9	$9^{599} \pmod{618391} = 196387$
K	10	$10^{599} \pmod{618391} = 437951$
L	11	$11^{599} \pmod{618391} = 426124$
M	12	$12^{599} \pmod{618391} = 140172$
N	13	$13^{599} \pmod{618391} = 307490$
O	14	$14^{599} \pmod{618391} = 554086$
P	15	$15^{599} \pmod{618391} = 345083$
Q	16	$16^{599} \pmod{618391} = 278260$
R	17	$17^{599} \pmod{618391} = 351641$
S	18	$18^{599} \pmod{618391} = 443797$
T	19	$19^{599} \pmod{618391} = 8402$
U	20	$20^{599} \pmod{618391} = 458788$
V	21	$21^{599} \pmod{618391} = 596181$
W	22	$22^{599} \pmod{618391} = 536990$
X	23	$23^{599} \pmod{618391} = 96794$
Y	24	$24^{599} \pmod{618391} = 113032$
Z	25	$25^{599} \pmod{618391} = 57189$

Donc Texte = { 'C', 'R', 'I', 'S', 'E' }

c) Si vous regardez attentivement la liste de textes à déchiffrer pour votre groupe de laboratoire, vous remarquez probablement des textes chiffrés avec des « 0 » ou des « 1 ». Quelles conclusions additionnelles pouvez-vous tirer sur le contenu des messages pour assurer le bon fonctionnement du RSA ? ok

Dans l'énoncé, le nombre 0 représente la lettre 'A' et le nombre 1 représente la lettre 'B'. Nous utilisons la formule $lettre_chiffrée = m^e \pmod n$ et nous remarquons que les valeurs chiffrées sont les mêmes que les nombres originaux. En effet, A est $0^{599} \pmod{618391} = 0 \pmod{618391} = (618391 * 0) + 0 = 0$ et B est $1^{599} \pmod{618391} = 1 \pmod{618391} = (618391 * 0) + 1 = 1$. Ainsi, même si les lettres 'A' et 'B' passent par la formule, il est facile de faire leur détection. Donc, ces lettres font en sorte que la sécurité du RSA diminue et que le fonctionnement n'est pas bon pour l'utilisation de 'A' ainsi que de 'B'.

Question 2 - Déchiffrement "simple" (0.75) 0.75/0.75 bon raisonnement

Texte à déchiffrer :

JBOVQYOFBJVOMIVPOMJOMIVPOCYOIARMBLVPAMOLASPBOMTPOHBPQMORPKOJFOULPCPZOQDDJIA
MOQOVPCPBOJFOMTPOKPHIRKQMIEPOZJLAZIKOJFOULCPZOMJOCORDPQ@PBOMTPBPJFOQASOVQ
YOBPVJEPTIVOQASOQDDJIAMOQAJMTPBOIAOTIRORMPQSOOL

Fréquences obtenues pour chaque lettre :

```

(inf4420a@inf4420a) [~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ echo "JBOVQYOFBJVOMIVPOMJOMIVPOCYOIARMBLVPAMOLASPBOMTPOHBPQMORPKOJFOULPCPZOQDDJIA
MOQOVPCPBOJFOMTPOKPHIRKQMIEPOZJLAZIKOJFOULCPZOMJOCORDPQ@PBOMTPBPJFOQASOVQ
YOBPVJEPTIVOQASOQDDJIAMOQAJMTPBOIAOTIRORMPQSOOL" > texteChiffreC1.bin

(inf4420a@inf4420a) [~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./frequency < texteChiffreC1.bin
@ : 0.500 %
A : 5.000 %
B : 5.000 %
C : 2.500 %
D : 2.500 %
E : 1.000 %
F : 2.500 %
H : 1.000 %
I : 5.500 %
J : 6.500 %
K : 2.000 %
L : 3.000 %
M : 7.500 %
O : 19.000 %
P : 12.500 %
Q : 6.500 %
R : 3.000 %
S : 2.000 %
T : 3.000 %
U : 1.000 %
V : 5.000 %
Y : 1.500 %
Z : 2.000 %

(inf4420a@inf4420a) [~/Downloads/utilitaireTP1/Source - Entropie - Chiffrement]
$ echo 1956576 2021-09-22
1956576 2021-09-22

```

Figure 38: Affichage des fréquences obtenues pour chacune des lettres

Le caractère le plus présent est le O. On peut donc en déduire qu'il s'agit de l'espace.

Par la suite, le P = E, car ils ont une fréquence très semblable (12%).

Le caractère qui est ensuite le plus fréquent est le **M = T**, puis le Q ou le J qui doivent correspondre au A ou O, ce n'est pas certain.

Ensuite, si on se fie encore aux fréquences, **I = I.**

MJ apparaît 2 fois, entouré d'espaces. Comme M = T et que J = O ou A, on peut en déduire que **J = O**, car TA n'a pas de sens en anglais. Par conséquent, **Q = A.**

AM revient 3 fois et la fréquence de A ressemble à celle de N. Comme M = T, il pourrait s'agir de NT et donc **A = N.**

Le seul mot de 7 lettres commençant par A et terminant par OINT avec 2 lettres identiques entre les 2 est "appoint", donc **D = P.**

Le seul mot de 7 lettres commençant par ANOT et contenant un E est "another". Donc, **T = H** et **B = R.**

Le seul mot de 7 lettres commençant par THEREO possible est "thereof", étant donné que le N est déjà représenté par le A. Donc, **F = F.**

Comme le T est déjà utilisé, le seul mot de 5 lettres terminant par REAT est "great". Donc, **H = G.**

Le seul mot avec _ EGI _ ATI _ E est "legislative". Donc, **K = L**, **R = S** et **E = V.**

Le seul mot de 10 lettres commençant par INSTR et terminant par ENT encore possible est "instrument". Donc, **L = U** et **V = M.**

Le seul mot de 6 lettres commençant par MEM et terminant par ER est "member", donc **C = B.**

Le seul mot de 5 lettres commençant par UN et terminant par ER est "under", donc **S = D.**

Le seul mot avec _ OUN _ IL est "council" donc **Z = C.**

Le seul mot de 5 lettres terminant par UEBEC est "quebec", donc **U = Q.**

Le seul mot de 7 lettres commençant par SPEA et terminant par ER est "speaker", comme le R est déjà représenté. donc, **@ = K.**

Finalement, on voit facilement que **Y = Y** ("by", "may").

G, N, W et X n'apparaissent jamais. Il doit s'agir de Z, W, X et J (interchangeables, comme on a pas l'information)

Tableau 7: Affichage de la correspondance entre les lettres chiffrées et les vraies lettres

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_
Q	C	Z	S	P	F	H	T	I		@	K	V	A	J	D	U	B	R	M	L	E			Y		O

Texte déchiffré :

OR MAY FROM TIME TO TIME BY INSTRUMENT UNDER THE GREAT SEAL OF QUEBEC APPOINT A MEMBER
OF THE LEGISLATIVE COUNCIL OF QUEBEC TO BE SPEAKER THEREOF AND MAY REMOVE HIM AND
APPOINT ANOTHER IN HIS STEAD U

Références

- [1] <https://www.venafi.com/blog/what-are-differences-between-http-https-0#:~:text=In%20a%20Nutshell&text=The%20difference%20between%20the%20two,uses%20HTTPS%20has%20HTTPS%3A%2F%2F>
- [2] <https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/>
- [3] <https://www.anf.es/en/blog-que-es-un-certificado-electronico/>
- [4] <https://www.ufrst.univ-evry.fr/certsrv/helpcrt.html>
- [5] <https://www.globalsign.com/en/ssl-information-center/what-are-certificationAuthorities-trust-hierarchies>
- [6] <https://www.digital-dynamics.fr/FR/tutos/ajouter-exception-site-certificat-non-fiable-google-chrome.html>
- [7] <https://www.certificat-ssl.info/doc/definition-certificat-ssl>