



# École Polytechnique de Montréal

## Département de génie informatique et génie logiciel

---

### Rapport du TP1

Cybersécurité

Travail pratique 1

---

Fares Laadjel, 2297799

Julien Cyr, 2278776

## Table des matières

|   |           |
|---|-----------|
| <b>1 Travail demandé</b>  | <b>2</b>  |
| 1.1 Entropie et sources d'information (10 points) . . . . .             | 2         |
| 1.2 La librairie de Babel (10 points) . . . . .                         | 3         |
| 1.3 Histogrammes (5 points) . . . . .                                   | 4         |
| 1.4 Déchiffrement simple (15 points) . . . . .                          | 8         |
| 1.5 Masque jetable (20 points) . . . . .                                | 10        |
| 1.6 Communication à clé publique, HTTPS et SSL (5 points) . . . . .     | 12        |
| 1.7 Codage (9 points) . . . . .   | 16        |
| 1.8 Changement de codage (5 points) . . . . .                           | 17        |
| 1.9 Chiffrement par bloc et modes d'opération (4 points) . . . . .      | 18        |
| 1.10 Organisation des mots de passe en UNIX/Linux (12 points) . . . . . | 20        |
| 1.11 Choix des mots de passe (5 points + 5 bonus) . . . . .             | 22        |
| <b>Bibliographie</b>  | <b>24</b> |

# 1 Travail demandé

## 1.1 Entropie et sources d'information (10 points)

- Supposez un alphabet de 50 symboles et une source qui produit un fichier de 500 caractères où chaque symbole a une probabilité égale de survenir. Quelle sera l'entropie moyenne théorique par lettre ? Fournir le détail du calcul. /3

**Étape 1 : probabilité de chaque symbole.**

$$p_i = \frac{1}{50}, \quad \text{pour } i = 1, 2, \dots, 50.$$

**Étape 2 : entropie de Shannon.**

$$H(X) = - \sum_{i=1}^{50} p_i \log_2(p_i).$$

Puisque les  $p_i$  sont identiques :

$$H(X) = -50 \times \frac{1}{50} \times \log_2\left(\frac{1}{50}\right) = \log_2(50).$$

**Étape 3 : calcul numérique.**

$$\log_2(50) = \frac{\ln(50)}{\ln(2)} \approx \frac{3,9120}{0,6931} \approx 5,644.$$

*L'entropie moyenne théorique par lettre est donc d'environ 5,64 bits. Cela signifie qu'en moyenne, il faut environ 5,64 bits pour coder un symbole de cet alphabet sans perte.*

- Peut-on dire que cette entropie est maximale ? Pourquoi ? /2

*Oui, l'entropie est maximale. En effet, pour un alphabet donné, l'entropie atteint sa valeur maximale lorsque tous les symboles possibles sont équiprobables. Dans ce cas, la source est totalement imprévisible, ce qui maximise l'information moyenne par symbole.*

- En considérant seulement le contenu du fichier en lui-même, serait-il possible de réduire la taille du fichier en utilisant un algorithme de compression standard ? Expliquez en vous basant sur le principe du taux de compression. /3

*Non, il ne serait pas possible de réduire la taille du fichier à l'aide d'un algorithme de compression standard, en considérant uniquement son contenu.*

*En effet, la source produit des symboles équiprobables, ce qui implique une entropie maximale égale à  $\log_2(N)$  bits par symbole. Dans ce cas, le nombre moyen de bits nécessaires pour représenter chaque symbole correspond déjà à la limite théorique minimale. Il n'existe donc aucune redondance statistique exploitabile par un algorithme de compression sans perte.*

*Selon le principe du taux de compression, lorsque l'entropie par symbole est égale au nombre de bits requis dans le pire cas, le taux de compression est égal à 1, ce qui signifie qu'aucun gain de compression n'est possible.*

- Concluez, en faisant un lien avec vos précédentes réponses, ce qui permet de compresser des images, ou encore du texte suivi, sans nécessairement faire référence à un algorithme de compression spécifique. /2

*Les images et les textes suivis sont compressibles parce qu'ils ne sont pas constitués de symboles indépendants et équiprobables. Ils présentent des régularités et des dépendances, par exemple certaines valeurs de pixels ou certaines lettres et suites de lettres apparaissent beaucoup plus souvent que d'autres. Cette redondance réduit l'entropie moyenne par symbole par rapport au cas d'entropie maximale. Comme l'information n'est pas répartie uniformément, il est alors possible de représenter le contenu avec moins de bits sans perte, contrairement à une source aléatoire uniforme.*

## 1.2 La librairie de Babel (10 points)

- Avec le premier matricule, récupérez le texte à matricule, wall 1, shelf 1, volume 1, page 1. Puis, calculez l'entropie par octet de ce texte (h-ascii). /2

```
kali@kali:~$ cd Documents/utilitaireTP1/Source -> Entropie - Chiffrement /
kali@kali:~/Documents/utilitaireTP1/Source -> Entropie - Chiffrement$ ./Entropie - Chiffrement name t extext1.txt
Nombre total d'octets : 3240
Entropie de l'entrée : 4.889572
kali@kali:~/Documents/utilitaireTP1/Source -> Entropie - Chiffrement$ echo 2278776-2297799 $date
2278776-2297799 Sat 07 Feb 2026 02:46:43 PM EST
kali@kali:~/Documents/utilitaireTP1/Source -> Entropie - Chiffrement$
```

- Ajoutez, à la suite du texte initial, le texte trouvé avec le deuxième matricule. Puis, calculez l'entropie par octet de ce texte (h-ascii). /2

```
kali@kali:~$ cd Documents/utilitaireTP1/Source -> Entropie - Chiffrement /
kali@kali:~/Documents/utilitaireTP1/Source -> Entropie - Chiffrement$ ./Entropie - Chiffrement name t extext1.txt > texte_complet.txt
kali@kali:~/Documents/utilitaireTP1/Source -> Entropie - Chiffrement$ cat texte_complet.txt
kali@kali:~/Documents/utilitaireTP1/Source -> Entropie - Chiffrement$ ./h-ascii < texte_complet.txt
Nombre total d'octets : 6482
Entropie de l'entrée : 4.892866
kali@kali:~/Documents/utilitaireTP1/Source -> Entropie - Chiffrement$ echo 2278776-2297799 $date
2278776-2297799 Sat 07 Feb 2026 02:53:15 PM EST
kali@kali:~/Documents/utilitaireTP1/Source -> Entropie - Chiffrement$
```

- Discutez de la variation d'entropie que vous observez. /3

*On observe que l'entropie par octet varie très peu lorsqu'on ajoute le second texte au premier. Cette faible variation s'explique par le fait que l'entropie dépend uniquement de la distribution des symboles présents dans le texte.*

*En concaténant deux textes issus de la librairie de Babel, on modifie légèrement les fréquences des caractères, mais sans changer leur répartition globale de manière significative.*

*Comme les deux textes proviennent de la même source et sont générés selon le même principe, leurs distributions de caractères sont similaires. L'ajout du second texte n'introduit donc pas de nouvelle structure ou régularité, ce qui explique que l'entropie reste pratiquement inchangée, avec seulement une légère variation due à l'ajustement des fréquences observées.*

4. Discutez de l'entropie de la librairie de Babel. /3

*L'entropie de la librairie de Babel, mesurée avec `h-ascii`, est d'environ 4,9 bits/octet pour les extraits considérés et demeure très proche lorsqu'on concatène plusieurs pages. Cela indique que, pour cette source, la distribution des octets observés est relativement stable d'un extrait à l'autre.*

*Pour interpréter cette valeur, il faut tenir compte du fait que la librairie de Babel utilise un alphabet restreint : elle ne mobilise pas forcément tous les 256 octets possibles. Le plafond pertinent est donc  $\log_2(|\text{alphabet}|)$ , et non pas 8 bits/octet. Dans ce cadre, une entropie vers 4,9 bits/octet peut être proche du maximum pour cet alphabet restreint si la distribution y est relativement uniforme.*

*En résumé, les pages de la librairie de Babel se comportent comme des sorties d'une même source dont les statistiques de symboles (octets) sont stables, ce qui se reflète par une entropie par octet voisine entre différents extraits.*

### 1.3 Histogrammes (5 points)

1. Utilisez la librairie de Babel pour trouver un texte avec votre prénom et des mots en anglais aléatoires. Donnez ce texte (lettres majuscules et espaces uniquement, pas de chiffres). Modifiez le texte trouvé si nécessaire.

*CARPENTERS REDISTRIBUTE DISQUISITIONS ORNAMENTED UN-HYSTERICAL CLAIRCOLLES UNHALLOW OLEORESINS FARES BEST-SELLERDOM UNGENTLEMANLIKE FERMENTATIVE RACILY ANTICOAGULANTS REREWARDS SLINKER*

2. Utilisez CyberChef pour chiffrer ce texte avec ROT13. Assurez-vous que le résultat ne contienne que des lettres majuscules et des espaces.

*PNECRAGREF ERQVFGEVOHGR QVFDHVFVGVB AF BEANZRAGRQ HAULFGREVPNY PYNVEPBYYRF HAUNYYBJ BYRBERFVAF SNERF ORFGFRYYREQBZ HATRAGYRZNAYVXR SREZRAGNGVIR ENPVYL NAGVPBNTHYNAGF ERERJNEQF FYVAXRE*

3. Utilisez `h-lettre` sur la version avec ROT13 et sur la version sans ROT13, puis faites deux histogrammes avec un tableau. /1

```

/home kali@kali:~/Desktop/TPI/utilitaireTPI/Source - Entropie - Chiffrement
Session Actions Edit View Help
t: ~/Desktop/TPI/utilitaireTPI/Source - Entropie - Chiffrement
(space)=15 A=14 B=8 C=1 D=1 E=16 F=15 G=12 H=6 I=1 J=2 K=0 L=2
M=0 N=13 O=2 P=6 Q=5 R=23 S=2 T=2 U=2 V=14 W=0 X=2 Y=14 Z=4
Nombre total : 182 Entropie : 4.074212

/home kali@kali:~/Desktop/TPI/utilitaireTPI/Source - Entropie - Chiffrement
t: ~/Desktop/TPI/utilitaireTPI/Source - Entropie - Chiffrement
(space)=15 A=13 B=2 C=6 D=5 E=23 F=2 G=2 H=2 I=14 J=0 K=2 L=14
M=4 N=14 O=8 P=1 Q=1 R=16 S=15 T=12 U=6 V=1 W=2 X=0 Y=2 Z=0
Nombre total : 182 Entropie : 4.074212

```

### Sortie h-lettre (données utilisées).

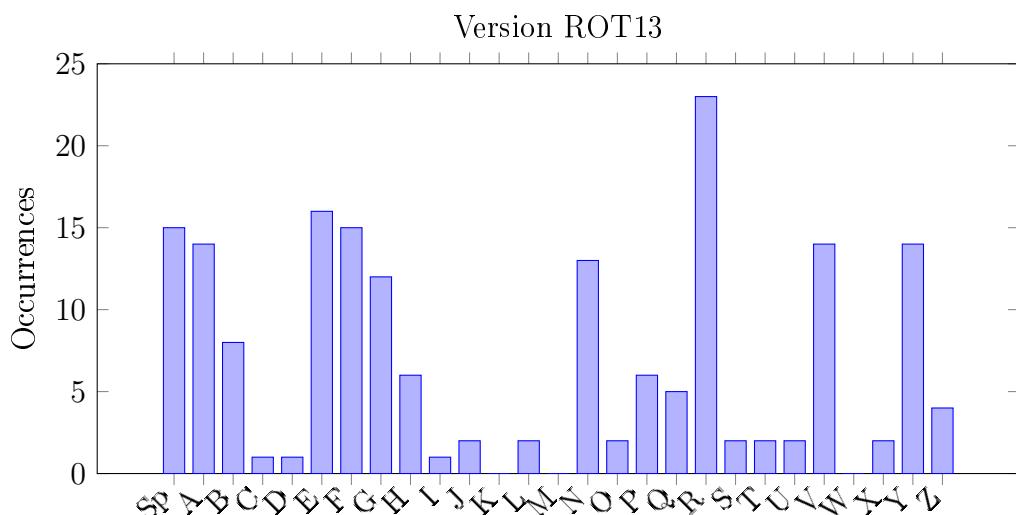
```

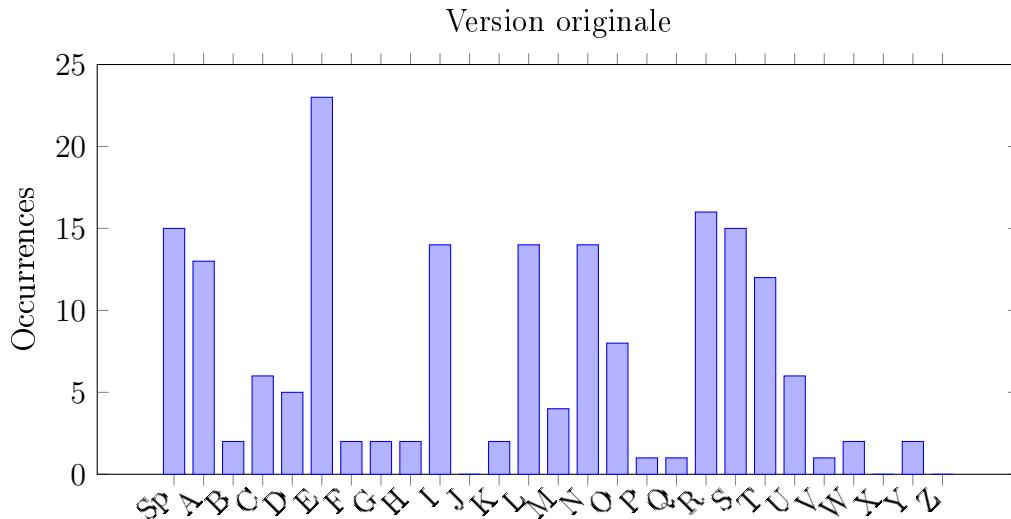
./h-lettre < Cyberchef-ROT13.txt
(spaces)=15 A=14 B=8 C=1 D=1 E=16 F=15 G=12 H=6 I=1 J=2 K=0 L=2
M=0 N=13 O=2 P=6 Q=5 R=23 S=2 T=2 U=2 V=14 W=0 X=2 Y=14 Z=4
Nombre total : 182 Entropie : 4.074212

./h-lettre < Cyberchef-original.txt
(spaces)=15 A=13 B=2 C=6 D=5 E=23 F=2 G=2 H=2 I=14 J=0 K=2 L=14
M=4 N=14 O=8 P=1 Q=1 R=16 S=15 T=12 U=6 V=1 W=2 X=0 Y=2 Z=0
Nombre total : 182 Entropie : 4.074212

```

### Histogrammes (fréquences par caractère).





(Sp = espace ; entropie dans les deux cas : 4,074212.)

4. Utilisez le programme lettre pour générer une séquence de 3200 caractères, puis encodez cette séquence avec ROT13.

*Voir le numéro ci-dessous.*

5. Utilisez h-lettre sur la version avec ROT13 et sur la version sans ROT13, puis faites deux histogrammes avec un tableau. /1

```

/home/kali/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement
$ ./h-lettre < Lettre-original.txt
Nombre total de caractères : 3200
Entropie de l'entrée : 4.253363
A=371 B=42 C=85 D=117 E=334 F=64 G=72 H=171 I=203 J=11
K=29 L=115
M=69 N=203 O=215 P=55 Q=7 R=148 S=173 T=238 U=81 V=32 W=57 X=2 Y=73
Z=2
Nombre total de caractères : 3200
Entropie de l'entrée : 4.253363
/home/kali/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement
$ ./h-lettre < Lettre-ROT13.txt
Nombre total de caractères : 3200
Entropie de l'entrée : 4.253363
/home/kali/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement
$ ./h-lettre < Lettre-rot13.txt
Nombre total de caractères : 3200
Entropie de l'entrée : 4.253363

```

Sortie h-lettre (données utilisées).

```

./h-lettre < Lettre-original.txt
(space)=371 A=231 B=42 C=85 D=117 E=334 F=64 G=72 H=171 I=203 J=11
K=29 L=115
M=69 N=203 O=215 P=55 Q=7 R=148 S=173 T=238 U=81 V=32 W=57 X=2 Y=73
Z=2
Nombre total : 3200 Entropie : 4.253363

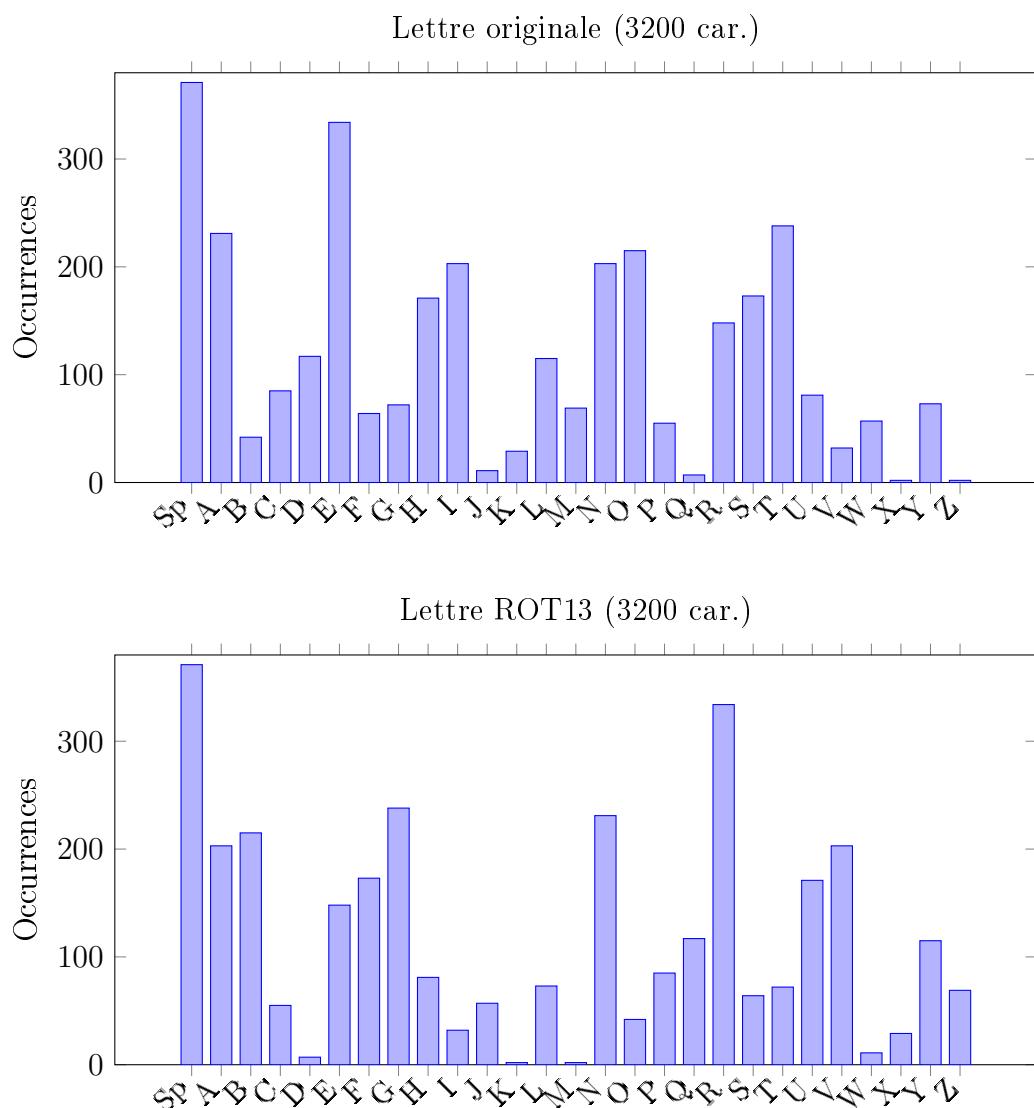
```

```

./h-lettre < Lettre-ROT13.txt
(space)=371 A=203 B=215 C=55 D=7 E=148 F=173 G=238 H=81 I=32 J=57 K
=2 L=73
M=2 N=231 O=42 P=85 Q=117 R=334 S=64 T=72 U=171 V=203 W=11 X=29 Y
=115 Z=69
Nombre total : 3200 Entropie : 4.253363

```

### Histogrammes (fréquences par caractère).



(Sp = espace ; entropie dans les deux cas : 4,253363.)

6. 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 ? /1,5

*En comparant les quatre histogrammes, on observe que le chiffrement ROT13 ne modifie pas les fréquences globales des caractères, mais les associe à d'autres lettres. Les histogrammes avant et après ROT13 ne sont donc pas identiques lettre par lettre : les barres de forte occurrence sont simplement déplacées le long de l'axe des lettres. Ce comportement est caractéristique d'un chiffrement par substitution.*

*On remarque également une différence entre les histogrammes issus de la source lettre et ceux issus du texte de la librairie de Babel. La source lettre, générée selon les fréquences de l'anglais, présente une structure plus marquée avec certaines lettres et l'espace beaucoup plus fréquents. À l'inverse, le texte Babel, surtout lorsqu'il est court, produit des histogrammes plus irréguliers, ce qui reflète une absence de structure linguistique stable.*

*Si les fréquences étaient comptabilisées sur deux lettres à la fois (digrammes), ces différences seraient encore plus visibles. Dans le cas de la source texte (anglais), certains digrammes courants comme (th) ou (ee) auraient des fréquences nettement plus élevées que les autres. En revanche, pour la source lettre, où les lettres sont générées indépendamment les unes des autres, les digrammes seraient beaucoup plus uniformément répartis et des digrammes comme (th) ou (ee) ne seraient pas particulièrement favorisés.*

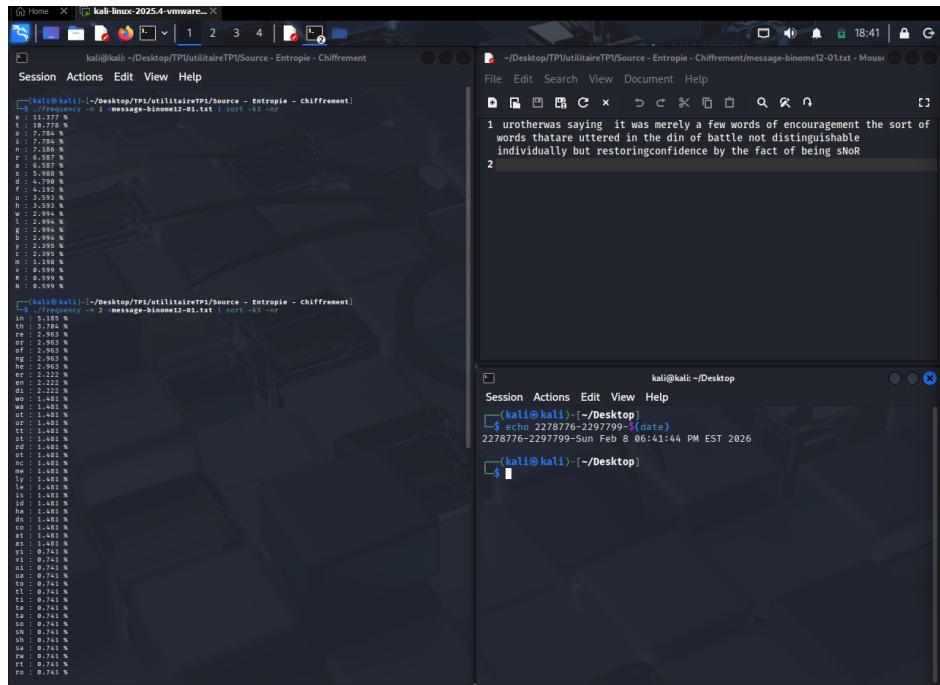
7. Est-ce que comptabiliser les fréquences sur 2 lettres faciliterait le déchiffrement du message dans le cas du texte pris sur Babel ? Qu'en est-il pour celui généré avec lettre ? /1,5

*Dans le cas du texte pris sur la librairie de Babel, comme l'extrait analysé contient de nombreux mots anglais valides, il présente des régularités locales propres à la langue. Le comptage des fréquences sur deux lettres (digrammes) mettrait alors en évidence des paires courantes de l'anglais, telles que th, er, re ou ll, ce qui fournirait des indices supplémentaires utiles pour le déchiffrement d'un chiffrement par substitution.*

*En revanche, pour le texte généré avec la source lettre, les lettres sont produites indépendamment les unes des autres selon une distribution donnée. Il n'existe donc pas de dépendance entre lettres successives, et le comptage des fréquences sur deux lettres n'apporte pas d'information supplémentaire par rapport à l'analyse lettre par lettre. Les digrammes comme th ou ee n'y sont pas favorisés et ne facilitent pas le déchiffrement.*

#### 1.4 Déchiffrement simple (15 points)

1. Donnez une capture d'écran de votre utilisation de l'utilitaire frequency.  
/1



2. Fournir le texte déchiffré, ainsi que votre démarche clairement expliquée.  
/14

### Texte déchiffré :

*urotherwas saying it was merely a few words of encouragement the sort of words that are uttered in the din of battle not distinguishable individually but restoring confidence by the fact of being spoken*

**Démarche.** Le déchiffrement a été effectué par analyse fréquentielle suivie d'une déduction progressive appliquée séquentiellement.

Dans un premier temps, l'utilitaire frequency a été utilisé pour mesurer les fréquences des caractères du message chiffré. Les résultats ont été triés avec sort -k3 -nr afin d'identifier rapidement les symboles dominants. La distribution observée étant compatible avec l'anglais, le symbole le plus fréquent a été associé à l'espace, conformément à l'indication fournie dans l'énoncé. Cette étape a servi de base pour construire une table de substitution initiale.

L'analyse des digrammes et trigrammes a ensuite permis d'identifier des motifs probables. Le trigramme *XJS* a été reconnu comme correspondant au mot très fréquent *the*, ce qui a permis de déduire  $X \rightarrow t$  et  $J \rightarrow h$ . Le digramme *IW* a ensuite été identifié comme *in*, menant à  $I \rightarrow i$  et  $W \rightarrow n$ . Ces substitutions ont immédiatement rendu plusieurs fragments de texte partiellement lisibles, confirmant la validité des hypothèses.

La progression s'est poursuivie par reconnaissance de mots lisibles par contexte. Par exemple, la séquence *UeinO* correspondait clairement à *being*, ce qui a permis d'associer  $U \rightarrow b$  et  $O \rightarrow g$ . Le digramme *bD* a été identifié comme *by*, donnant  $D \rightarrow y$ , et *LT* comme *of*. La lettre *C* a été confirmée par apparition dans des mots cohérents.

À ce stade, plusieurs mots entiers devenaient reconnaissables : encouragement, few, words, battle, individually. Ces reconnaissances ont permis de

*valider les substitutions existantes et d'en proposer de nouvelles, notamment  $G \rightarrow a$ , afin de préserver la cohérence grammaticale globale.*

*Les lettres restantes correspondaient à des symboles de faible fréquence, cohérents avec des lettres rares en anglais. La résolution finale s'est appuyée sur la structure grammaticale de la phrase et sur la plausibilité linguistique globale, confirmant la cohérence du message reconstruit.*

*Le déchiffrement a donc été réalisé par une combinaison d'analyse statistique, d'identification de motifs fréquents, d'application progressive des substitutions et de validation contextuelle jusqu'à obtention d'un texte cohérent.*

## 1.5 Masque jetable (20 points)

- Créez un programme qui génère en binaire un masque jetable (probabilité égale 0/1). Donnez le code, expliquez le fonctionnement. Utilisez le module random de base et une alternative plus sécuritaire ; expliquez en quoi cette alternative est plus sécuritaire. /12

*Le programme ci-dessous génère un masque jetable sous forme binaire, stocké dans un fichier **key.txt**. Le masque est une suite d'octets, de longueur **size** octets, afin de correspondre à la taille du message à chiffrer en octets, condition requise pour un masque jetable (OTP).*

*Chaque octet est tiré uniformément parmi 0..255. Cette uniformité implique que, pour chaque position de bit (parmi les 8 bits d'un octet), la probabilité d'obtenir 0 ou 1 est égale à 1/2, car sur les 256 valeurs possibles, 128 ont ce bit à 0 et 128 l'ont à 1.*

*La version non sécuritaire utilise le module **random** (tirages pseudo-aléatoires déterministes), tandis que la version sécuritaire utilise **secrets**, conçu pour la cryptographie et s'appuyant sur une source d'aléa du système d'exploitation.*

```
from random import randint
import secrets
import sys

def notSecureKey(size: int) -> None:
    # Génère size octets pseudo-aléatoires (non cryptographiques)
    with open("key.txt", "wb") as key_file:
        for _ in range(size):
            k = randint(0, 255)
            key_file.write(bytes([k]))

def secureKey(size: int) -> None:
    # Génère size octets avec une source adaptée à la cryptographie
    with open("key.txt", "wb") as key_file:
        key_file.write(secrets.token_bytes(size))

secureKey(100) if sys.argv[1] == "secure" else notSecureKey(100)
```

**Sécurité des deux méthodes.** Le module `random` produit une suite pseudo-aléatoire déterministe basée sur un état interne. Si cet état est récupéré ou deviné, la suite peut devenir prédictible, ce qui est incompatible avec les exigences de sécurité d'un OTP. Le module `secrets` est conçu pour générer des valeurs non prédictibles pour un attaquant, en utilisant une source d'aléa cryptographiquement robuste fournie par le système d'exploitation [1].

- Utilisez votre programme pour générer des clés pour les 100 premiers caractères du texte de 4.3.1 (sans espaces). D'abord avec `random` de base, puis avec l'alternative sécuritaire.

Le fichier `smallText.txt` est produit par le programme, en retirant les espaces du texte puis en conservant les 100 premiers caractères. Ensuite, on génère une clé de 100 octets et on l'écrit dans `key.txt`.

Commandes utilisées :

```
python3 otp.py random
python3 otp.py secure
```

The screenshot shows a terminal window with several command-line operations:

- Line 1: `./masque key.txt 100 smallText.txt cipher.bin`
- Line 2: `$ python3 otp.py secure`
- Line 3: `./masque key.txt 100 smallText.txt cipher.bin`
- Line 4: `$ echo 2278776-2297799-$(date)` followed by the current date and time: 2278776-2297799-Sun Feb 8 08:03:36 PM EST 2026
- Line 5: `$ ./masque key.txt 100 smallText.txt cipher.bin`
- Line 6: `$ ./noSpace100Char(text)`
- Line 7: `$ with open('key.txt', 'wb') as key_file:`
- Line 8: `for i in range(size):`
- Line 9: `k = randint(0,255)`
- Line 10: `key_file.write(bytes([k]))`
- Line 11: `def secureKey(size):`
- Line 12: `with open('key.txt', 'wb') as key_file:`
- Line 13: `for i in range(size):`
- Line 14: `k = secrets.randbits(8)`
- Line 15: `key_file.write(bytes([k]))`
- Line 16: `def noSpace100Char(text):`
- Line 17: `text = text.replace(" ", "")`
- Line 18: `with open('smallText.txt', 'w') as text_file:`
- Line 19: `text_file.write(text[:100])`
- Line 20: `secureKey(100) if sys.argv[1] == "secure" else notSecureKey(100)`
- Line 21: `noSpace100Char('CARPENTERREDISTRIBUTE DISQUISITIONS ORNAMENTED UNHYSTERICALLY CLATCOLLES UNHALLOW OLEORESINS FARES BESTSELLERDOM UNGENTLEMANLIKE FERMENTATIVE RACILY ANTCOAGULANTS REREWARDS SLINKER')`

- Utilisez l'utilitaire masque pour appliquer vos clés sur le texte. Calculez l'entropie par bit (h-bit) et par octet (h-ascii) avant et après l'application de chaque masque. /3

The screenshot shows two terminal windows side-by-side. Both are running on a Kali Linux desktop environment. The left window is titled 'kali@kali: ~/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement' and the right window is titled 'kali@kali: ~/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement'. Both windows show the same command-line session.

```

Session Actions Edit View Help
Session Actions Edit View Help
(kali㉿kali)-[~/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement]
$ python3 otp.py secure
(kali㉿kali)-[~/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-ascii < smallText.txt
Nombre total d'octets : 100
Entropie de l'entrée : 3.873977
(kali㉿kali)-[~/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-bit < smallText.txt
0 = 475
1 = 325
Nombre total de bits : 800
Entropie du texte entre : 0.974489
(kali㉿kali)-[~/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./masque key.txt 100 smallText.txt cipher.bin
(kali㉿kali)-[~/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-ascii < cipher.bin
Nombre total d'octets : 100
Entropie de l'entrée : 6.241210
(kali㉿kali)-[~/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-bit < cipher.bin
0 = 380
1 = 415
Nombre total de bits : 800
Entropie du texte entre : 0.999982
(kali㉿kali)-[~/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement]
$ echo 2278776-2297799-$(date)
2278776-2297799-Sun Feb 8 07:59:09 PM EST 2026
(kali㉿kali)-[~/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement]
$ ./h-bit < cipher.bin
0 = 380
1 = 415
Nombre total de bits : 800
Entropie du texte entre : 0.998958
(kali㉿kali)-[~/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement]
$ echo 2278776-2297799-$(date)
2278776-2297799-Sun Feb 8 07:59:04 PM EST 2026
(kali㉿kali)-[~/Desktop/TP1/utilitaireTP1/Source - Entropie - Chiffrement]
$ 

```

4. Discutez des résultats. Dites à quoi vous vous attendiez, si les résultats sont conformes ou non, et pourquoi. /5

*Avant l'application du masque jetable, l'entropie est limitée par la structure du texte : l'alphabet est restreint (lettres majuscules) et les fréquences ne sont pas uniformes, ce qui explique une entropie par octet d'environ 3,87 et une entropie par bit d'environ 0,97.*

*Après application du masque jetable, l'entropie par bit devient très proche de 1 dans les deux cas, ce qui est conforme à l'attendu : le XOR avec une clé aléatoire de même longueur rend les bits de sortie proches d'une distribution équilibrée 0/1.*

*L'entropie par octet du texte chiffré se situe autour de 6,2–6,3 bits/octet pour un fichier de 100 octets. Cette valeur est cohérente avec la taille de l'échantillon : avec seulement 100 octets observés, l'entropie empirique par octet ne peut pas atteindre 8, et elle est bornée par  $\log_2(100) \approx 6,64$  bits/octet. Avec davantage d'octets chiffrés, l'estimation se rapprocherait de 8 bits/octet.*

*Les résultats random et secure sont proches du point de vue de l'entropie mesurée sur un petit échantillon. La différence principale entre les deux méthodes est la prédictibilité : random n'est pas conçu pour des usages cryptographiques, alors que secrets vise l'imprévisibilité face à un attaquant, ce qui est requis pour la sécurité effective d'un OTP.*

## 1.6 Communication à clé publique, HTTPS et SSL (5 points)

1. Expliquez la différence entre HTTP et HTTPS dans vos mots. /0,5

*HTTP et HTTPS sont deux protocoles utilisés pour la communication entre un navigateur web et un serveur. La différence principale entre les deux réside dans la sécurité des échanges.*

*HTTP transmet les données en clair sur le réseau. Cela signifie que les informations échangées, comme les identifiants, mots de passe ou données personnelles, peuvent être interceptées et lues par un attaquant qui aurait accès au trafic réseau.*

*HTTPS est la version sécurisée de HTTP. Il utilise un chiffrement basé sur TLS afin de protéger les communications. Les données échangées sont chiffrées, ce qui empêche un tiers de les lire ou de les modifier. De plus, HTTPS permet au navigateur de vérifier l'identité du serveur grâce à un certificat numérique, ce qui réduit les risques d'usurpation de site [2].*

2. Expliquez pourquoi il est impossible de se connecter au dossier étudiant avec http et quelle solution sécuritaire pourrait être mise en place. /1

*Il est impossible de se connecter au dossier étudiant en utilisant le protocole HTTP, car ce service exige une communication sécurisée. Le dossier étudiant traite des informations sensibles, comme des données personnelles et académiques, qui ne doivent pas transiter en clair sur le réseau. Pour cette raison, le serveur refuse les connexions non chiffrées afin d'éviter les risques d'interception ou de modification des données par un attaquant.*

*Une solution sécuritaire consiste à forcer l'utilisation de HTTPS en redirigeant automatiquement toute requête HTTP vers la version HTTPS du site. Ainsi, même si un utilisateur consulte le lien en HTTP, la connexion est immédiatement établie de manière chiffrée, garantissant la confidentialité et l'intégrité des échanges.*

3. Quelle est l'utilité du header « Strict-Transport-Security » ? /1

*Le header Strict-Transport-Security a pour rôle de forcer le navigateur à utiliser uniquement des connexions HTTPS pour un site donné pendant une durée déterminée. Lorsqu'un navigateur reçoit ce header, il mémorise que le site doit toujours être contacté via HTTPS, même si l'utilisateur tente d'y accéder en utilisant HTTP.*

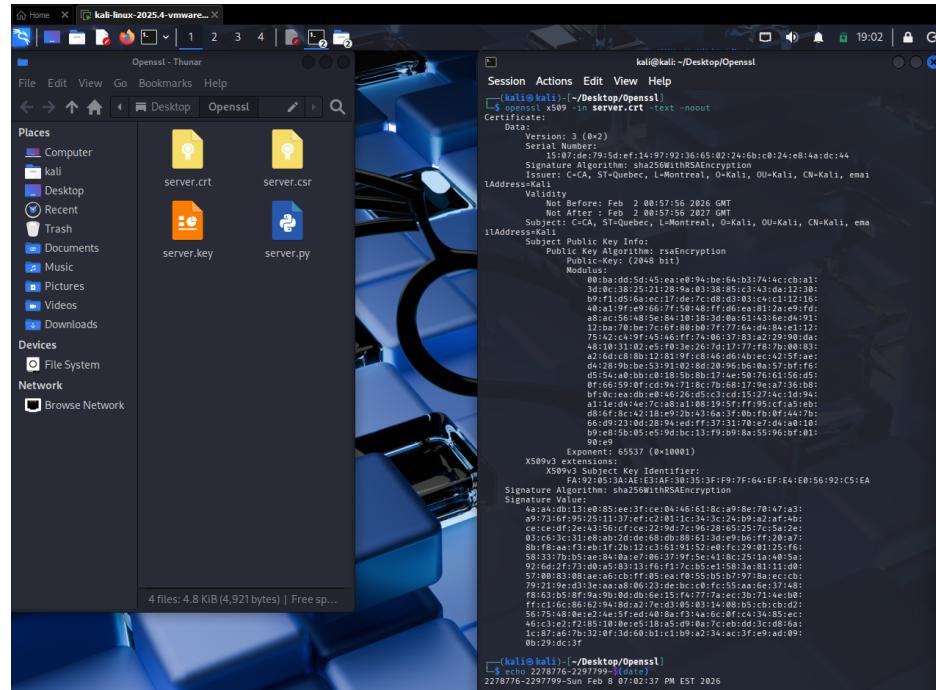
*Ce mécanisme empêche les attaques de type « downgrade » ou « man-in-the-middle » où un attaquant tenterait de rediriger l'utilisateur vers une version non sécurisée du site. En imposant systématiquement le chiffrement des communications, Strict-Transport-Security renforce la sécurité globale des échanges entre le navigateur et le serveur [3].*

4. À quoi sert un certificat à clé publique ? Comment votre navigateur vérifie-t-il l'identité du propriétaire du site ? /0,5

*Un certificat à clé publique, aussi appelé certificat TLS (souvent désigné comme certificat SSL), sert à associer une clé publique à l'identité d'un site afin de permettre des communications sécurisées. Il est utilisé pour authentifier le serveur et établir une connexion chiffrée entre le navigateur et le site web.*

*Le navigateur vérifie l'identité du propriétaire du site lors de la négociation TLS en validant le certificat présenté par le serveur. Il s'assure que le certificat est signé par une autorité de certification de confiance, que le nom de domaine correspond au site visité et que le certificat est valide. Si ces vérifications sont satisfaites, le navigateur fait confiance à l'identité du site [4].*

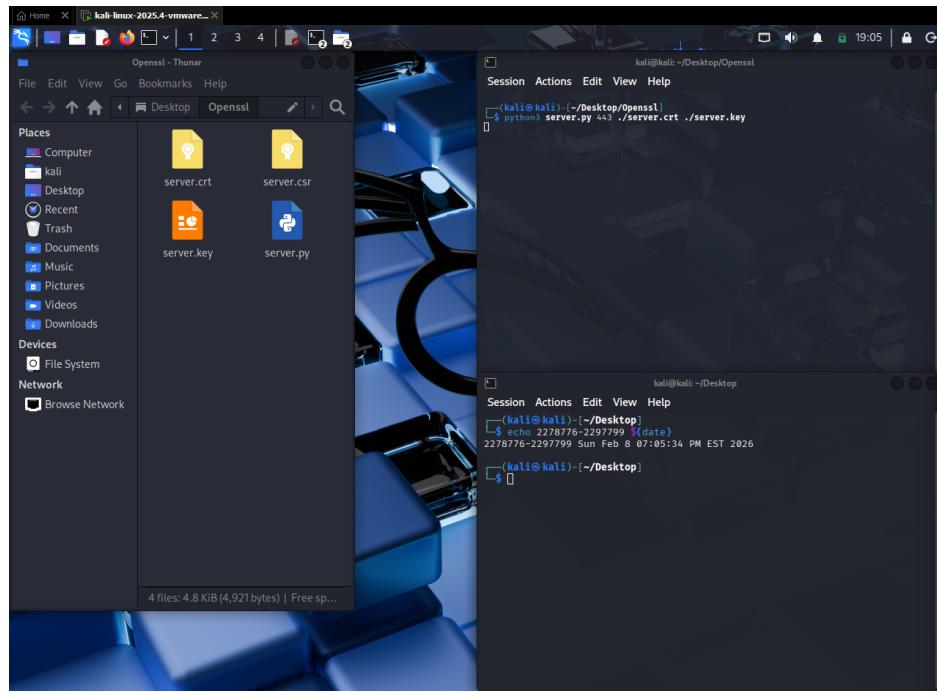
5. Utilisez openssl pour générer un certificat « Self-Signed ». Donnez dans un tableau les champs de votre certificat. /0,5



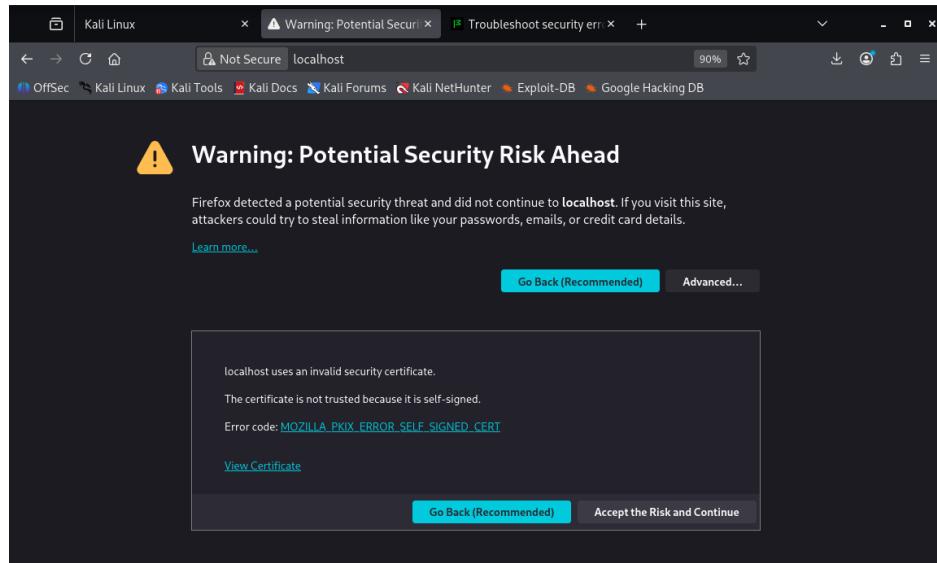
Champs du certificat (sortie `openssl x509 -text`) [5] :

| Champ                  | Valeur   |
|------------------------|--|
| Version                | 3 (0x2)  |
| Serial Number          | 15 :07 :de :5d :ef :14 :97 :92 :36 :65 :02 :24 :6b :c0 :24 :e8 :4a :dc :44     |
| Signature Algorithm    | sha256WithRSAEncryption  |
| Issuer                 | C=CA, ST=Quebec, L=Montreal, O=Kali, OU=Kali, CN=Kali, emailAddress=Kali       |
| Not Before             | Feb 2 00 :57 :56 2026 GMT  |
| Not After              | Feb 2 00 :57 :56 2027 GMT  |
| Subject                | C=CA, ST=Quebec, L=Montreal, O=Kali, OU=Kali, CN=Kali, emailAddress=Kali       |
| Public Key Algorithm   | rsaEncryption  |
| Public-Key Exponent    | 2048 bit<br>65537 (0x10001)  |
| Subject Key Identifier | FA :92 :05 :3A :AE :E3 :AF :30 :35 :3F :F9 :7F :64 :EF :E4 :E0 :56 :92 :C5 :EA |

6. Lancez un serveur python local qui utilise votre certificat (script fourni). Fournissez la clé et le certificat en format .pem.



7. Tentez d'accéder au serveur avec Firefox. Quel problème rencontrez-vous et comment le régler ? Expliquez votre démarche. /1,5



*Lors de l'accès au serveur HTTPS dans Firefox, le navigateur bloque la connexion et affiche un avertissement de sécurité indiquant que le certificat est invalide ou non approuvé. Le problème vient du fait que le certificat utilisé par le serveur est auto-signé. Il n'est donc pas signé par une autorité de certification reconnue par Firefox, ce qui empêche le navigateur de valider l'identité du serveur et d'établir une relation de confiance. Un code d'erreur du type **MOZILLA\_PKIX\_ERROR\_SELF\_SIGNED\_CERT** est affiché.*

*Pour régler ce problème dans le cadre du laboratoire, la démarche consiste à ajouter une exception de sécurité dans Firefox. En accédant à l'URL du serveur, on clique sur « Avancé » dans la page d'avertissement, puis on choisit l'option « Accepter le risque et poursuivre ». Firefox enregistre alors une exception locale pour ce site et permet l'accès au serveur via HTTPS. Cela confirme que le chiffrement TLS fonctionne, mais que le certificat n'était pas approuvé par défaut.*

*Une solution plus « propre » et permanente consisterait à utiliser un certificat signé par une autorité de certification de confiance, ou à créer une autorité de certification locale, l'ajouter comme autorité de confiance dans Firefox, puis signer le certificat du serveur avec cette autorité. Dans tous les cas, le problème initial observé est l'absence de chaîne de confiance reconnue pour un certificat auto-signé.*

## 1.7 Codage (9 points)

1. Expliciter les alphabets  $\sigma$ ,  $\tau$  et  $\tau'$  (sortie de la source, du codeur, du bloc de chiffrement). /3

$\sigma$  est l'alphabet de sortie de la source, c'est-à-dire l'ensemble des symboles possibles d'un NIP. Il contient les dix chiffres et les vingt-six lettres majuscules :  $\sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, \dots, Z\}$ . Un NIP est une suite de 8 symboles appartenant à  $\sigma$ .

$\tau$  est l'alphabet de sortie du codeur, avant chiffrement. Chaque symbole de  $\sigma$  est encodé en ASCII sur un octet, puis la suite de 8 octets est dupliquée.  $\tau$  correspond donc à l'ensemble des valeurs ASCII des chiffres et des lettres majuscules :  $\tau = \{0x30, \dots, 0x39, 0x41, \dots, 0x5A\}$ . La sortie du codeur est une suite de 16 octets sur  $\tau$ .

$\tau'$  est l'alphabet du bloc de chiffrement AES-128. AES-128 manipule des blocs de taille fixe de 128 bits.  $\tau'$  est donc l'ensemble de tous les blocs possibles de 128 bits, soit l'ensemble de toutes les suites possibles de 16 octets.

2. Identifier les langages provenant de ces alphabets. /3

**Langage sur  $\sigma$ .**

Le langage associé à l'alphabet  $\sigma$  est l'ensemble de tous les NIP possibles. Il s'agit de toutes les suites de longueur 8 composées de symboles de  $\sigma$ . On peut l'écrire :  $L(\sigma) = \sigma^8$ .

**Langage sur  $\tau$ .**

Le langage associé à l'alphabet  $\tau$  est l'ensemble des sorties possibles du codeur. Chaque mot est formé de 16 octets, correspondant à la répétition deux fois de l'encodage ASCII d'un NIP valide.

Le langage est donc l'ensemble des mots de longueur 16 sur  $\tau$  ayant la structure  $X\|X$ , où  $X$  est une suite de 8 symboles de  $\tau$  correspondant à un NIP valide.

**Langage sur  $\tau'$ .**

Le langage associé à l'alphabet  $\tau'$  est l'ensemble des blocs pouvant être traités par AES-128. Il contient toutes les suites possibles de 128 bits, sans contrainte particulière de structure.

On peut l'écrire :  $L(\tau') = (\tau')^1$ , c'est-à-dire l'ensemble de tous les blocs de 128 bits.

3. Identifiez les attaques auxquelles le système est vulnérable. /3

Premièrement, le système est vulnérable à une attaque par rejeu. Un attaquant peut intercepter un message chiffré correspondant à un changement

*de NIP et le réinjecter plus tard sur le réseau. Comme le message chiffré est toujours identique pour un même NIP et qu'aucune information temporelle ou aléatoire n'est incluse, la banque ne peut pas distinguer un message légitime d'un ancien message rejoué.*

*Deuxièmement, le système est vulnérable à une attaque par substitution. Un attaquant peut remplacer, sur le réseau, le message chiffré d'un client par un autre message chiffré valide qu'il a préalablement capturé. Étant donné que l'identité du client est supposée transmise de façon indépendante, le serveur peut associer à tort le NIP contenu dans le message substitué au mauvais compte.*

*Troisièmement, le système est vulnérable à une attaque par reconnaissance de motifs. Le codage étant déterministe, deux NIP identiques produisent exactement le même bloc clair, et donc le même bloc chiffré. Un attaquant observant le trafic peut détecter qu'un même NIP est réutilisé ou qu'un client revient à un ancien NIP, même sans connaître sa valeur.*

*Enfin, le système est vulnérable à une attaque par modification de message. Un attaquant peut altérer arbitrairement des bits du message chiffré. En l'absence de mécanisme d'authentification ou d'intégrité, le déchiffrement produira un NIP incohérent sans que la banque puisse détecter que le message a été modifié, ce qui peut mener à un déni de service ou à un état incohérent du système.*

## 1.8 Changement de codage (5 points)

- Pour chacun des trois codages (figures 1, 2, 3), dites quelles attaques du 4.7.3 ils permettent de bloquer. /3

### Codage 1 (nouveau NIP + parité + nombre aléatoire).

*Ce codage permet de bloquer l'attaque par reconnaissance de motifs. L'ajout d'un nombre aléatoire fait que deux NIP identiques ne produisent plus le même bloc clair, et donc plus le même bloc chiffré.*

*En revanche, il ne bloque pas l'attaque par rejet, car un message chiffré ancien peut toujours être rejoué tel quel. Il ne bloque pas non plus l'attaque par substitution, puisqu'un message chiffré valide capturé peut être remplacé par un autre sur le réseau. Enfin, il ne protège pas contre l'attaque par modification de message, en l'absence de mécanisme d'intégrité.*

### Codage 2 (nouveau NIP + parité + nombre aléatoire + horodatage).

*Ce codage permet de bloquer l'attaque par reconnaissance de motifs, grâce à la présence d'aléa. Il permet également de bloquer l'attaque par rejet de messages anciens, puisque l'horodatage permet à la banque d'invalider les messages trop vieux.*

*En revanche, il ne bloque pas complètement l'attaque par substitution, car un attaquant peut encore remplacer un message par un autre message récent dont l'horodatage est valide. Comme les autres, il ne protège pas contre l'attaque par modification de message, faute de mécanisme d'authentification ou d'intégrité.*

### Codage 3 (nouveau NIP + ancien NIP + horodatage).

Ce codage permet de bloquer l'attaque par rejet, grâce à l'horodatage. Il permet aussi de bloquer l'attaque par substitution, car un message n'est accepté que si l'ancien NIP correspond à celui attendu par la banque pour le compte concerné, ce qu'un attaquant ne peut pas produire sans connaître cet ancien NIP.

La reconnaissance de motifs est fortement limitée par la présence du timestamp, qui empêche la répétition exacte des blocs. En revanche, comme pour les autres codages, l'attaque par modification de message reste possible, puisqu'aucun mécanisme d'intégrité n'est ajouté.

2. Selon vous, quel est le meilleur codage ? Pourquoi ? /2

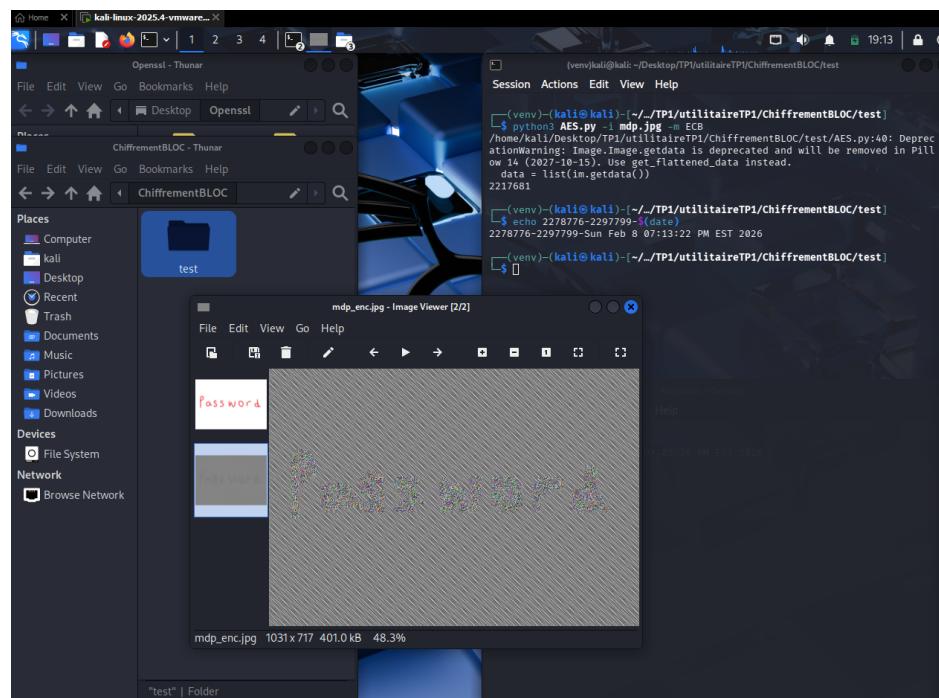
*Le meilleur codage est le codage 3.*

En effet, parmi les trois propositions, c'est le seul codage qui permet de bloquer l'attaque la plus critique dans ce scénario, à savoir l'attaque par substitution. En incluant l'ancien NIP dans le message, le changement de NIP est lié à l'état courant du compte, ce qui empêche un attaquant de réutiliser ou de substituer un message chiffré capturé pour un autre client sans connaître l'ancien NIP correspondant. De plus, la présence d'un horodatage permet de bloquer les attaques par rejet de messages anciens.

Les codages 1 et 2 améliorent principalement la confidentialité face à l'observation du trafic en empêchant la reconnaissance de motifs, mais ils ne protègent pas contre la substitution de messages, qui est une attaque active plus dangereuse dans le modèle de menace considéré.

## 1.9 Chiffrement par bloc et modes d'opération (4 points)

1. Chiffrez le fichier mdp.jpg en mode ECB avec le script AES.py. Observez le fichier de sortie et commentez. /1



Le fichier mdp.jpg, contenant un mot de passe enregistré sous forme d'image, a été chiffré à l'aide du script AES.py en mode ECB. L'exécution de la

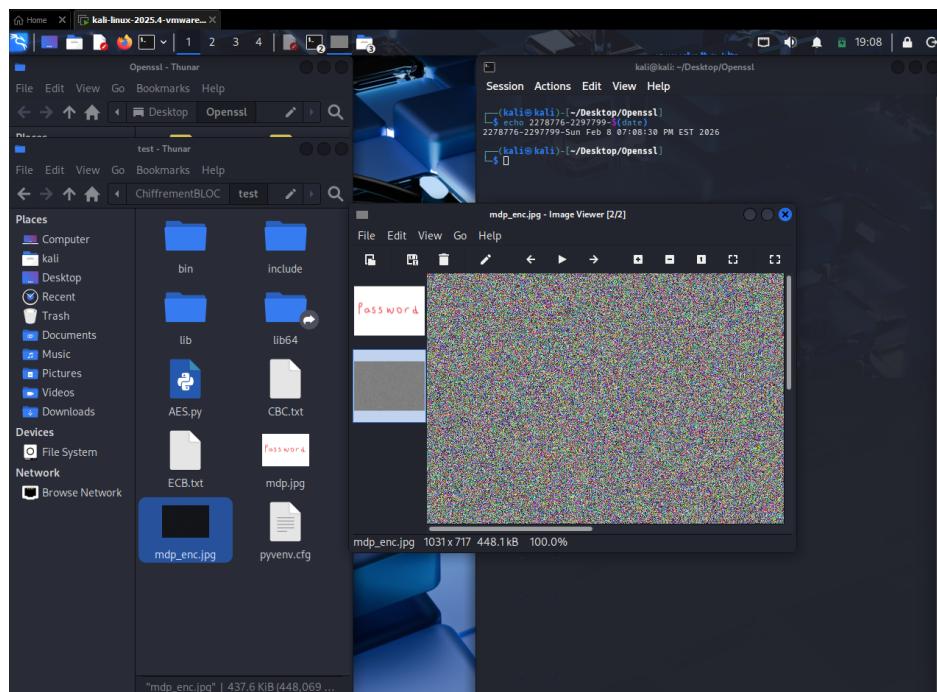
commande `python3 AES.py -i mdp.jpg -m ECB` produit un fichier image chiffré (`mdp_enc.jpg`).

À l'observation du fichier de sortie, on constate que l'image chiffrée conserve des structures visuelles reconnaissables de l'image originale, en particulier la forme et la position des lettres du mot « Password », bien que les couleurs soient altérées et semblent aléatoires. Cela s'explique par le fonctionnement du mode ECB : chaque bloc de 128 bits est chiffré indépendamment avec la même clé. Ainsi, deux blocs identiques en entrée produisent deux blocs identiques en sortie.

Dans une image, de nombreuses zones contiennent des pixels identiques ou très similaires. Le mode ECB préserve donc les motifs et les répétitions présentes dans les données, ce qui entraîne une fuite d'information visuelle malgré l'utilisation d'un algorithme de chiffrement considéré comme sécurisé (AES).

Cette expérience montre que, même si AES est cryptographiquement robuste, le mode ECB est inadapté pour chiffrer des données structurées comme des images, car il ne garantit pas la confidentialité des motifs et révèle des informations sur le contenu original.

2. Chiffrez maintenant le fichier en mode CBC. Observez le fichier puis commentez. /1



Le fichier `mdp.jpg` a été chiffré à l'aide du script `AES.py` en mode CBC. Le fichier de sortie obtenu apparaît comme un bruit aléatoire uniforme, sans aucune structure visuelle reconnaissable provenant de l'image originale. Contrairement au mode ECB, aucune forme, aucun motif ni aucune information sur le contenu initial ne peut être distinguée.

Cela s'explique par le fonctionnement du mode CBC : chaque bloc de données est combiné (par XOR) avec le bloc chiffré précédent avant d'être chiffré. De plus, le premier bloc dépend d'un vecteur d'initialisation (IV).

*Ainsi, deux blocs identiques en entrée ne produisent pas le même bloc chiffré en sortie, même s'ils apparaissent au même endroit dans le fichier.*

*Le mode CBC casse donc les répétitions présentes dans les données et empêche la reconnaissance de motifs, ce qui assure une bien meilleure confidentialité pour des données structurées comme des images. Cette expérience montre que, bien que l'algorithme AES soit identique, le choix du mode d'opération est déterminant pour la sécurité : le mode CBC est nettement plus adapté que le mode ECB pour le chiffrement de fichiers contenant des structures répétitives.*

3. Expliquez la différence entre le mode ECB et le mode CBC. Concluez sur l'importance des modes d'opération. /2

*Le mode ECB chiffre chaque bloc de données indépendamment des autres avec la même clé. Ainsi, deux blocs identiques en entrée produisent toujours deux blocs chiffrés identiques en sortie. Ce mode ne masque donc pas les structures ni les répétitions présentes dans les données, ce qui peut entraîner des fuites d'information, comme l'apparition de motifs visibles lors du chiffrement d'images.*

*Le mode CBC, en revanche, introduit une dépendance entre les blocs. Chaque bloc en clair est combiné avec le bloc chiffré précédent avant d'être chiffré, et le premier bloc dépend d'un vecteur d'initialisation. Cela empêche des blocs identiques de produire des blocs chiffrés identiques et supprime les motifs observables dans les données.*

*En conclusion, les modes d'opération sont essentiels dans le chiffrement par bloc, car ils déterminent comment l'algorithme est appliqué aux données réelles. Un algorithme robuste comme AES peut devenir vulnérable s'il est utilisé avec un mode inadapté, tandis qu'un mode approprié, comme CBC, améliore significativement la confidentialité des données.*

## 1.10 Organisation des mots de passe en UNIX/Linux (12 points)

1. Examinez le fichier /etc/shadow. Contient-il des mots de passe ? Pourquoi ? Quelles sont ses permissions ? Pourquoi ? /1

*Le fichier /etc/shadow ne contient pas les mots de passe des utilisateurs en clair. Il contient les hachés cryptographiques des mots de passe ainsi que des informations associées à leur gestion, comme la date du dernier changement et les politiques d'expiration. Cette organisation permet de vérifier un mot de passe sans jamais avoir besoin de stocker ou de manipuler sa valeur en clair, ce qui limite l'impact d'une compromission.*

*Sur une installation Kali Linux, les permissions du fichier /etc/shadow sont restrictives. Le fichier appartient à l'utilisateur root et n'est lisible et modifiable que par celui-ci (et éventuellement par le groupe shadow). Les utilisateurs ordinaires n'ont aucun droit d'accès à ce fichier.*

*Ces permissions sont nécessaires car le fichier contient des informations sensibles. Même si les mots de passe sont stockés sous forme de hachés, un accès non autorisé permettrait de lancer des attaques hors ligne, comme*

*des attaques par dictionnaire ou par force brute. La restriction d'accès protège donc le mécanisme d'authentification et réduit la surface d'attaque du système [6].*

2. Examinez le fichier /etc/passwd. Contient-il des mots de passe ? Pourquoi ? Quelles sont ses permissions ? Pourquoi ? /1

*Le fichier /etc/passwd ne contient pas les mots de passe des utilisateurs. Les champs de mot de passe ont été déplacés vers /etc/shadow afin d'éviter que les mots de passe (ou leurs hachés) soient accessibles aux utilisateurs ordinaires [7].*

*Sur une installation Kali Linux, le fichier /etc/passwd est lisible par tous les utilisateurs et modifiable uniquement par root. Ces permissions sont nécessaires car ce fichier contient des informations publiques sur les comptes (nom d'utilisateur, identifiant, groupe, répertoire personnel, shell) dont de nombreux programmes ont besoin pour fonctionner correctement, tout en empêchant toute modification non autorisée.*

3. Ajoutez un utilisateur avec useradd. Observez passwd et shadow. Lequel ou lesquels sont modifiés ? Pourquoi ? /2

*Lors de l'ajout d'un utilisateur avec la commande **useradd -g users -s /bin/bash -m NOM**, les fichiers /etc/passwd et /etc/shadow sont tous les deux modifiés.*

*Le fichier /etc/passwd est modifié pour y ajouter une nouvelle ligne correspondant au nouvel utilisateur. Cette ligne contient les informations publiques du compte, comme le nom d'utilisateur, l'identifiant (UID), le groupe principal (GID), le répertoire personnel et le shell associé.*

*Le fichier /etc/shadow est également modifié afin d'y ajouter une entrée pour le nouvel utilisateur. Cette entrée contient le champ du mot de passe sous forme de hash (ou un champ vide/verrouillé tant qu'aucun mot de passe n'est défini), ainsi que les informations liées à la gestion et à l'expiration du mot de passe.*

*Les deux fichiers sont donc modifiés car ils ont des rôles complémentaires : /etc/passwd décrit les comptes utilisateurs visibles du système, tandis que /etc/shadow stocke les informations sensibles liées à l'authentification.*

4. Donnez un mot de passe avec passwd. Qu'est-ce que vous remarquez dans passwd et shadow ? /1

*Après avoir défini un mot de passe pour l'utilisateur avec la commande **sudo passwd USERNAME**, on observe que seul le fichier /etc/shadow est modifié.*

*Dans /etc/shadow, le champ correspondant à l'utilisateur, qui était auparavant vide ou verrouillé, est remplacé par un haché du mot de passe ainsi que par les informations de date liées au changement de mot de passe. Le fichier /etc/passwd, quant à lui, reste inchangé, car il ne contient pas les informations sensibles d'authentification.*

*Cela montre que la gestion des mots de passe est entièrement séparée des informations publiques des comptes utilisateurs.*

5. Changez à nouveau le mot de passe avec le même mot de passe. Est-ce que les informations ont changé ? Pourquoi ? /2

*Oui, les informations du mot de passe ont changé.*

*Selon la page de manuel **shadow(5)**, lors d'un changement de mot de passe, le champ encrypted password du fichier /etc/shadow est réécrit et le champ date of last password change est mis à jour. Ainsi, même si le mot de passe en clair est identique, les informations stockées dans /etc/shadow sont différentes après chaque modification.*

6. Créez un deuxième utilisateur, copiez dans shadow le champ mot de passe du premier. Connectez-vous au deuxième avec le mot de passe du premier. Est-ce possible ? Expliquez. Quel est le problème ? /2

*Oui, il est possible de se connecter au deuxième utilisateur en utilisant le mot de passe du premier utilisateur.*

*En copiant la valeur du champ encrypted password du premier utilisateur dans le fichier /etc/shadow du deuxième utilisateur, on copie en réalité le haché du mot de passe (tel qu'interprété par **crypt(3)**). Lors de l'authentification, le système ne vérifie pas l'identité de l'utilisateur associée au haché : il compare simplement le haché stocké avec le haché du mot de passe saisi. Comme les deux comptes possèdent désormais exactement le même haché, le même mot de passe est accepté pour les deux utilisateurs.*

*Le problème mis en évidence est que l'authentification repose uniquement sur la valeur du haché stockée dans /etc/shadow. Si un attaquant obtient un accès en écriture à ce fichier, il peut cloner un mot de passe entre plusieurs comptes sans jamais connaître le mot de passe en clair. Cela montre pourquoi /etc/shadow doit être strictement protégé et modifiable uniquement par root.*

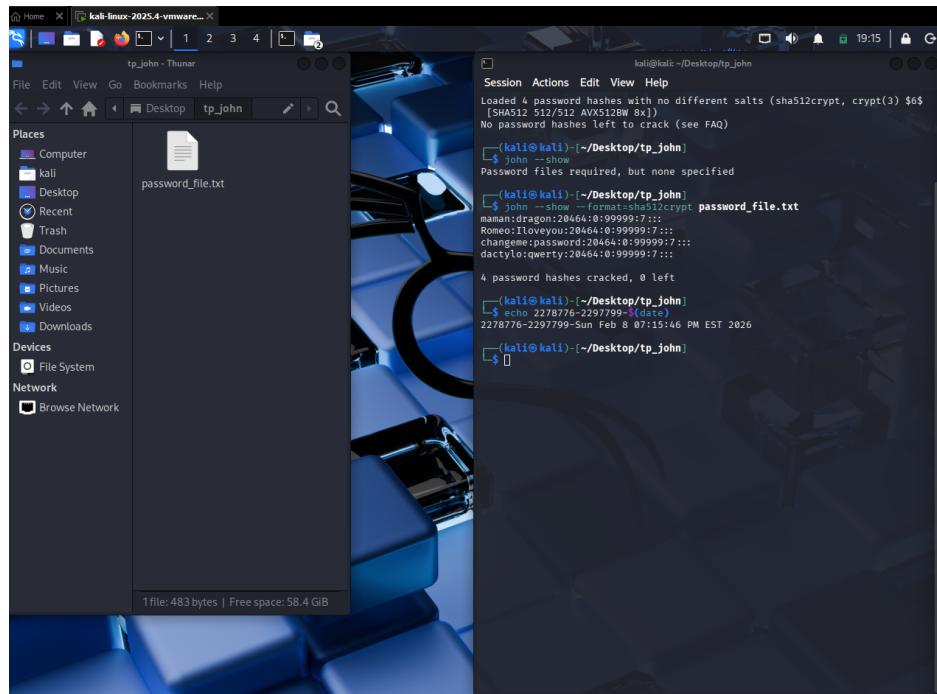
7. Dites comment il est possible de déchiffrer un mot de passe hashé. /3

*Il est possible de retrouver un mot de passe à partir de son hash en faisant des essais et en comparant les résultats. Concrètement, on récupère dans /etc/shadow la valeur du champ encrypted password et on teste des mots de passe candidats. Selon **shadow(5)**, ce champ est une chaîne « encrypted password » dont l'interprétation est définie par **crypt(3)**. On calcule alors, pour chaque candidat, la valeur produite par **crypt(3)** (en utilisant l'algorithme et le sel encodés dans la chaîne stockée), puis on compare avec la valeur présente dans /etc/shadow. Quand les deux coïncident, le mot de passe en clair est retrouvé.*

*Les candidats peuvent être générés par une attaque par dictionnaire ou par force brute. Selon **crypt(3)**, l'usage d'un sel fait que le résultat dépend de ce sel, ce qui force l'attaquant à recalculer les essais plutôt que de réutiliser des tables pré-calculées de façon générale.*

## 1.11 Choix des mots de passe (5 points + 5 bonus)

1. Utilisez John The Ripper avec le dictionnaire rockyou.txt pour identifier le mot de passe correspondant à chaque utilisateur du fichier fourni. /2



*Mots de passe identifiés : maman – dragon ; Romeo – Iloveyou ; changeme – password ; dactylo – qwerty.*

- Pourquoi est-il conseillé de ne pas utiliser le même mot de passe partout ? Donnez un exemple de situation réelle. /3

*Il est déconseillé de réutiliser le même mot de passe, parce qu'une fuite sur un service permet ensuite de compromettre d'autres comptes via réutilisation d'identifiants. Un attaquant prend une paire email/mot de passe trouvée dans une base compromise et l'essaie automatiquement sur d'autres services (courriel, réseaux sociaux, plateformes de jeux, banque, etc.). Exemple réel typique : une fuite sur un site peu important (forum, petit service) donne accès au compte courriel si le mot de passe est le même ; et une fois le courriel compromis, l'attaquant peut réinitialiser des mots de passe d'autres services.*

- BONUS /+5** Faites un lien entre votre réponse en 4.10.6 et votre réponse en 4.11.2.

*Dans la question 4.10.6, on a montré qu'en copiant la valeur du champ encrypted password dans /etc/shadow d'un premier utilisateur vers un deuxième utilisateur, le deuxième compte accepte alors le même mot de passe. Autrement dit, deux comptes deviennent équivalents du point de vue de l'authentification dès qu'ils partagent la même « preuve » d'authentification (la valeur stockée dans le champ de mot de passe).*

*Le lien avec 4.11.2 est que la réutilisation du même mot de passe sur plusieurs services reproduit exactement ce problème, mais à l'échelle d'Internet. Si un mot de passe est réutilisé, la compromission d'un seul service (fuite de hashes puis craquage via dictionnaire, ou fuite directe) permet à un attaquant de se connecter à d'autres comptes avec les mêmes identifiants. C'est le même principe que 4.10.6 : une seule information d'authentification « copiée » ou récupérée devient valable pour plusieurs comptes, ce qui transforme une compromission locale en compromission en chaîne.*

## Bibliographie

Indiquez toutes vos sources d'information (humaines ou documentaires).

## Références

- [1] `secrets` – Generate secure random numbers for managing secrets. Python 3 Documentation. <https://docs.python.org/3/library/secrets.html>. Consulté le 30 janvier 2026.
- [2] HTTP vs HTTPS – Différence entre les protocoles de transfert. AWS. <https://aws.amazon.com/fr/compare/the-difference-between-https-and-http/>. Consulté le 30 janvier 2026.
- [3] Strict-Transport-Security header. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Strict-Transport-Security>. Consulté le 30 janvier 2026.
- [4] Qu'est-ce qu'un certificat SSL ? Cloudflare Learning. <https://www.cloudflare.com/fr-fr/learning/ssl/what-is-an-ssl-certificate/>. Consulté le 30 janvier 2026.
- [5] OpenSSL Self-Signed Certificate. Baeldung. <https://www.baeldung.com/openssl-self-signed-cert>. Consulté le 30 janvier 2026.
- [6] Understanding /etc/shadow file format. Cybergit.biz. <https://www.cybergit.biz/faq/understanding-etcshadow-file/>. Consulté le 30 janvier 2026.
- [7] What is the difference between /etc/shadow and /etc/passwd ? Unix & Linux Stack Exchange. <https://unix.stackexchange.com/questions/461022/what-is-the-difference-between-etc-shadow-and-etc-passwd>. Consulté le 30 janvier 2026.