Polytechnique Montréal

INF8085

Cybersécurité

Travail pratique 1

Automne 2025

Table des matières

1. Sujets

Les sujets de ce premier travail pratique sont :

— Entropie, théorie de l'information, cryptographie

— Communication sécurisée, SSL, HTTPS

— Sécurité des mots de passe et environnement Linux

— Programmation et utilisation d'outil d'analyse, de force brute, de chiffrement et de déchiffrement

2. Directives

Le travail devra être remis le 5 octobre avant 23h59. À moins que cela ne soit explicitement demandé dans le sujet, vous ne devez remettre qu'un fichier PDF nommé selon le format TPX-matricule1-matricule2.pdf. Vous pouvez inclure des annexes dans votre rapport si vous jugez que cela améliore la lisibilité
(code source, ...)

— Le travail devra être fait par équipe de deux. Toute exception (travail individuel, équipe de trois) devra être approuvée au préalable par le professeur.

— L'orthographe et la forme seront prises en compte pour chaque question.

— Indiquez toutes vos sources d'information, qu'elles soient humaines ou documentaires.

Pour chaque réponse vous devez montrer comment vous l'avez obtenue, que ce soit à l'aide de sources dûment citées ou de captures d'écran. Chaque capture d'écran doit contenir les matricules des membres et la date.

[Linux] `echo matricule1-matricule2 `date``

Le TP contient 100 points avec une possibilité de 5 points bonus. La note maximale ne peut dépasser 100 et aucun point ne sera transféré sur un autre TP. La question BONUS est évaluée en mode 0 ou 5.

3. Fonctionnement des utilitaires et fondements

3.1. Sources d'information

Il a été vu en classe que la notion abstraite d'une source peut être décrite comme une boîte noire qui, à chaque fois qu'on lui demande (qu'on pèse sur le « bouton »), donne un nouveau symbole. L'ensemble de symboles qu'une source peut générer est son « alphabet ». Pour les besoins de ce travail pratique, nous allons « simuler » ces boîtes abstraites par des programmes exécutables. Ces programmes se trouvent dans l'archive « Utilitaires TP1 » sur le site Moodle :

— « lettre » génère une lettre de façon aléatoire. La probabilité qu'une lettre déterminée soit générée correspond à sa fréquence dans la langue anglaise.

— « texte » est similaire à « lettre », excepté qu'à chaque invocation le nombre de lettres indiqué sont tirées, dans l'ordre, d'un vrai texte en anglais. À chaque invocation, le texte choisi est différent.

Il s'agit de lettres de l'alphabet anglais (26 lettres) et l'espace. Dans le cas de texte, l'apostrophe est représentée par un espace.

Ces programmes pourraient nécessiter qu'on leur accorde des permissions d'exécution (chmod +x) ou qu'on installe un paquet pour supporter les programmes 32-bits sur un système 64-bits.

L'usage de ces programmes est très simple : à chaque invocation du programme, un nouveau symbole est généré et affiché à la console :

```
$ ./lettre
E
```

Vous pouvez spécifier un argument numérique qui spécifie le nombre de symboles qui seront générés

```
$ ./lettre 4
E GK
```

Vous pouvez utiliser les commandes de redirection « > », « < » pour rediriger la sortie de la source vers un fichier, ou même « | » pour passer la sortie par un autre programme.

Les compromis d'implémentation (« features ») suivants ont été introduits :

— Pour les sources « alphabétiques » (lettre et texte), ce qui est généré est les versions ASCII des lettres majuscules correspondantes.

La visualisation directe de la sortie des sources alphabétiques est possible sur la console ou via un utilitaire comme cat ou more. Quoique la sortie des sources binaires puisse être observée directement, il est préférable d'utiliser un éditeur ou un visualisateur de fichiers binaires, par exemple fb, xxd ou hexdump qui sont gratuits et disponibles sur Linux.

3.2. Entropie

Nous vous avons fourni trois outils qui vous permettront d'estimer l'entropie des différentes sources. Ils calculent tous l'entropie de l'entrée, mais ce de façon différente.

— « h-bit » calcule les fréquences bit par bit, et calcule l'entropie « par bit », c'est-à-dire comme si l'entrée était la sortie d'une source binaire. L'utilitaire utilise l'entrée standard. Par exemple :

```
$ ./h-bit < fichier.bin
0 = 490
1 = 534
Nombre total de bits : 1024
Entropie du texte entre : 0.998668
```

Notez que dans notre cas, et étant donné que la sortie des sources « alphabétiques » est déjà codée en binaire (via le codage ASCII) nous pouvons aussi appliquer cet outil sur ces sources.

```
$ ./lettre 128 > texte.bin
$ ./h-bit < texte.bin
0 = 647
1 = 377
Nombre total de bits : 1024
Entropie du texte entre : 0.949252
```

— « h-ascii » calcule les fréquences octet par octet d'une source, et calcule l'entropie « par octet », c'est-à-dire en considérant l'entropie comme s'il s'agissait d'une source générant des octets. Tout comme h-bit, l'utilitaire utilise l'entrée standard.

— « h-lettre » calcule les fréquences octet par octet, mais seulement pour les codes ASCII représentant des lettres majuscules et l'espace. À partir de ces fréquences, l'entropie (en bits) est calculée pour une source générant des lettres majuscules et des espaces. Cet utilitaire utilise aussi l'entrée standard.

3.3. Chiffrement

— « masque » est une implémentation du masque jetable (« one-time pad »). L'utilitaire nécessite 4 arguments qui sont les suivants, en ordre :

  — fichier contenant la clé (qui doit se trouver dans le même répertoire que le programme)

  — taille de la clé (en octets)

  — fichier contenant le message à chiffrer (qui doit se trouver dans le même répertoire que le programme)

  — fichier de sortie qui contiendra le message chiffré (qui sera créé dans le même répertoire que le programme)

4. Travail demandé

4.1. Entropie et sources d'information (10 points)

1. Supposez un alphabet de 35 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
2. Peut-on dire que cette entropie est maximale? Pourquoi? /2
3. 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
4. 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. Bref, se baser sur l'idée générale permettant la compression. /2

4.2. La librairie de Babel (10 points)

La librairie de Babel (https://libraryofbabel.info/) est un site internet qui exploite un concept unique. Il faudra utiliser la fonction « Browse » de la librairie afin d'obtenir un texte unique. En ce sens, vous utiliserez votre matricule et celui de votre binôme.

1. D'abord, avec le premier matricule, vous allez récupérer le texte se situant à matricule, wall 1, shelf 1, volume 1, page 1. Puis, calculez l'entropie par octet de ce texte (h-ascii). /2
2. 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
3. Commentez la variation d'entropie que vous observez. /3
4. Commentez sur l'entropie de la librairie de Babel. /3

4.3. 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 en vous assurant qu'il ne contienne que des lettres majuscules et des espaces. Il ne devrait pas contenir de chiffres.
2. Utilisez le site cyberchef (https://gchq.github.io/CyberChef/) pour chiffrer ce texte avec l'algorithme ROT13. Assurez-vous que le résultats ne contienne que des lettres majuscules et des espaces.
3. Utilisez le programme h-lettre sur la version avec ROT13 et sur la version sans ROT13, puis faites deux histogrammes avec un tableur pour le représenter visuellement. /1
4. Utilisez le programme lettre pour générer une séquence de 3200 caractères, puis encodez cette séquence avec ROT13.

5. Utilisez le programme h-lettre sur la version avec ROT13 et sur la version sans ROT13, puis faites deux histogrammes avec un tableur pour le représenter visuellement. /1
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
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

4.4. Masque jetable (20 points)

Vos réponses à cette section doivent être adéquatement documentées. Vous ne pouvez pas citer un modèle de language, ni Wikipédia.

1. Utilisez le langage de programmation de votre choix pour créer un programme qui génère en binaire un masque jetable, avec une probabilité égale d'avoir un 0 ou un 1. Python est suggéré. Vous devez donner le code de ce programme et expliquer son fonctionnement. Votre devez utiliser le module random de base du langage choisi, et trouver une alternative plus sécuritaire pour générer des nombres aléatoires pour le second. Expliquez aussi en quoi cette alternative est plus sécuritaire. /12
2. Utilisez votre programme pour générer des clefs ayant autant de bit que nécessaire (normalement 8 par caractère) pour couvrir les 100 premiers caractères du texte fourni en 4.3.1 **après avoir enlevé les espaces**. D'abord avec le module random de base, puis avec l'alternative plus sécuritaire.
3. Utilisez l'utilitaire masque pour appliquer vos clés sur le texte et calculer l'entropie par bit (h-bit) avant et après l'application de chaque masque. Faites de même avec l'entropie par octet (h-ascii). /3
4. Commentez les résultats. Dites à quoi vous vous attendiez, si les résultats sont conformes à vos attentes ou non, et pourquoi. Au besoin, mentionnez des formes de biais possible. Votre justification est évaluée ici et non le résultat en soi. /5

4.5. Communication à clé publique, HTTPS et SSL (5 points)

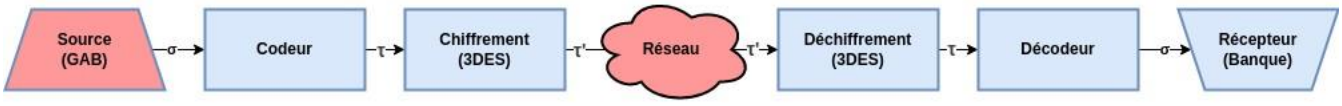
Vos réponses à cette section doivent être adéquatement documentées. Vous devrez certainement effectuer des recherches. Vous ne pouvez pas citer un modèle de langage, ni Wikipédia.

1. Expliquez la différence entre HTTP et HTTPS dans vos mots. /0,5
2. Expliquez pourquoi il est impossible de se connecter au dossier étudiant si on spécifie le protocole http (http://dossieretudiant.polymtl.ca) et dites quelle solution sécuritaire pourrait être mise en place pour que quelqu'un qui consulte ce lien puisse accéder au dossier étudiant. /1

3. Quelle est l'utilité du header « Strict-Transport-Security »? Vous pouvez constater sa présence en ouvrant l'inspecteur de votre navigateur sous « network » en visitant par exemple le site de la banque TD (https://www.td.com/) /1
4. À quoi sert un certificat à clé publique? Comment votre navigateur vérifie-t-il l'identité du propriétaire du site que vous avez visité? /0,5
5. Utilisez openssl pour générer un certificat « Self-Signed ». Donnez dans un tableau les champs de votre certificat. /0,5
6. Lancez un serveur python local qui utilise votre certificat avec le script python au lien suivant (Vous devez fournir la clef et le certificat en format .pem.) : https://gist.github.com/SeanPesce/af5f6b7665305b4c45941634ff725b7a
7. Tentez d'accéder au serveur avec le navigateur Firefox. Quel problème rencontrez-vous et comment pourriez-vous le régler? Expliquez votre démarche et votre raisonnement. /1,5

4.6. Codage (9 points)

La Banque de Sherbrooke a installé dans chacune de ses succursales une machine d'écriture de carte bancaire qui permet au client de changer le NIP (numéro d'identification personnel de 4 caractères (chiffres et lettres majuscules) associé à leurs cartes bancaires et qui est utilisé comme mot de passe par les guichets automatiques bancaires (GAB). Comme les NIPs doivent être envoyés à l'ordinateur central de la banque pour que les GAB puissent les vérifier lors de transactions bancaires, les machines d'écriture enregistrent les nouveaux NIP et les envoient à l'ordinateur central par réseau à chaque fois qu'un NIP est changé. Actuellement, le guichet automatique encode un NIP en prenant la suite de 4 caractères encodés en ASCII répété 2 fois pour détecter les erreurs de transmission. Donc, le NIP « 1,2,3,A » serait représenté par la séquence « 123A123A » en ASCII. Ceci créé donc un bloc de 8 octets, donc 64 bits. Pour éviter que les NIPs soient interceptés, ce bloc est alors chiffré avec l'algorithme Triple DES avant d'être envoyé par réseau. Étant donné que la clé de chiffrement est stockée dans une puce à l'épreuve des manipulations à l'intérieur du guichet automatique, il est assez difficile de changer la clé. Avec l'implémentation décrite, le système de transmission des NIPs est possiblement vulnérable. Vous devez développer un codeur pour remplacer le codage existant afin de minimiser la vulnérabilité du système à l'interception. Le reste de la transmission (identité du client, ...) est considérée sûre et indépendante.



Les zones en rouge dans le schéma sont les seules ou un attaquant peut intervenir. Il peut se rendre sur place et interagir avec le GAB et il peut manipuler les paquets sur le réseau. Il ne peut pas faire de bruteforce sur le chiffrement 3DES en l'état.

1. Expliciter les alphabets σ, τ et τ' qui sont respectivement les alphabets pour la sortie de la source, du codeur et du bloc de chiffrement. /3
2. Identifier les langages provenant de ces alphabets. /3
3. Ensuite, identifiez les attaques auxquelles le système est vulnérable. Pour identifier ces attaques, rappelez-vous qu'un attaquant peut connaitre 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. /3

4.7. Changement de codage (5 points)

On propose trois nouveaux codages pour le problème du 4.6.

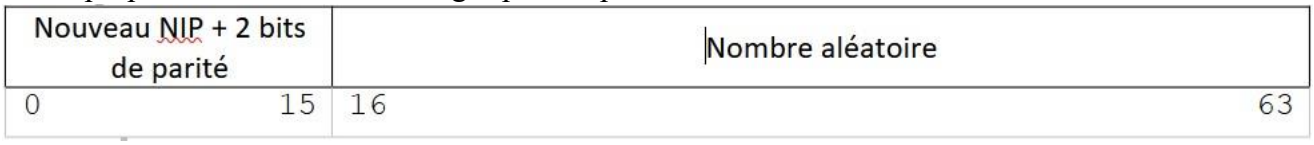


Figure 1 – Codage 1 : le nouveau NIP est codé en binaire sur 14 bits plus 2 bits de parité (les détails ne sont pas importants). Un nombre aléatoire sur 48 bits est concaténé pour former un bloc de 64 bits.

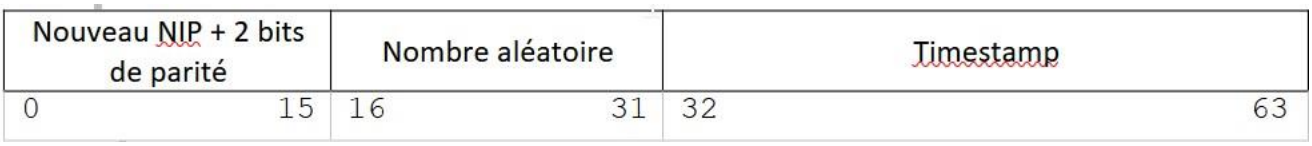


Figure 2 – Codage 2 : le nouveau NIP est codé en binaire sur 14 bits plus 2 bits de parité. Un nombre aléatoire de 16 bits puis un « timestamp » Unix de 32 bits sont concaténés pour former un bloc de 64 bits.

Figure 3 – Codage 3 : le nouveau et l'ancien NIP sont codés en binaire sur 14 bits plus 2 bits de parité puis concaténés. Un « timestamp » Unix de 32 bits est concaténé pour former un bloc de 64 bits.

Remarque : on suppose que le « timestamp » peut être utilisé par la banque pour invalider les messages dont le timestamp est trop vieux.

1. Pour chacun des trois codages, dites quelles attaques du 4.6.3 ils permettent de bloquer. /3
2. Selon vous, quel est le meilleur codage? Pourquoi? /2

4.8. Chiffrement par bloc et modes d'opération (4 points)

L'administrateur réseau de la compagnie WebSimple 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).

1. 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, chiffrez ce fichier en mode ECB. (Exécutez le script sans argument pour connaître son fonctionnement). Observez le fichier de sortie et commentez. /1
2. Chiffrez maintenant le fichier en mode CBC. Observez le fichier puis commentez. /1
3. Concluez sur l'importance des modes d'opération des algorithmes de chiffrement par bloc. /2

4.9. Organisation des mots de passe en UNIX/Linux (12 points)

Utilisez une machine Kali fraîchement installée sur laquelle vous avez accès root. Vos réponses à cette section doivent être adéquatement documentées. Vous ne pouvez pas citer un modèle de language, ni Wikipédia.

1. Examinez le fichier /etc/passwd. Contient-il des mots de passe? Pourquoi? Quelles sont ses permissions d'accès? Pourquoi? /1
2. Observez les fichiers passwd et shadow qui se trouvent sous le répertoire /etc/. Ajoutez un utilisateur avec la commande ci-dessous. Observez ce qui se passe dans les fichiers passwd et shadow. Lequel ou lesquels de ces deux fichiers sont modifiés? Pourquoi? /2
   ```
   $ useradd –g users -s/bin/bash -m NOM
   ```
3. Donnez un mot de passe à l'utilisateur que vous avez créé avec la commande ci-dessous.
   ```
   $ sudo passwd USERNAME
   ```
   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? /2
4. 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? /2
5. 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? /2
6. Dites comment il est possible de déchiffrer un mot de passe hashé. /3

4.10. Choix des mots de passe (5 points + 5 bonus)

Dans cette question, vous allez vous familiariser avec l'outil « John The Ripper » qui permet de trouver des mots de passe. Le principe de base est très simple : en regardant les entrées dans un fichier /etc/passwd, « John » va essayer de trouver des mots de passe qui, une fois haché avec le bon « salt », vont donner les hash des utilisateurs. Il prend des mots dans un dictionnaire de mots de passe courants et génère des hash avec les « salt » retrouvés et les compare avec chacun des hash. Il utilise des fichiers /etc/passwd à l'ancienne (qui contiennent aussi le hash de mot de passe). Cependant, il est possible de reconstruire ce type de fichier à partir du fichier /etc/passwd moderne et du fichier /etc/shadow en utilisant la commande « unshadow » qui vient avec John The Ripper. Pour cette question on vous fournit le fichier « password_file.txt » sur Moodle.

1. Utilisez « John The Ripper » avec le dictionnaire « rockyou.txt », et identifier le mot de passe correspondant à chaque utilisateur du fichier « passwords » sur Moodle. Le dictionnaire est pré-installé sur Kali dans /usr/share/wordlists/, mais il est compressé. Vous pouvez le décompresser avec la commande « gunzip ». /2
2. A votre avis pourquoi il est conseillé de ne pas utiliser le même mot de passe partout? Donnez aussi un exemple de situation réelle. /3
3. BONUS 5 pts - Faites un lien entre votre réponse en 4.9.6 et votre réponse en 4.10.2. /+5

4.11. Déchiffrement simple (15 points)

Vous êtes en possession d'un extrait de 200 caractères chiffrés qui provient d'un texte en anglais qui vous est inconnu. Rien n'indique que le début de cet extrait est un début de mot et que la fin est une fin de mot. Il y a 27 caractères possibles dans l'extrait chiffré que vous possédez, soient les 26 lettres de l'alphabet et le caractère « @ ». Ceci correspond à un alphabet en clair composé des 26 lettres de l'alphabet et de l'espace. Il est également important de mentionner qu'une fin de phrase est représentée par deux espaces dans le texte original, soit avant que le chiffrement ait été effectué. Récupérez le texte chiffré qui vous a été assigné sur Moodle. L'utilitaire frequency (dossier « Source – Entropie - Chiffrement ») est mis à votre disposition pour vous aider à effectuer le déchiffrement. Il sert à calculer le nombre d'occurrences des caractères dans un texte. L'option –n permet de spécifier la taille de bloc. Dans le cas où celle-ci est omise, une taille de 1 sera utilisée.

1. Donnez une capture d'écran de votre utilisation de l'utilitaire. /1
2. Fournir le texte déchiffré, ainsi que votre démarche clairement expliquée. /14

Pour faciliter votre tâche de déchiffrement, les fréquences des lettres en anglais vous sont fournies dans le tableau de la figure 4.

Il est également à noter que l'espace est 1.07 fois plus fréquent que la lettre e en anglais.

Les digrammes les plus courants en anglais sont, en ordre : th, he, in, en, nt, re, er, an, ti, es, on, at, se, nd, or, ar, al, te, co, de, to, ra, et, ed, it, sa, em, ro.

Les trigrammes les plus courants en anglais sont, en ordre : the, and, tha, ent, ing, ion, tio, for, nde, has, nce, edt, tis, oft, sth, men.

Lettre	Frequence	Lettre	Frequence
e	12.702%	m	2.406%
t	9.056%	w	2.360%
a	8.167%	f	2.228%
o	7.507%	g	2.015%
i	6.966%	y	1.974%
n	6.749%	p	1.929%
s	6.327%	b	1.492%
h	6.094%	v	0.978%
r	5.987%	k	0.772%
d	4.253%	j	0.153%
l	4.025%	x	0.150%
c	2.782%	q	0.095%
u	2.758%	z	0.074%

Figure 4 – Fréquences