INF4420: TP1

Troclet Philippe 1815208 Delaite Antoine

20 octobre 2015

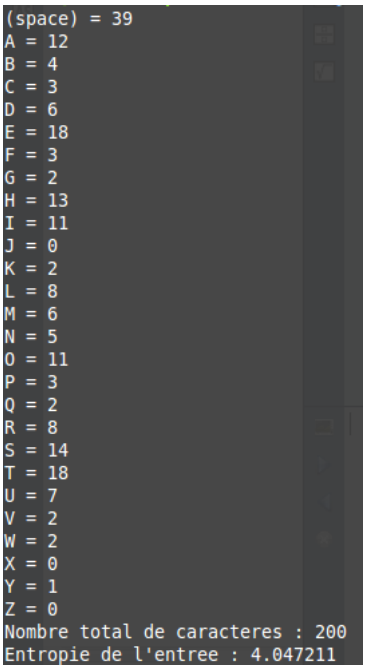

FIG. 1 – Calcul de l'entropie sur un texte issu de ./texte

1 Sources d'information, entropie et gestion du risque

1.1 Question 1 : étude de l'entropie

Après avoir généré un texte de 200 charactères via l'utilitaire texte (le texte est donc soumis aux règles de la langue anglaise), nous obtenons les résultats suivants visibles sur la figure 1. On obtient alors une entropie proche de $4bits$. Intuitivement, cela signifie qu'il faut en moyenne $4bits$ pour coder une lettre de l'alphabet (espace compris). Toutefois, la valeur obtenue (figure 1), est légèrement supérieure à 4, il serait donc plus juste de dire, qu'en moyenne, il suffit de quatre bits pour coder une des 26 lettres de l'alphabet. Sous réserve de ne pas différencier les majuscules des minuscules. A titre de comparaison, on pourrait calculer l'entropie d'une variable (notée S) aléatoire générant les 27 symboles de façon équiprobable. On aurait alors :

$$H(S) = \log_2(27) \approx 4.75$$

Dans le cas présent, le taux de compression est égal à $t_c = \frac{H(S)}{log_2(N)}$, on obtient donc approximativement $t_c \approx \frac{4}{4.75} \approx 0.84$.
Considérons maintenant une source générant des lettres (ou un espace) de manière à ce que leur fréquence respective soit celle de la langue anglaise. On veut calculer l'entropie d'un texte de 200 caractères afin de comparer avec les résultats précédents. On obtient les résultats exposés dans la figure 2. Dans ce cas, on obtient donc une entropie de 4.11. Il apparaît que l'entropie des deux

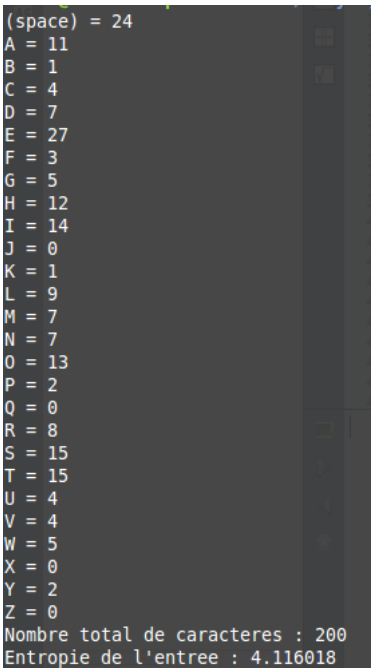

FIG. 2 – Calcul de l'entropie sur un texte issu de ./lettre

textes est très proche. Ce résultat, surprenant au premier abord, s'explique de la manière suivante : le calcul de l'entropie se base uniquement sur la distribution des caractères. De ce fait, les règles de grammaire, et la prévisibilité qu'elles impliquent ne figurent pas dans l'entropie. De plus, d'un point de vue global, les fréquences d'apparition des différents caractères dans les deux textes devraient être proches. En effet, le premier est issu d'un ouvrage anglais, et le second suit la répartition des caractères dans la langue anglaise. Il est donc naturel que pour un texte suffisamment grand les fréquences soient proches.

1.2 Question 2 - Histogrammes

Dans ce qui suit, nous allons étudier les différentes fréquences du texte suivant :

MIGHTY A NATION DERBY BE THOU EMBASSADOR FOR VS VNTO OUR FATHER IN LAW THE EARLE OF HENALT MAKE HIM ACQUAINTED WITH OUR ENTERPRISE AND LIKEWISE WILL HIM WITH OUR OWNE ALLIES THAT ARE IN FLAUNDSRS TO S

Texte, qui, une fois chiffré via le chiffrement de César, devient :

PLJKWB D QDWLRQ GHUEB EH WKRX HPEDVVDGRU IRU YV YQWR RXU IDWKHU LQ ODZ WKH HDUOH RI KHQDOW PDNH KLP DFTXDLQWHG ZLWK RXU HQWHUSULVH DQG OLNHZLVH ZLOO KLP ZLWK RXU RZQH DOOLHV WKDW DUH LQ IODXQGVUV WR V

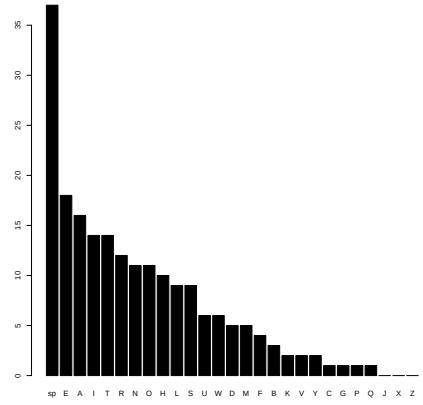

FIG. 3 – Texte original

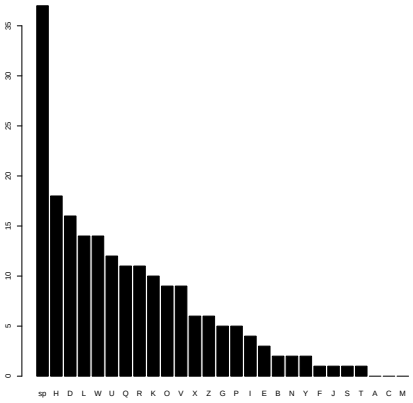

FIG. 4 – Texte chiffré

On peut observer la répartition des fréquences, avec ou sans chiffrement, sur les figures suivantes : 3 et 4.

L'étude du texte précédent sera accompagnée de celle d'un texte généré via le programme lettre. Ce texte est le suivant :

SIRNOIEN C TIDTE ELBET HV GOO ESDSTANRTLETMFOERMHHMA VDIT EGIAY D HER DOCDOEO SE MIAY REWAOTFSI I OHUAE DTCO WGT AH DNYCAEPVHNEERAIA PSTOCS ISSAODT SAEP ENUUUETNSRA WOPSDS ISWH EHMPOONNKTEGR SRDHHIRE

Ce texte, une fois chiffré par le même algorithme que précédemment, devient :

VLUQRLHQ F WLGWH HOEHW KY JRR HVGVWDQUWOHWPIRHUPKKPD YGLW HJLDB G KHU GRFGRHR VH PLDB UHZDRWIVL L RKXDH GWFR ZJW DK GQBFDHSYKQHHUDLD SVWRFV LVVDRGW VDHS HQXXXHWQVUD ZRSVGV LVZK HKPSRRQQNWHJU VUGKKLUH

On peut observer l'histogramme des fréquences de ce texte avant le chiffrement sur la figure 5 et après sur la figure 6. Dans les deux cas, que le texte provienne du programme lettre ou du programme texte, on contaste que les diagrammes avant et après chiffrement ont la même forme. Cela provient du fait que le chiffrement de César est une simple transposition : la version chiffrée d'une lettre ne dépend que de la lettre elle-même. De ce fait, dans le cas présent, un "E" en entrée donnera toujours un "H" en sortie. Cette propriété de ce chiffrement (ou sa mauvaise diffusion), semble indiquer qu'une analyse fréquentielle serait faisable.

De plus, les différents histogrammes avant chiffrement sont certes proches de l'histogramme des fréquences en anglais. Cet histogramme est visible sur la figure 7. Toutefois, il y a certaines différences dans la répartition des caractères, aussi l'analyse sera relativement compliquée à mettre en place dans le cas général : un simple analyse des fréquences ne suffira pas, il faut identifier grâce aux fréquences la lettre "e" et en déduire le chiffrement utilisé.

Comptabiliser les fréquences des digrammes permet de déchiffrer plus facilement un texte issu de l' utilitaire ./texte. En effet, dans la plupart des cas, on

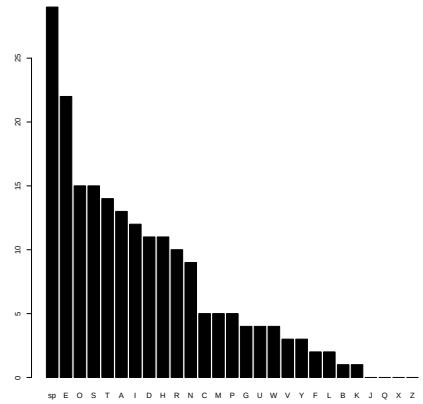

FIG. 5 – Texte (issu de lettre) original

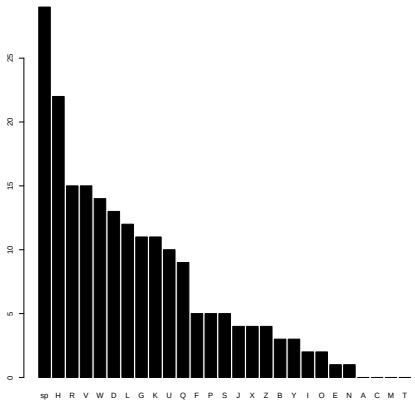

FIG. 6 – Texte (issu de lettre) chiffré

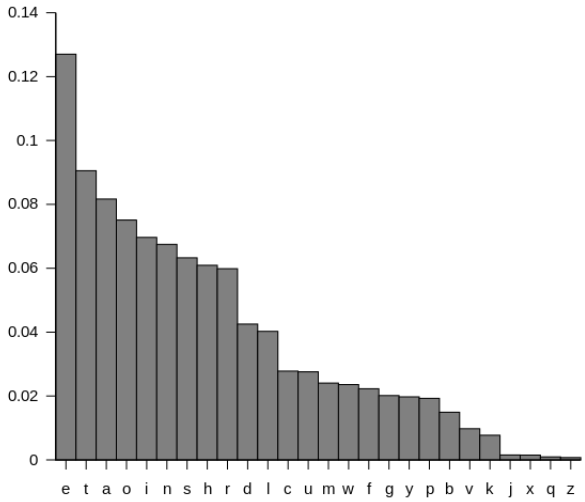

FIG. 7 – fréquence des lettres en anglais

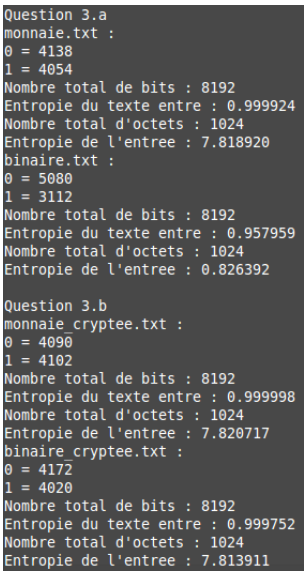

FIG. 8 – Calcul de l'entropie via h-bit et h-ascii

devrait avoir $H(S^2) < 2 * H(S)$ et donc $\frac{H(S^2}{2} < H(S)$. L'entropie par caractère étant alors plus faible que si on n'utilisait pas les digrammes, il devrait être plus facile de déchiffrer le texte. Cela provient du fait, qu'on texte anglais obéit à une certaine structure. Par exemple, si on lit un "t", il est probable que la lettre suivante soit un "h" (cette lettre est la plus fréquente après un "t" d'après le tableau des digrammes page 41 des acétates crypto1). On devrait donc avoir un histogramme moins plat si on utilise des digrammes dans ce cas.

En revanche, la méthode des digrammes n'est pas efficace dans le cas de la source ./lettre, cette dernière étant une source markovienne l'entropie du digramme est égale à deux fois l'entropie de la source initiale. De ce fait, l'entropie par symbole est la même que nous utilisions des digrammes ou non. L'histogramme devrait donc pas connaître de changement majeur si on utilise des digrammes. Ainsi, le doublon "ee" devrait être très présent dans le texte issu de ./lettre, car "e" est très fréquent en anglais, alors que dans un texte issu de ./texte, la fréquence de ce digramme sera faible. A l'inverse, le digramme "th" sera très présent dans le texte issu de la langue anglaise (ce digramme a une fréquence de l'ordre de 3.7% d'après le tableau de fréquence des acétates), mais moins présent dans le cas d'un texte provenant de lettre (la probabilité du digramme sera alors égale au produit des probabilités, cette dernière devrait donc être de l'ordre de $0.06 * 0.09 = 0.005$)(le calcul se base sur les fréquences de la figure 7).

De plus, déchiffrer un tel texte présente un problème majeur : le texte étant aléatoire, il n'a pas de sens, il est donc extrêmement difficile de savoir si le texte que l'on pense avoir déchiffré est réellement le bon texte.

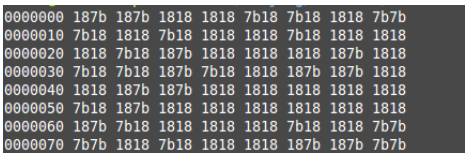

FIG. 9 – Une sortie de ./binaire en héxadécimal

1.3 Question 3 : One-time padding

On génère deux fichiers, un via l'utilitaire ./monnaie, l'autre via ./binaire. On calcule alors l'entropie par bit puis par caractère. Les résultats de ces calculs sont visibles sur la figure 8. Pour monnaie, on trouve une entropie par bit proche de 1 (0.9999) et une entropie par caractère proche de 8 (7.8189). Cela provient du fait que les bits produits par monnaie sont indépendants et uniformément distribués. L'entropie est donc maximale. En revanche le résultat obtenu pour le fichire issu de binaire est plus surprenant. On a une entropie élevée par bit (0.9579), mais une entropie faible par octet (0.8263). Pour comprendre ce phénomène, observons les valeurs produites par ./binaire une fois traduit en hexadécimal, ces dernières sont visibles sur la figure 9. On notera que la première colonne correspond à la position dans le fichier. On remarque alors que binaire ne fournit que deux octets différents : $0x18$ et $0x7b$. Ce qui explique la faible entropie du texte lorsqu'on considère des octets. De plus ces octets sont complémentaires en nombre de "0" et de "1" d'où l'entropie élévée par bit.

On génère maintenant une clé via monnaie pour chiffrer les deux fichiers générés précédemment. L'entropie des fichiers cryptés ainsi obtenus est visible sur la figure 8. On remarque que dans les deux cas, les entropies par bit et par octet sont presque maximales. C'est-à-dire proche de 1 dans le premier cas et proche de 8 dans le second. Cela traduit le fait que la méthode du masque jetable est un algorithme très sécuritaire : si la clé est totalement aléatoire, il est impossible pour un attaquant de déchiffrer le message. Le caractère aléatoire de la clé maximise l'entropie du message chiffré. Précisons que pour qu'il soit totalemant sécuritaire, il faut la clé ne soit utilisée qu'une fois.

1.4 Question 4 : Analyse de risque

a) Si le but de l'entreprise est de devenir un acteur du jeu de poker en ligne sur le long terme alors il est évident qu'il faut choisir l'ile paisible. En effet un impératif pour un fournisseur de service passant par internet est la fiabilité de la connexion. On ne pourrait pas se permettre de couper le traffic (cela peut en plus arriver en pleine partie) et d'entamer de longues réparations. Les utilisateurs cherchent un site fiable et stable. De plus notre coeur de cible sont les joueurs addict donc une interruption de notre service nous ferait perdre beaucoup de clients. Sans avoir de chiffres exacts on aurait une esperance de perte d'un quart de l'argent investi (100 000) plus le manque à gagner, les réparations et la perte définitive de clients. Disons que cela fasse 100 000$, ce risque étant présent tous les ans, cela nous couterait en moyenne 100 000 par an. Ce serait moins rentable au dela de cinq ans.

Acteur	Capacité	Opportunité	Motivation	Probabilité	Impact	Risque
Tricheur	4	4	4	64	2	128
C.O	1	4	1	4	2	8
Concurrent	2	4	2	16	2	32

b) Le scénario 1 touche à l'intégrité de notre système. En effet les données transmises à nos serveurs et autres joueurs sont altérées et ne rentrent plus dans le cadre du fonctionnement normal.

Le scénario 2 touche à la disponibilité du système. Un script kiddy envoie une quantité de requêtes qui sont traitées par nos serveurs. Le but de celles-ci est de surcharger nos serveurs en travail et ralentir le bon déroulement des parties.

Le scénario 3 touche à la confidentialité du système. En effet il va profiter d'une faille de sécurité et accéder à des données confidentielles de tierces personnes auxquelles il ne devrait pas avoir accès.

c) Scenario 1 :

Les tricheurs sont le risque principal : Un invidu par le biais d'une tierce aurait récupéré le code de l'application et pourrait compter les cartes et deviner à l'avance les cartes qui vont être triées.

Scénario 2 :

Acteur	Capacité	Opportunité	Motivation	Probabilité	Impact	Risque
Tricheur	1	4	1	4	4	16
C.O	4	4	1	16	4	64
Concurrent	2	4	4	32	4	128

Le concurrent représente un risque important.

Scénario 3 :

Acteur	Capacité	Opportunité	Motivation	Probabilité	Impact	Risque
Tricheur	1	3	1	3	3	9
C.O	4	3	4	48	3	144
Concurrent	1	3	2	6	3	18

La bande organisée représente le plus gros risque.

d) Dans le premier scénario, les concurrents vont faire face à une baisse de revenus. Si ils ne trouvent pas de moyen d'inverser la tendance en trouvant de nouvelles solutions pour se relancer ils iront vers la faillite. Face à la faillite et à la perte de revenus, toutes les solutions peuvent être envisagées par un esprit mal intentionné. Dans une situtation ou il n'aurait rien à perdre, un concurrent pourrait décider d'attaquer la disponibilité (par exemple) de notre système en représailles ou pour nous faire perdre du marché. Le facteur de motivation du concurrent va donc grandement augmenter, quoique le facteur de capacité puisse

diminuer si l'entreprise perd ses moyens financiers et humains elle pourrait lancer l'attaque avant de tout perdre.

Dans le second, on peut imaginer que la mafia ne sera pas très contente et voudra se venger ou alors mettre la pression sur notre entreprise. Dans ce cas la motivation va augmenter du côté de la mafia ainsi que ses moyens si elle n'en dispose pas déjà. Les opportunités étant déjà assez forte au vu des hypothèses (proximité du patron avec des agents de la mafia), on imagine qu'ils ont accès plutôt facilement aux systèmes informatiques (inside men). Le risque va donc grandement s'élever.

Dans le dernier qui concerne les tricheurs, leurs motivation et opportunités ne vont pas changer. Cependant leur capacité de tricher va être grandement diminuée. Une fois un tricheur détecté on peut imaginer annuler une partie, bloquer son compte et ses gains, bloquer son adresse IP ou autre. Ainsi on réduit la capacité des tricheurs à zéro ou à une seule partie sans conséquence avec un bon système.

e)

Acteur	Capacité	Opportunité	Motivation	Probabilité	Impact	Risque
Tricheur	1	3	1	3	3	9
C.O	4	3+x	4	48+16x	3	144+48x
Concurrent	1	3	2	6	3	18

La bande organisée représentait déjà le plus gros risque. En faisant appel à un fournisseur de service extérieur on ne prend pas forcément de risque. Il faut toutefois faire attention à l'intégrité de celui-ci, basé en Russie connue pour ses hackers, qui ne craignent pas la justice française ou européenne. Il y a donc bien une augmentation du risque. En réalité on a surout augmenté l'opportunité pour le crime organisé car il sera plus proche physiquement et pourra attaquer directement le fournisseur, soudoyer un employé, ou dans le pire des cas le fournisseur appartient directement à la mafia. Imaginons qu'on ait doublé l'opportunité du C.O. (de 3 à 6) on va aussi doubler le risque (144 à 288). Il n'est donc pas recommandé de s'attacher les services de ce fournisseur.

2 Certificats, mots de passe et chiffrement

2.1 Question 1 : Codage

Commençons par identifier les différents alphabets utilisés dans la situation présente :
- de σ : les chiffres de 0 à 9
- de τ : les chiffres de 0 à 9
- de τ' : 0 ou 1

De ces alphabets, on déduit les langages suivants :
- de σ : w=(0-9)* avec |w|=4 (Le NIP de l'utilisateur)

- de τ : w=aa avec a=(0-9)* et |a|=4 (le NIP répété pour détecter les erreurs)
- de τ' : w=(0+1)* avec |w|=64

Le système de transmission est vulnérable à deux types d'attaque :
- Brouiller les messages (les modifier) afin que le système de soit plus fonctionnel.
- Casser la clé via une méthode de forcce brute

Détaillons les attaques décrites ci-dessus. Pour la premier type, l'attaquant va intercepter le message et le modifier. Il peut d'une part modifer certains bits du message pour que ce dernier soit invalide, afin d'empêcher le changement de NIP. Ou alors, il peut remplacer le message par un message plus ancien. Ce qui rend le système non-fonctionnel portant ainsi atteinte à la réputation de la banque qui sera accusée d'avoir du matériel de mauvaise qualité. De plus, on peut voir la deuxième attaque décrite (remplacer le message par un ancien) comme une manière d'obtenir le NIP de beaucoup de clients. En effet, si un complice va tôt le matin (ou tard le soir, l'idée étant qu'il soit seul dans la banque à se moment là, afin qu'il n'y qu'un seul message sur le réseau) on peut alors intercepter un message chiffré dont on connaît le contenu en clair (il provient d'un complice). On peut alors remplacer tous les messages suivants par un message valide dont on connaît le contenu : on connaîtra tous les NIP de tous les utilisateurs qui changeront leur NIP.

Pour le second type, c'est-à-dire une attaque de type force brute, on peut la mettre en place du fait que la longueur de la clé (56 bits) est facilement cassable de nos jours avec le matériel approprié et suffisamment d' ordinateurs. Notons que, dans le cas présent, cette attaque est d'autant plus faisable que, la clé étant intégrée dans le matériel, la banque ne peut la changer simplement (on n'a donc pas de limite de temps pour casser la clé). De plus, on peut prétendre utiliser la force brute du fait que lorsque qu'on tente de casser la clé sur un message, on peut, sans rencontrer de problème majeur, savoir si le message "déchiffré" obtenu est valide. En effet, le langage τ ne comporte que 10000 mots possibles or les messages étant codés sur 64 bits, on a 2^{64} possibilités. De ce fait, la probabilité qu'on tombe par hasard sur un message valide en utilisant une clé incorrecte est très faible.

Supposons maintenant qu'on change de code, et considérons le codage 1. Le NIP est maintenant codé sur 14 bits (on notera que $\log_2(10000) \approx 14$), on a 2 bits de parité, et le reste du message est constitué de bits aléatoires. Ce nouveau codage a une entropie bien plus élevée que le codage précédent ($H(Codage1) \geq 2^{61}$). Ce qui rend une attaque par force brute impossible car il est très difficile de savoir si le message qu'on obtient après avoir essayé une clé est bien celui envoyé : la majorité des messages qu'on obtiendra seront des messages valides. De plus, les bits de parité permettent de préservé l'intégrité du message : s'ils ont été modifiés, on le sait.

Utilisons maintenant le codage 2. Comme précédemment, le fait de coder le nombre sur 14 bits et d'ajouter un nombre aléatoire permet de contrer la force brute. Et l'ajout de bits de parité permet d'identifier les modifications indésirables. La force de ce codage, réside dans l'ajout d'un timstamp qui permet de vérifier, que le message est récent, ce qui empêche un attaquant de remplacer un message légitime par un message plus ancien. Ou du moins par un message

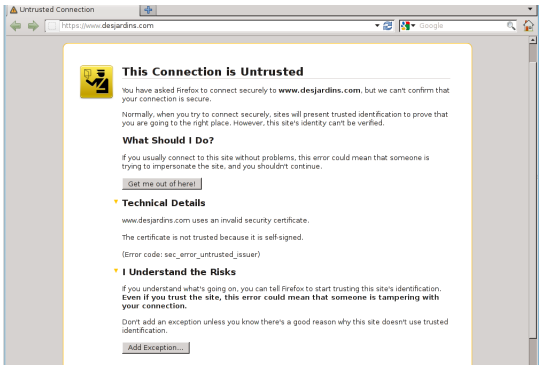

FIG. 10 – Message d'erreur affiché par firefox lors de la première connection au faux site www.Desjardins.com

trop ancien. Car alors, le timestamp serait incorrect.

Considérons le codage 3. Les considérations relatives au timestamp restent valides, ce qui a été dit sur la force l'est également, il est plus difficile de casser la clé via la force brute que lorsque le codage originel était utilisé, mais il est plus simple de la casser que dans les cas précédents. En effet, le message avant chiffrement est plus déterministe. En revanche, si l'attaquant parvient à remplacer le message par un autre dont le timestamp est encore valide, alors la présence de l'ancien NIP devrait, dans la majorité des cas, permettre de détecter que le message est frauduleux. (En effet la probabilité que deux personnes ait le même NIP est, si on considère que les NIP sont équiprobables, $\frac{10000}{10^8} = \frac{1}{10^{-4}}$).

Il nous faut maintenant choisir un codage à utiliser. Ce codage serait idéalement le meilleur des trois précédents. Ce choix dépend de l'efficacité du timestamp. S'il est simple d'utiliser d'anciens messages malgré le timestamp, alors le codage 3 s'impose. En revanche, si une telle manipulation est comliquée voire infaisable, le codage 2 est un choix judicieux. (La présence de bits aléatoires rendant une attaque par force brute très difficile). Dans le cas présent, du fait qu'il s'agit de machine d'écriture de cartes, la vitesse d'envoie des messages sur le réseau, est déterminée par la vitesse à laquelle les clients s'enchaînent sur une machine. Or on peut supposer que plusieurs secondes s'écouleront entre l'envoie du premier message et l'arrivée du second client qui devra s'identifier, choisir l'opération voulue, taper son nouveau NIP et finalement valider la transaction. Ce qui, par rapport à la vitesse des ordinateurs actuels est très long. De ce fait, un timestamp devrait suffir à empêcher qu'un message soit remplacer par un autre plus ancien. Il semble donc que le meilleur codage soit le codage 2.

2.2 Question 2 : Certificats à clé publique, HTTPS et SSL

Lors de la première connection avec le faux site, le navigateur refuse de se connecter au site et affiche un message d'erreur disant que la connection n'est pas sûre. Ce message est visible sur la figure 10.

Expliquons l'origine de ce message. Le https présent dans l'adresse du site nous informe que nous avons demandé l'utilisation d'une connection sécurisée. Aussi lors de l'établissement de la connexion, le navigateur cherche à vérifier

FIG. 11 – Certificat fourni par le site

l'identité du site. Pour cela, il consulte le certificat du site. Et, au moment de vérifier l'identé de l'autorité ayant validé le certificat, il apparaît que c'est le site lui-même qui se certifie authentique. Ce qui entraine la production d'un message d'avertissement.

On peut alors se douter que le site n'est pas sûr, en effet, il est très peu probable que le site d'une banque ait un certificat invalide. Un très grand nombre d'opérations sensibles sont susceptibles d'avoir lieu sur un tel site (virements, etc), aussi permettre aux utilisateurs de vérifier l'authenticité du site est une priorité. De plus, pour les mêmes raisons, le certificat d'un tel site sera toujours validé par une authorité de confiance connue et extérieure. On ne devrait donc jamais avoir de message du type : " the certificate is not trust because it is self-signed ".

Par ailleurs, si on examine le certificat, visible sur la figure 11, on peut voir que les noms d'organisation et de département sont plus que suspects.

Si maintenant on ajoute l'exception de sécurité (via le bouton "add exception" sur la figure 10) et qu'on redémarre firefox, la connexion se fera sans heurt. en effet, en ajoutant l'exception, nous avons dit à firefox que nous considérions que le site était légitime et sûr. De plus, l'exception ayant été ajoutée de façon permanente, le comportement sera le même à chaque connexion. En particulier, un autre utilisateur utilisant le même ordinateur pour se connecter au site n'aura aucun moyen de savoir que le site en question n'est pas légitime.

Créons mainteant notre propre certificat, pour cela il faut d'abord créer un CA qui signera le certificat produit pour le site. Ensuite ce CA sera ajouté dans firefox. Ainsi, si firefox rencontre un certificat qui prétend avoir été signé par ce CA, il pourra le vérifier. Ensuite, on génère un certificat pour le site et un l'ajoute sur le serveur. Après avoir effectué ces opérations si on tente de se connecter au site, il n 'y aura aucun problème et l'utilisateur verra la page du site sans aucun mesage d'erreur. Ceci provient du fait que le certificat fourni

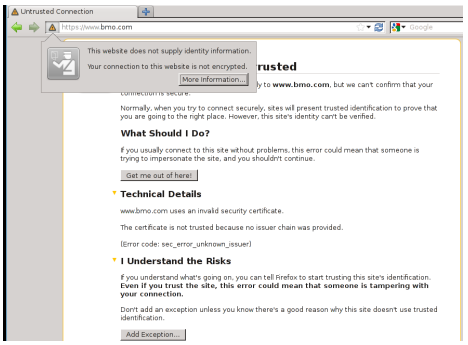

FIG. 12 – Message d'erreur sur le site de la bmo

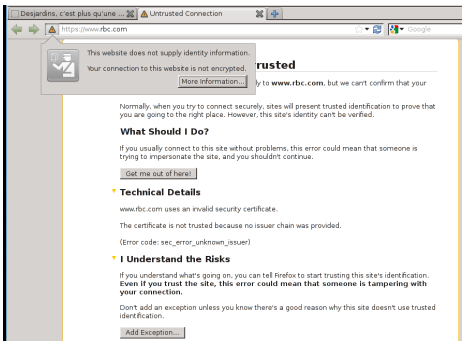

FIG. 13 – Message d'errreur sur le site de la rbc

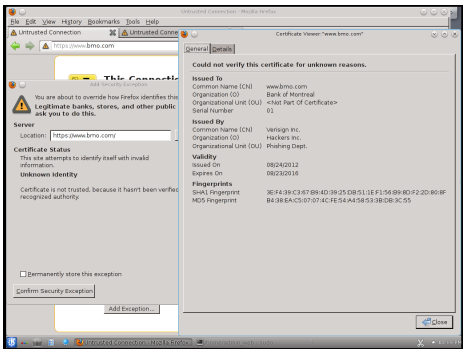

FIG. 14 – Certificat du site de la bmo

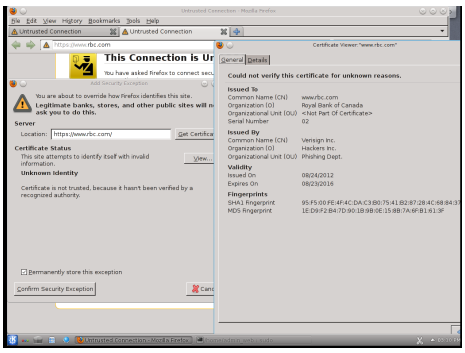

FIG. 15 – Certificat du site de la rbc

par le site est signé par un CA que nous avons ajouté à firefox et qui est donc considéré comme fiable.

Essayons maintenant de nous connecter aux deux sites suivants : `https://www.rbc.com`, `https://www.bmo.com`. Le navigateur nous affiche alors un message d'erreur nous informant que l'identité des sites n'a pas pu être vérifiée car les certificats fournis sont signés par des CA inconnues. ces messages sont visibles sur les figures 12 et 13.

En regardant les certificats fournis par les deux sites, visibles sur les figures 14 et 15. On peut voir que le CA inconnu de firefox dont il est question est Verisign Inc. Si on ajoute une exception temporaire pour le site de la bmo, on peut alors accéder au site `https://www.bmo.com`, en revanche si on change de site puis qu' on retourne sur le site on n'a plus accès à ce dernier. Cela provient du fait qu'on a ajouté une exception temporaire : on a demandé au navigateur de nous connecter au navigateur malgré les risques, en revanche firefox n'a pas enregistré le certificat. Aussi, lorsque ce dernier tente de se reconnecter au site, le certificat du site sera toujours invalide. En conséquence, le navigateur affiche un message d'erreur.

Ajoutons maintenant une exception de sécurité temporaire pour le site de la rbc, on applique la procédure habituelle, au terme de cette dernière, une fenêtre (figure 16) apparaît nous demandant si on veut ajouter un nouveau CA au navigateur. Si on retourne sur le site de la rbc après avoir vidé le cache du

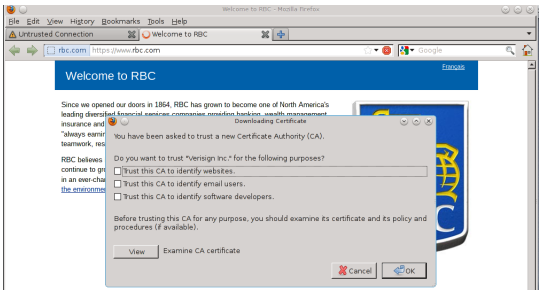

FIG. 16 – Demande d'ajout d'un CA

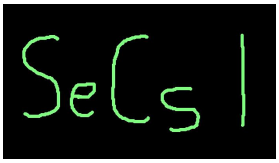

FIG. 17 – Image contenant le mot de passe

navigateur, l'accès au site se fait sans message d'erreur. De plus, si on tente de se connecter au site de la bmo, cette dernière se fait sans problème. En effet, ces deux site sont certifiés par le même CA, or on a ajouté ce CA au moment d'ajouter une exception temporaire pour le site de la rbc. Aussi, lorsque firefox cherche à vérifier l'identité des sites, il constate que leur certificat sont signés par un CA qui est considéré comme sûr.

Pour conclure cette partie, l'ajout d'un certificat self-signed est dangereux car on a aucune garantie sur l'identité du site : les informations fournies lors de la création d'un certificat ne sont vérifiées que s'il est signé par une authorité de confiance. Rien n'empêche une personne mal-intentionnée de communiquer des informations frauduleuses afin que le certificat paraisse légitime. De ce fait, un tel certificat peut cacher un site falsifié. De plus, l'ajout d'un certificat CA est particulièrement dangereuse car alors tous les sites certifiés par ce CA seront considérés comme légitimes. L'utilisateur pourra alors se connecter à de nombreux sites sans qu'aucun avertissement ne lui soit adressé. En particulier, si le CA a signé des certificats pour des sites malhonnêtes, l'utilisateur risque de transmettre des données confidentielles à des pirates qui en feront mauvais usage.

2.3 Question 3 Chiffrement par bloc et modes d'opérations

L'admistrateur a dissimilé un mot de passe dans une image (figure 17). Cette image sera chiffrée en utilisant l'algorithme AES 256. Nous allons comparer les différents modes de chiffrement utilisables (ECB et CBC). Dans un premier temps nous utilisons le mode ECB. La figure 18 est le résutat de l'utilisation d'un tel mode. On remarque alors que le mot de passe est encore clairement visible

FIG. 18 – Image chiffrée avec le mode ECB

FIG. 19 – Image chiffrée avec le mode CBC

bien que l'image soit chiffrée. Á l'inverse si on utilise le mode CBC, l'image chiffrée n'est pas directement exploitable par un pirate : le mot de passe n'est plus lisible. On peut voir le résultat de l'utilisation de ce mode est visible sur la figure 19. Cela vient du fait que le mode ECB chiffre tous les blocks identiques de la même façon, idépendamment du contexte. Ainsi il est important de bien choisir le mode utilisé, sinon, on met en danger la sécurité du message. En effet, l'utillisation de CBC, permet de diffuser un changement dans un block dans tout le message, et permet également de compliquer la relation entre le message est la clé.

2.4 Question 4 Organisation des mots de passe en UNIX/Linux

Un premier constat surprenant est que le fichier passwd ne contient pas de mot de passe. Son contenu est visible sur la figure 20. La raison de ce choix est à chercher du côté du contenu du fichier et de ses permissions. En effet, ce fichier

FIG. 20 – Contenu du fichier password

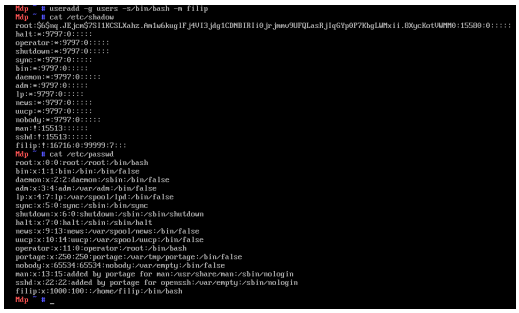

FIG. 21 – Contenu des fichiers passwd et shadow après la création d'un utilisateur filip

contient des informations sur les différents utilisateurs, telles que leur identifiant d'utilisateur ou leur identifiant de groupe. Certains programmes , tout à fait légitmes ont besoin de ces informations pour fonctionner. De ce fait tout le monde doit pouvoir lire ce fichier. En revanche, seul le propriétaire de la machine doit pouvoir modifier ce fichier, en effet, si un utilisateur modifiait ce fichier, il pourrait changer les identifiants et se donner accès à des ressources qui lui sont gardées. De manière générale si n'importe qui pouvait modifier ce fichier, il y aurait un risque de modifications accidentelles qui endommageraient le système.

De ce fait, il est impératif que seul l'utilisateur root puisse modifier ce fichier, mais qu'il soit accessible en lecture à tous. Ainsi, si les mots de passe étaient dans ce fichier, sous une quelconque forme, ils seraient visibles par tous et des attaquants pourraient utiliser cette vulnérabilité pour tenter de trouver des mots de passe. Il en résulte que ces derniers doivent être dans un autre fichier qui lui ne sera accessible que par l' administrateur système.

Si maintenant nous ajoutnos un nouvel utilisateur du nom de "filip", on observe que les deux fichiers changent, ces changements que nous allons commenter sont visibles sur la figure 21. Un premier constat est l'ajout d'une ligne débutant par le nom de l'utilisateur dans chaque fichier. De plus, le nouvel utilisateur a un "!" dans le champ qui correspond au mot de passe, cela caractérise le fait qu'il n'a pas encore de mot de passe.

On notera également que le mot de passe de root n'est pas directement visible, il a subi un hachage avant d'être stoqué. Par ailleurs, même la longueur du mot de passe n'est pas visible. Sa taille a été augmentée avant que le hachage ne soit effectué.

Nous allons maintenant donner un mot de passe à utilisateur. L'impact d'une telle opération est visible sur la figure 22. Comme sur cette figure, seul le fichier shadow a été modifié. En effet, l'ajout d'un mot de passe ne change en rien les informations qui sont contenues dans psswd : le nom et les identifiants de l'utilisateur restent les mêmes. Il en va de même pour son répertoire home et son shell. Ces données étant indépendantes de la présence d'un mot de passe.

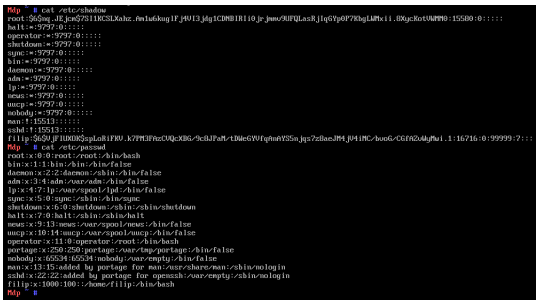

FIG. 22 – Contenu des fichiers passwd et shadow après l'ajout du mot de passe de filip

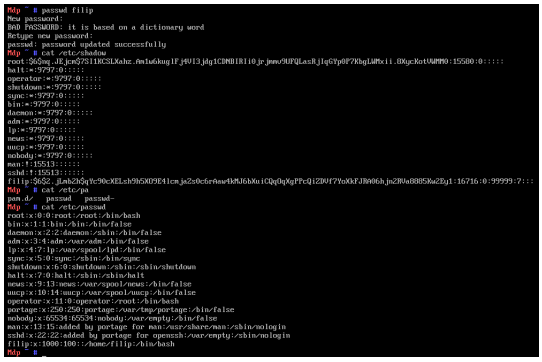

FIG. 23 – Changement du mot de passe de l'utilisateur

Après avoir ajouté un mot de passe, nous allons maintenant le changer, afin d'observer les conséquences que cette opération entraine. L'effet de cette manipulation sera visible sur la figure 23. Pour les mêmes raisons que précédemment, seul shadow est modifié. On peut en conclure que, les mots de passe sont bien stoqués dans le fichier shadow, que seul l'utilisateur root peut lire et modifier. Comme il a été dit précédemment, si un autre utiliateur pouvait lire ce fichier, il aurait des informations sur les mots de passe.

Un autre test intéressant consisite à changer le mot de passe d'un utilisateur, par le même mot de passe. Le résultat de ce test est visile sur la figure 24. Si on compare le résultat avec la figure 23, on constate que les mots de passe cryptés sont différents. Ce résultat qui peut surprendre, après tout il s'agit du même mot de passe, provient de la manière dont ces derniers sont chiffrés. D'arès le man, `http://man7.org/linux/man-pages/man3/crypt.3.html`, le chiffrement dépend du mot de passe mais aussi d'un salt (située entre "$" et "."). Aussi, les salt étant différents, on obtient deux résultats différents. Ce système permet de camoufler le fait que deux utilisateurs ont le même mot de passe.

Nous allons maintenant tenter de créer un deuxième utilisateur avec un mot de passe, puis de changer son mot de passe dans le fichier shadow. Une telle

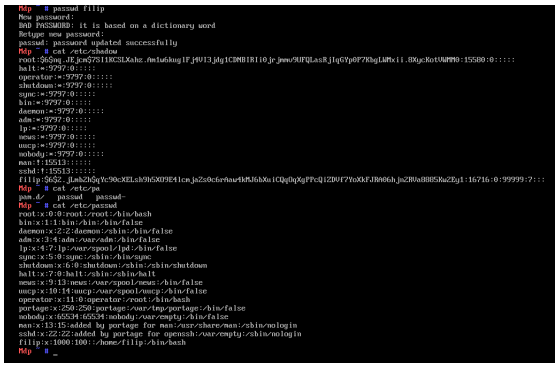

FIG. 24 – Changement par le même mot de passe

FIG. 25 – suppression d'un utilisateur

opération fonctionne, en effet, le mot de passe contient toutes les informations néccessaires pour reproduire le hachage : l'algorithme (entre les "$"), le salt (avant le ".") et le hash pour comparer avec le hash du mot de passe rentré. Ainsi, quand on rentre le mot de passe associé au hash qu'on a introduit, le calcul se fait correctement et l'authentification est acceptée. Notons qu'un tel remplacement n'a été possible que parce que nous étions root, cela ne constitue donc pas une faille de sécurité.

Pour conclure sur cette partie, nous allons maintenant supprimmer un utilisateur pour observer l'impact sur les fichiers passwd et shadow. Cela est visible sur la figure 25. On constate alors que les lignes correspondant à cet utilisateur ont été supprimées dans chaque fichier.

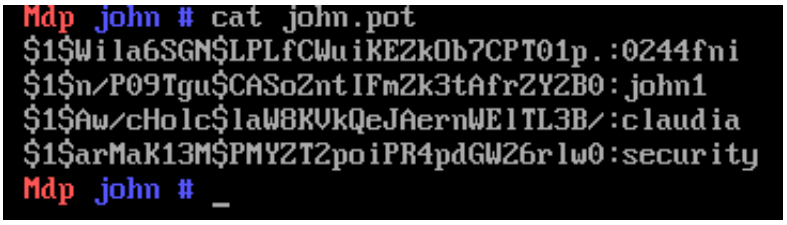

FIG. 26 – Résultat de l'attaque par john the ripper

2.5 Question 5 Contrôle de qualité de choix de mot de passe

a) On voit (cf 26) que john a trouvé 4 mots de passe lors d'une durée d'attaque d'approximement 15 minutes. Mais il avait trouvé les 4 premiers mots de passes très vite en réalité (environ 1 min).

John nous renvoit les hash (qui se trouvent dans etc/shadow) et le mot de passe en clair décrypté à coté. Pour trouver la correspondance il faut regarder dans etc/shadow à quel utilisateur le hash est associé.

John a trouvé les deux premiers mots de passe lors de sa phase 1, voici les combinaisons nom d'utilisateur/mot de passe de ceuw ci : inf4420/0244fni et john/john1.

On voit que le premier mot de passe est le nom d'utilisateur écrit à l'envers et le deuxième est le nom d'utilisateur plus un chiffre. On peut remarquer qu'en terme de caractères les mots de passes sont très proche des noms d'utilisateur. En effet dans la phase 1 john va tester des modifications simples appliquées au nom d'utilisateur : rotations, passage de caractères en majuscules, ajout de chiffres à la fin... puis en combinant ces transformations. Les possibilités ne sont pas très grandes ce qui fait que cette phase 1 est très rapide et efficace contre les mots de passe basiques.

Les deux derniers mots de passe (claudia et security) ont été trouvé lors de la phase 2, qui est la phase d'attaque par dictionnaire, c'est à dire que john va tester des mots connus (noms propres inclus) avec majuscule, chiffres... mais ici il n'en a même pas eu besoin. On peut imaginer que claudia est quelqu'un de l'entrourage de mark (l'utilisateur)

b) L'entropie de Shannon maximum correspondant à une répartition des caractères (fréquence/probabilité) également répartie :

- Pour [a-zA-Z], c'est donc $log_2(52)$ bits soit 5.7 bits/symbole
- Pour [a-zA-Z0-9], c'est donc $log_2(62)$ bits soit 5.95 bits/symbole
- La table ASCII contenant 128 caractères on a au maximum $log_2(128)$ soit 7 bits/symbole

c) Pour qu'un mot de passe soit fort il faut qu'il y ait le plus de caractère différent, nottament une variété de caractères ASCII qui augmente grandement les possibilités. On voit que l'utilisation des chiffres n'augmente pas de beauciup la complexité. On voit aussi qu'il faut tendre vers une répartition égale des caractères, éviter les répétitions.

d) On peut déduire 3 critères sur le mot de passe en analysant l'attaque de john :

- Il ne faut pas qu'il soit proche du nom d'utilisateur, il ne faut pas les même lettre dans l'idéal, même réarrangée même si cela forme un autre mot.

- Il ne faut pas pouvoir le trouver dans le dictionnaire, cela signifie pas de noms communs ni de noms propres.

- Il faut éviter d'utiliser des caractères communs, par exemple le e,r,s,t en français car john va dans l'étape 3 faire une attaque en se basant sur la fréquence d'apparition des caractères de la langue.

3 Déchiffrement de mot de passe

3.1 Question 1 échec de RSA

Une telle méthode de chiffrement présente plusieurs failles : tout d'adord, le message étant chiffré caractère par caractère, il s'agit d'un chiffrement par subtitution. De plus, ces derniers sont chiffrés avec la clé publique, qui est donc connue de tous même d'un attaquant. Aussi, lorsqu'un attaquant intercepte un message, il peut chiffrer tous les caractères de l'alphabet de la source, puis déchiffrer le texte en faisant la correspondance entre les caractères chiffrés du texte, et ceux de son alphabet chiffré. Précisons que cette attaque n'est possible que du fait qu'il n'y a que 26 lettres à chiffrer.

Nous allons maintenant mettre en pratique cette attaque sur le texte fourni pour le matricule 1815208. Le texte à déchiffrer est le suivant : 102647411606187071039064174 20 , de plus on a, $e = 349$ et $N = 496829$. Le message une fois décodé est : "unies".

Par ailleurs, les nombres 0 et 1 ne sont pas sensibles à l'élévation à la puissance, aussi quel que soit le couple (p,q) utilisé, ces nombres ne seront jamais modifié par le chiffrement, il serait donc souhaitable que ces deux nombres ne soient pas utilisés. Enfn, de manière générale, dans le but de se prémunir contre des attaques comme celle que nous avons menée, il est primordial que l'alphabet de la source soit le plus grand possible, on peut imager coder plusieurs caractères en même temps, ou encore coder un caractère à la fois mais le concaténer à une valeur aléatoire.

3.2 Question 2 déchiffrement simple

Nous avons déchiffré le texte associé au matricule 1815208, après une analyse fréquentielle nous obtenons :

s and commissioner of agriculture and public works and in quebec solicitor general or shall disqualify him to sit or vote in the house for which he is elected provided he is elected while holding such