Transmission de donnees

March 19, 2021

1 Cryptographie

Un intérêt de la transmission de donnée et d'essayer d'assurer sa confidentialité. Aujourd'hui les méthodes de chiffrement RSA permettent d'assurer la fiabilité des transmissions même sur des réseaux publiques comme le réseaux internet.

Nous allons étudier des méthodes de chiffrement antérieure à cette technique.

1.1 Encodage d'un texte

Afin d'appliquer des méthodes de cryptographie à des textes, nous allons devoir effectuer des opérations sur les caractères (lettres, accents, ponctuation, ...) d'un texte. Afin de faciliter la manipulation de ces calculs numériques on met en place une correspondance entre les caractères et des nombres entiers. Pour cela on utilisera la norme Unicode qui est déjà en place à l'aide de deux fonctions ord() et chr()

[24]:
```
chr(65)
```

[24]:
```
'A'
```

[25]:
```
ord('z')
```

[25]:
```
122
```

Ecrire une fonction qui convertie un texte vers une liste de nombre entier représentant ses caractères dans la norme Unicode, ainsi que la fonction réciproque.

[ ]:
```

```

Tester vos fonctions pour le texte 'azerty'

[ ]:
```

```

On remarque que certains nombre entier seuleument sont utilisés pour les lettres de l'alphabet. Déterminer quels nombres sont utilisé pour les lettres minuscules, majuscules, et les accents.

[ ]:
```

```

Dans les algorithmes de chiffrement mis en place on se contentera de ne chiffrer que les lettres sans distinction entre majuscule, minuscule, accents et on gardera tels quels les autres caractères: ponctuation, apostrophes, ...

Il faudra alors faire une distinction de cas selon les valeurs des codes rencontrés.

2 Chiffrement de César

Une première stratégie de cryptographie consiste à décaler toutes les lettre d'un texte. Par exemple dans un texte remplacer tous les a par des c, les b par des d, les c par des e, ...

Lorsque le a est remplacé par un c, on dit que la clef du chiffrement est le 'c'.

Ecrire une fonction chiffre de César qui prend en argument une clef et un texte et retourne le texte chiffré.

[ ]:

On pourra tester le chiffrement sur le texte "Toutes choses sont faites d'atomes, petites particules animées d'un mouvement incessant, qui s'attirent lorsqu'elles sont distantes les une des autres, mais se repoussent lorsqu'on les pousse à se serrer trop près." avec la clef "c".

[ ]:

Une fois le texte chiffré on remarque qu'il est illisible à moins d'avoir la clef. Ecrire une fonction decodage_de_Cesar() qui prend en argument la clef et le texte chiffré et retourne le texte en clair.

[ ]:

On pourra tester notre fonction sur le texte chiffré obtenu précédement avec la bonne clef et une mauvaise clef.

[ ]:

La confidentialité du message est donc garantie par la confidentialité de la clef. Un moyen de décoder le message est de trouver une stratégie pour deviner la clef utilisée.

Une stratégie efficace pour retrouver la clef et d'effectuer une analyse en fréquence. L'idée est de remarquer qu'en langue française certaines lettres apparaissent plus souvent que d'autres. Par exemple en moyenne la lettre "e" est la lettre la plus utilisée en français. Bien sur ceci peut varier d'un texte à l'autre selon l'auteur on peut penser notamment au roman de Georges Perec: La Disparition.

Ecrire une fonction qui compte le nombre d'occurence de chaque lettre de l'alphabet (sans distinction de majuscule ou d'accent) dans un texte.

[ ]:

Tester votre fonction sur le texte "Toutes choses sont faites d'atomes, petites particules animées d'un mouvement incessant, qui s'attirent lorsqu'elles sont distantes les une des autres, mais se repoussent lorsqu'on les pousse à se serrer trop près." . Que remarquez-vous, commenter ?

[ ]:

Si l'algorithme de chiffrement utilisé est un algorithme de César à décalage, on observera alors simplement un décalage de la lettre la plus utilisée.

Ecrire un programme qui montre cet effet et mettre en oeuvre un protocole pour deviner la clef lors d'un chiffrement de César.

[ ]:

2.1 Chiffrement de Vigenère

Une stratégie pour renforcer la sécurité du chiffrement de César est d'utiliser un mot de plusieurs lettre comme clef, par exemple la clef n'est plus "c" mais "feynman".

L'utilisation d'une clef à plusieurs caractère consiste à effectuer un chiffrement de César avec la clef "f" sur la première lettre, puis la clef "e" sur la seconde, puis la clef "y" sur la troisième, et ainsi de suite jusqu'à la fin du mot "feynman" où on remcommence à la clef "f".

Ecrire une fonction chiffre de Vigenère qui prend en argument une clef et un texte et retourne le texte chiffré.

[ ]:

Puis tester votre fonction toujours sur le même texte avec la clef "feynman".

[ ]:

Ecrire ensuite une fonction de décodage qui à partir de la clef et du texte chiffré donne le texte initial, puis testez là.

[ ]:

L'enjeu est à nouveau de trouver une technique pour deviner la clef à partir du texte chiffré seuleument. On peut commencer par essayer d'utiliser une analyse en fréquence.

Suivez la même procédure que précédement que remarquez vous ?

[ ]:

Une stratégie consiste à se ramener à la situation d'un chiffrement de César en commençeant par deviner la longueur de la clef. L'idée pour obtenir cette taille de clef et de remarquer que la langue française présente des répétitions de 3 lettres dans différents mots, par exemple les trois lettres "mes" dans mesquin, mesure, termes, ... Ces successions de trois lettres sont appelés trigrammes. Et on peut aussi retrouver dans le texte chiffré des trigrammes qui se répètent. Si tel est le cas c'est qu'ils ont été codé par le même trigramme de la clef. La probabilité que deux trigrammes du texte différents encodés par deux trigrammes de la clef différent donne les même trigrammes est faibles.

Quelle relation a-t-on alors entre la distance de répétition entre les trigrammes dans le texte chiffré et la longueur de la clef ?

[ ]:

Comment peut-on alors retrouver la longueur de la clef, à partir du repérage de multiple répétitions dans le texte ?

[ ]:

Une fois la longueur de la clef trouvée comment peut-on en déduire la clef ?

[ ]:

Pour mettre en place l'algorithme de recherche de la longueur de la clef, on ne travaille plus sur les caractères individuellement mais sur les trigrammes.

Ecrire une fonction qui prend en argument un texte et qui retourne une liste dont chaque élément est un nombre entier représentant un trigramme.

[ ]:

Tester votre fonction sur le texte chiffré par la méthode de Vigenère.

[ ]:

Ecrire une fonction qui prend en argument la liste des trigrammes et retourne une liste de toutes les distances entre répétition de trigramme.

[ ]:

Ecrire une fonction qui prend en argument la liste des distances entre les répétitions et en déduit la longueur de la clef.

[ ]:

Testez vos algorithme de détermination de la longueur de clef sur le texte fournit et chiffré par la clef “feynman”.

[ ]:

2.2 Chiffrement de Vernam

Enfin pour éviter les attaques sur le chiffrement de Vigenère, une stratégie est de choisir une clef de même longueur que le texte à chiffrer et pour éviter toute répétition dans la clef de tirer au sort tous les caractères de la clef.

Mettre en place ce chiffrement et montrer que l’on ne peut pas deviner la clef à partir du message chiffré seuleument.

[ ]:

Quel problème voyez vous à l’utilisation d’un chiffrement de Vernam sur un réseau publique comme le réseau internet ?

Recherchez la solution apportée par le chiffrement RSA.

3 Transmissions fiables de données et codes correcteurs d’erreur

Lors de la transmission d’information à travers un réseau, des phénomènes physiques aléatoires peuvent modifier le message et générer des erreurs dans le message initialement transmis. Pour assurer le bon fonctionnement des transmissions d’information numérique, il faut être capable de détecter ces erreurs et si possible de les corriger automatiquement.

3.1 Transmission de Bit d’information et bit de parité

La transmission d’information numérique peut toujours se ramener à l’étude de la transmission de code binaire. En effet une information numérique est quantifié sur un nombre fini de niveaux de quantification que l’on peut compter en base 2.

On va dans cette partie étudier la transmission d’un texte :

[17]:
```
texte = "Toutes choses sont faites d'atomes, petites particules animées d'un␣
 ↪mouvement incessant, qui s'attirent lorsqu'elles sont distantes les une des␣
 ↪autres, mais se repoussent lorsqu'on les pousse à se serrer trop près."
print(texte)
```

```
Toutes choses sont faites d'atomes, petites particules animées d'un mouvement
incessant, qui s'attirent lorsqu'elles sont distantes les une des autres, mais
se repoussent lorsqu'on les pousse à se serrer trop près.
```

Ce texte peut être codé comme une liste d’entier en base 10 grace à la correspondance établie par la norme Unicode. Et les fonctions ord() et chr() en python réalisent cette correspondance.

[18]:
```
print(chr(65))

print(ord('A'))
```

```
A
65
```

Enfin on peut transformer un nombre en base 10 en nombre binaire à l'aide de la fonction bin().

[20]:
```
bin(66)
```

[20]: `'0b1000010'`

On remarque que cette fonction prend en argument un entier en base 10 et ressort le code binaire correspondant sous le format d'une chaîne de caractère: la fonction type renvoie -> str.

[22]:
```
type(bin(66))
```

[22]: `str`

Cette chaîne de caractère est précédée du préfixe '0b' pour tout les codes binaires (il indique qu'il s'agit d'un code en base 2) puis des caractères '1' ou '0' selon la valeur du bit en débutant par le bit de poids le plus fort jusqu'au bit de poids le plus faible.

[23]:
```
print(bin(0))
print(bin(1))
print(bin(2))
```

```
0b0
0b1
0b10
```

Afin de faciliter la manipulation et la visualisation des effets des programmes sur le texte écrire une fonction qui prends en argument une chaine de caractère texte, et qui renvoie une liste contenant les codes binaires de chaque caractère. Ces codes binaires de chaque caractère seront eux-même encodé sous forme d'une liste de nombre entier 1 ou 0 correspondant aux valeurs des bits du code binaire.

[ ]:
```

```

Pour s'assurer de la transmission d'un message sans erreur une stratégie consiste à rajouter une information redondante dans le message. En effet on peut vérifier après transmission que cette répétition est toujours cohérente et donc qu'il n'y a pas eu de modification aléatoire d'une partie du message.

Une information redondante que l'on ajoute de manière usuelle à un code binaire est le bit de parité. Par définition ce bit de parité est : - égal à 0, si le code contient un nombre pair de 1 (donc si les bits sont de somme paire) - égal à 1, si le code contient un nombre impair de 1 (donc si les bits sont de somme impaire)

Calculer à la main le bit de parité pour des entiers 3, 16, 255

Ecrire une fonction parite() qui prend en argument un code binaire sous forme de liste de nombre entier 1 ou 0 et retourne la valeur du bit de parité comme l'entier 0 ou 1.

[ ]:
```

```

3.2 Somme de contrôle (checksums)

Les sommes de contrôle sont des stratégies pour vérifier si le message est transmis sans altération du message ou si le message doit être renvoyé. Le protocole consiste avant émission de calculer la somme des bits présent dans le code binaire, puis de transmettre le code et sa somme en même temps. Après réception du message le destinataire recalcule de son côté la somme des bits du code est vérifie qu'il obtient bien un résultat en accord avec la somme transmise avec le message.

Expliquer pourquoi le bit de parité constitue une somme de contôle.

Ecrire une fonction controle_de_parite() qui prend en argument un code binaire et un bit de parité puis vérifie si le message à été altéré ou pas.

[ ]:

Tester votre fonction à l'aide du texte fournit, du calcul du bit de parité, et d'un tirage aléatoire pour savoir si un bit est modifié ou pas.

[ ]:

Quelle est la limite de cette technique de somme de contrôle ? Donner un exemple de mauvaise transmission qui n'est pas détectable par la technique du bit de parité. Puis montrer numériquement l'erreur qui survient.

[ ]:

Quelle stratégie permet de limiter l'apparition de cette erreur indétectable ?

3.3 Code de Hamming

Le code de Hamming est un autre protocole d'utilisation des bits de parité pour détecter les erreurs mais aussi pour les corriger automatiquement sans avoir besoin de retransmettre le message.

Un exemple de code de Hamming est le code dit (7,4), qui consiste à envoyer tout les 7 bits : 4 bits de données du message (d_1, d_2, d_3, d_4) et 3 bits de parité (p_1, p_2, p_3). Les bits de parité sont définis comme: - p_1 est le bit de parité de (d_1, d_2, d_4) - p_2 est le bit de parité de (d_1, d_3, d_4) - p_3 est le bit de parité de (d_2, d_3, d_4)

Ecrire une fonction encode_hamming() qui prend en argument une liste de 4 bit de donnée et retourne une liste des 7 bits de donnée et de parité.

[ ]:

Le contrôle du message après réception peut dans un premier temps consister à recalculer les 3 bits de parité et vérifier s'il n'y a pas eut d'altération. Mais la technique porposée par Hamming consiste à calculer 3 autres bits de parités. A partir du message $(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (p_1, p_2, p_3, d_1, d_2, d_3, d_4)$ on calcule : - c_1 le bit de parité de (m_4, m_5, m_6, m_7) - c_2 le bit de parité de (m_2, m_3, m_6, m_7) - c_3 le bit de parité de (m_1, m_3, m_5, m_7)

On peut montrer alors que si le message a bien été encodé et transmis sans altération alors les trois bits de contrôle doivent être égal à 0. Si ce n'est pas le cas alors il y a une erreur de transmission.

Verifier cette affirmation en calculant numériquement les bits de parité c en fonction d'un message m non altéré ou altéré.

[ ]:

Dans le cas d'une erreur unique on remarque non seuleument que les bits de contrôle c sont différents de $(0, 0, 0)$ mais qu'ils indiquent aussi en binaire la position du bit affecté par l'erreur.

Par exemple si $c = (0,1,1)$ alors l'erreur porte sur le troisième bit du message. On peut alors corriger automatiquement l'erreur.

Ecrire une fonction decode_hamming() qui prend en argument une liste de 7 bits et retourne une liste de 4 bit de données décodés. En cas d'erreur, on renverra un message d'avertissement indiquant la position du bit affecté et on effectuera la correction.

[ ]:

Déterminer le codage de Hamming de la donnée 1011, puis la donnée décodée par l'algorithme dans l'hypothèse où les deux premiers bits du message codé ont été incorrectement transmis. Quel a été l'effet de la correction automatique sur les bits de donnée dans ce cas ?

[ ]:

Proposer une méthode simple de différencier une double erreur d'une erreur unique au moyen d'un bit de parité supplémentaire et expliquer comment cela permet d'éviter le problème mis en évidence à la question précédente. On ne cherchera pas à corriger la double erreur, mais juste à la détecter.

[ ]: