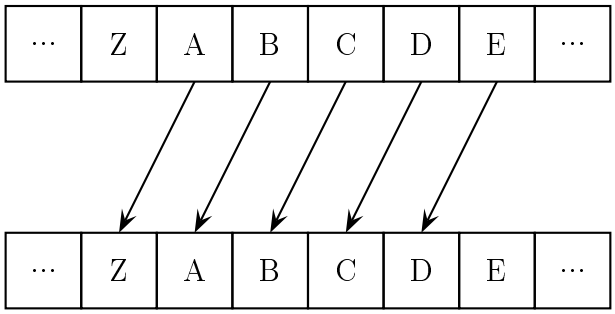
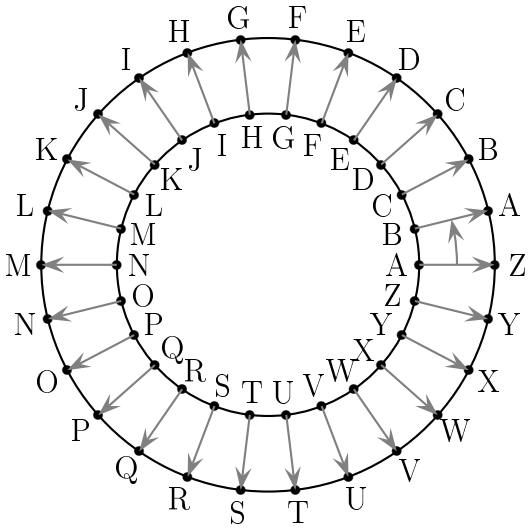
TP : Sécurisations des communications

1 Chiffrement de César

Le chiffrement de César, nommé ainsi car utilisé par César dans ces correspondances, est une méthode de chiffrement symétrique consistant à créer un décalage, ou plus rigoureusement une rotation, dans l'alphabet. Il s'agit d'une substitution monoalphabétique : à chaque lettre en correspond une et une seule autre.





Par exemple, la phrase « Il y a un serpent dans ma botte » peut ainsi être chiffrée sous la forme

Hk x z tm rdqodms czmr lz anssd.

La clé de cette méthode réside dans les deux lettres qui définissent ce décalage. Dans l'exemple ci-dessus, la clé est B→A, autrement dit, on a un décalage de -1 ou de 1 vers la gauche dans l'alphabet. Pour déchiffrer le message, il suffit d'effectuer le décalage inverse : A→B.

Exercice 1. [Chiffre de César]

1. Programmer une fonction permettant de chiffrer et déchiffrer un message selon la méthode chiffre de César.
2. Programmer une fonction attaquant un chiffrement de César :
   (a) par force brute ;
   (b) par analyse fréquentielle.

Indications : `ord(caractère)` permet d'obtenir le numéro d'un caractère en unicode. `chr(numéro)` permet d'obtenir un caractère à partir de son numéro en unicode.

— Les numéros des lettres majuscules vont de 65 (pour A) à 90 (pour Z).
— Les numéros des lettres minuscules vont de 97 (pour a) à 122 (pour z).
— La plupart des lettres accentuées ont leur numéro compris entre 192 et 256.

Exercice 2. Déchiffrer le texte suivant.

Fmzue : Ì fage xqe sanqgde pq yagotqe, aghdql sdmzp hae adquxxqe ! V'mu fdaghõ gzq zaghqxxq mdotq ! Oayyq hage xq emhql bqgf-öfdq, x'aghqdfgdq pq xm bdqyuôdq mdotq ux k m ouzc mze m põoxqzotõ gzq dõmofuaz qz otmúzq cgu qz m dõhõxõ p'mgfdqe...
Yd Fadsgq : Az e'qz ragf ! ! ! Fg f'qz ruotqe pq fagf óm, otmeeqgd pq x'mdotq ? ! Fau, fg hqgj pg ngfuz, pqe sdae bqoe ! Pqe qjbxaeuaze ! Yau, o'qef Fadsgq qf vq egue xì bagd fq baeqd gzq qf gzq eqgxq cgqefuaz : PQE QJBXAEUAZE ? ? ?

2 Chiffrement par permutation

Permutation

Une permutation σ d'un ensemble E dans E est une fonction vérifiant la propriété de bijectivité de E dans E :

$$\forall y \in E,\ \exists ! x \in E\ :\ y = \sigma(x).$$

Autrement dit, pour tout élément y de l'ensemble E, il existe un unique élément x de E tel que $y = \sigma(x)$.

Cette correspondance peut être représentée à l'aide d'un tableau comme dans l'exemple ci-dessous à gauche qui représente une permutation de $E = \{1, 2, 3, 4\}$. L'exemple ci-dessous à droite ne représente pas une permutation car 4 ne possède pas un unique antécédent mais deux alors que 1 n'en a pas.

1	2	3	4
3	1	2	4

1	2	3	4
3	4	2	4

Une permutation se représentant donc sous la forme d'un tableau, on peut y associer une matrice. La matrice associée à l'exemple précédent est

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix}$$

Principe du chiffrement par permutation

Les permutations permettent de chiffrer du texte. Concentrons nous sur l'alphabet et numérotons ses lettres de la façon suivante :

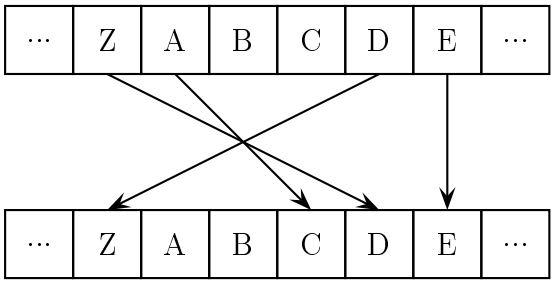
Lettre	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Numéro	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

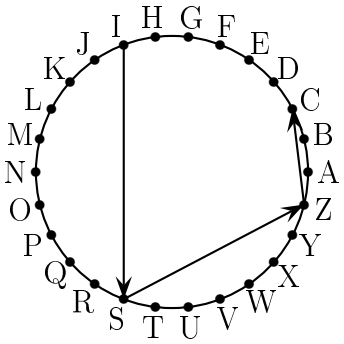
Lettre	U	V	W	X	Y	Z
Numéro	20	21	22	23	24	25

On peut alors effectuer une permutation de celui-ci à l'aide d'une matrice de permutation. Les permutations ont deux grands avantages :

— elles deviennent rapidement très nombreuses lorsque que la taille de l'ensemble considéré augmente (confère exercice ci-dessous), ce qui invalide l'attaque par force brute ;

— elles ne possèdent pas nécessairement de « schéma de fabrication » – i.e. de formule(s) – permettant de déterminer leur fonctionnement puisque celles-ci peuvent être créées aléatoirement.





Remarque : dans les deux illustrations ci-dessus, tous les éléments n'ont pas d'image par soucis de lisibilité. Cependant, c'est bel et bien le cas pour une vraie permutation.

Le chiffrement de César est un cas de particulier de permutation. En effet, les rotations peuvent être vues comme des cas particuliers de permutations. Si on note $d \in [\![-25\,;25]\!]$ le décalage du chiffrement de César, ce dernier peut s'écrire grâce à la permutation

$$\sigma(x) = (x+d)[26] = \begin{cases} x+d & \text{si } x+d \in [\![0\,;25]\!]\,, \\ x+d-26 & \text{si } x+d \geq 26, \\ x+d+26 & \text{si } x+d \leq 0, \end{cases}$$

pour $x \in [\![0\,;25]\!]$.

La matrice représentant la permutation associée au chiffrement de César donné dans l'exemple précédent est :

$$\begin{pmatrix} 0 & 1 & 2 & \ldots & 24 & 25 \\ 25 & 0 & 1 & \ldots & 23 & 24 \end{pmatrix}$$

Le chiffrement par permutation est donc comme le chiffrement de César un chiffrement symétrique – car il suffit d'inverser la matrice de permutation pour retrouver le message en clair – par substitution monoalphabétique : à chaque lettre correspond une et une seule lettre.

Exercice 3. [Permutations]

1. Écrire les matrices des permutations de taille deux et trois.
2. Combien de permutation de l'alphabet existe-t-il ?
3. Combien de permutations d'un ensemble de taille n existe-t-il ?
4. Avec un processeur cadencé à 1 GHz, combien de temps faudrait-il pour essayer toutes ces possibilités ?

Exercice 4. [Chiffrement par permutations]

1. Programmer une fonction générant aléatoirement une matrice de permutation.
2. Programmer une fonction chiffrant et déchiffrant un message à l'aide d'une matrice de permutation.

Exercice 5. [Peux-tu seulement le comprendre ? !] Déchiffrer le texte suivant :

Zv wigj dgt yt jt uifvt. Yt megvizb bzquvtqtoj jt czft dgt, dgt jg o'ib uib vt cfezj, jg o'ib uib vt cfezj ct q'izqtf aeqqt jg wizb. Jg o'ib uib vt cfezj ct q'tjft wzctvt. Ji wzctvzjt uefjt vt batig ct jeo tnezbqt, uifat dgt qez, at dgt yt mtgp, a'tbj yt mtgp kzto dgt jg q'izqtb qizb biob iqegf. Yt mtgp kzto mzmft imta jez qizb btuiftqtoj. Yt mtgp kzto imezf go towioj dgz jt ftbbtqkvt qizb uib ct jez ! Qizb çi utgp-jg btgvtqtoj vt aequftocft ? ! Utgp-jg btgvtqtoj vt aequftocft ? ! Utgp-jg btgvtqtoj vt aequftocft ? !

Exercice 6. [Mr Torgue] Déchiffrez le texte ci-dessous afin d'en savoir plus sur Mr Torgue.

xjjub ://regjg.kt/pE4bTSO2zn8 ?bz=wLWxKNaefRPHP1M9

3 Chiffrement de Vigenère

Le chiffrement de Vigenère est une méthode de chiffrement symétrique par substitution polyalphabétique : une lettre n'est pas substituée par une seule mais par plusieurs.

Bien que nommé d'après le diplomate du XVI^{e} siècle Blaise de Vigenère qui l'a décrit dans un traité paru 1586, on trouve des méthodes de chiffrement basées sur des procédés analogues qui lui sont antérieures.

Les méthodes de substitution polyalphabétique pour grand avantage de rendre l'analyse fréquentielle très difficile. En effet, dans un chiffrement monoalphabétique par permutation, si tous les E deviennent des B et si le message est assez long, alors on devrait retrouver dans celui-ci un pourcentage d'apparition du B proche de celui du E dans la langue utilisée ; ce qui permet d'en déduire que le E a probablement été substitué par B. Cela n'est pas possible avec une substitution polyalphabétique. En effet, dans notre exemple du E, celui-ci pourrait tantôt être remplacé par un B, tantôt par K, tantôt par D, etc.

Ces méthodes ne sont toutefois pas inviolables. Le chiffrement de Vigenère a été percé au XIX^{e} siècle et de manière générale, il est possible de s'attaquer à ces méthodes de chiffrement par analyse fréquentielle à condition de connaître la taille de la clé de chiffrement ; sur laquelle il est possible

d'obtenir des informations grâce à l'indice de coïncidence, outil mathématique développé au début du XX^{e} siècle.

Principe

Le principe du chiffrement de Vigenère est celui du chiffre de César appliqué avec une clé – autrement dit un décalage – différente pour chaque lettre du message à chiffrer. Chiffrons la phrase « Peux-tu seulement le comprendre ? ! » sans se préoccuper de la ponctuation. Pour cela, on se base sur un mot ou une phrase clé. Ce mot ou cette phrase est répété de façon à atteindre la longueur du message à chiffrer tout en coïncidant avec chaque lettre de celui-ci ; prenons comme clé « inconnus ». On obtient alors

Peux-tu seulement le comprendre ? !
Inco-nn usinconnu si nconnusinc ? !

La table de Vigenère indique comment est chiffrée chaque lettre à partir de la lettre clé qui lui est associée. Par exemple, le P devient un X lorsque que la lettre clé qui lui est associée est I comme dans l'exemple ci-dessus. Il s'agit en réalité d'un chiffrement de César appliqué à chaque lettre avec pour décalage A $\rightarrow$ « lettre de la clé » :

X = chiffre_cesar(P, A, I).

Ainsi, après chiffrement de Vigenère, la phrase « Peux-tu seulement le comprendre ? ! » est devenue

Xrwl-gh mwcygaran dm pqaceyfleg ? !

On peut observer que la lettre e, par exemple, a été chiffrée de plusieurs façons différente, ce qui rend l'analyse fréquentielle difficile, voire impossible si le texte chiffré est trop court.

Pour déchiffrer, il suffit d'appliquer le chiffrement de Vigenère en inversant les rôles de A et de la lettre de la clé, ce qui donne par exemple :

P = chiffre_cesar(X, I, A).

Exercice 7. [Chiffre de Vigenère] Programmer une fonction effectuant un chiffrement et un déchiffrement de Vigenère. On ne considérera pas les lettres accentuées afin de simplifier dans un premier temps.

Exercice 8. [Le vieil oncle Philibert]

1. Chiffrer à l'aide de votre fonction de chiffrement de Vigenère et de la clé « inconnus » la phrase

   « Je crois que je suis amoureuse du vieil oncle Philibert ».

2. Les liens suivants ont été chiffrés avec comme clé le chiffrement de « Philibert » de la question précédente. Déchiffrer ce texte afin de savoir.

   fstwx ://vwqns.ae/avas0g_O8Xf ?sp=4GjxN515hZgd43-E
   fstwx ://vwqns.ae/a1_exXbq-Ub ?sp=STAAgiyk3PJqSd2N

4 Masque jetable

Le masque jetable est une méthode de chiffrement symétrique inventée fin du XIX^{e}, début du XX^{e}. Bien qu'ayant un principe similaire au chiffrement de Vigenère, le masque jetable est théoriquement inviolable, comme l'a démontré Claude Shannon en 1949, sous les trois conditions suivantes :

— La clé de chiffrement, ou « masque », est de longueur supérieure ou égale à celle du message à chiffrer.
— La clé de chiffrement a ses caractères choisis de façon aléatoire.
— La clé n'est utilisée qu'une et une seule fois.

Cette théorique inviolabilité est la raison pour laquelle il fut utilisé aussi bien pour les communications entre le Kremlin et la Maison Blanche qu'entre Che Guevera et Fidel Castro. Toutefois, il n'est pas adéquat pour sécuriser les échanges sur Internet. En effet, le masque devant être plus long que le message à chiffrer, si on a un canal suffisamment sûr pour échanger les clés dessus, autant échanger directement les messages plutôt que les clés, cela sera plus court. Il est donc incompatible avec l'architecture cryptographie asymétrique puis symétrique couramment utilisée sur Internet comme le protocole HTTPS.

Principe

Le principe du masque jetable exceptées les conditions données au dessus est identique au chiffrement de Vigenère. Chiffrons par exemple la phrase « Qui a laissé le frigo ouvert ? ». Pour cela, on utilise la clé générée aléatoirement (en tapant au hasard sur le clavier, méthode non viable pour de réels messages) :

bdjjfbjqfbgitnqnlehoirbknghiomjrklbjrgvjkebjtiohzl

Comme pour le chiffrement de Vigenère, on fait correspondre à chaque caractère du message un caractère de la clé.

Qui a laissé le frigo ouvert ?
Bdj j fbjqfb gi tnqnl ehoirb ?

Il suffit de chiffrer en utilisant le même principe que dans le chiffrement de Vigenère : chaque caractère subit un chiffrement de César dont le décalage est donné par la lettre correspondante de la clé par rapport à A. Par exemple, on a

R = chiffre_cesar(Q, A, B).

En itérant sur l'ensemble du message et de la clé, on obtient alors

Rxr j qbrixf rm yeytz sbjmiu ?

Comme pour le chiffrement de Vigenère, le déchiffrement s'effectue en réalisant le décalage inverse lors du parcours du message et de la clé. On obtient par exemple le Q en faisant

Q = chiffre_cesar(R, B, A).

Exercice 9. [Vers l'infini et au delà !]

1. Combien de mots possibles peuvent correspondre au déchiffrement d'un mot de deux lettres chiffré avec une clé aléatoire ? De trois et quatre lettres ?
2. Combien de possibilités de déchiffrement a-t-on pour un message de longueur n quelconque chiffré avec une clé aléatoire ?

Méthode informatisée

Sous forme informatisées, données et clé se retrouvent sous forme binaire. Le chiffrement et le déchiffrement s'effectuent alors simplement grâce à l'opération XOR : si A représente les données et B la clé, alors le chiffré de A par B est $C = A \oplus B$. Il s'agit d'une opération très simple à réaliser et ses propriétés permettent également le déchiffrement. En effet, XOR vérifie

1. $(A \oplus B) \oplus C = A \oplus (B \oplus C)$ (associativité) ;
2. $(A \oplus A) = 0$;
3. $(A \oplus 0) = A$.

Avec ses trois propriétés, il est aisé de montrer que $A = B \oplus C$.

Exercice 10. [XOR]

1. Démontrer les trois propriétés de XOR énoncées ci-dessus.
2. Montrer que, si A représente les données et B la clé, C le chiffré de A par B : $C = A \oplus B$, alors $A = B \oplus C$.

Inviolabilité

Comme dit plus haut, les trois conditions suivantes garantissent l'inviolabilité du masque jetable.

1. La clé de chiffrement, ou « masque », est de longueur supérieure ou égale à celle du message à chiffrer.
2. La clé de chiffrement a ses caractères choisis de façon aléatoire.
3. La clé n'est utilisée qu'une et une seule fois.

Le premier point garantit que des attaques fréquentielles ne fonctionnent pas, comme cela peut être le cas avec le chiffrement de Vigenère ; en effet, les attaques de ce dernier reposent la déduction de la longueur de la clé et à partir de là des analyses fréquentielles ; ce qui n'est pas possible si la clé est plus longue que le message à chiffrer.

Le second point garanti ce qu'on appelle une sécurité sémantique. Comme les caractères de la clé sont choisis de façon aléatoire, indépendante, équiprobable, toutes les clés sont aussi probables les unes que les autres. Ainsi, sur une attaque par force brute, on obtiendrait toutes les possibilités et parmi elles, nombreuses seraient celles qui auraient du sens. Par exemple, une attaque par force brute sur « bjcae ncavso » ferait ressortir les possibilités suivantes

raton laveur
balai volant
koala gentil
porte unique
terra matter

Toutes ces possibilités font sens. Et pourtant la liste n'est pas exhaustive. Laquelle choisir ? Il est impossible de le faire, c'est ce qu'on appelle la sécurité sémantique.

La question de l'aléatoire pour générer la clé est primordiale. Une clé générée de façon pseudo aléatoire peut s'avérer insuffisante face à la cryptanalyse. L'idéal est donc de l'aléatoire parfait généré à partir de phénomènes physiques.

Le troisième point est essentiel. Si la clé est utilisée plusieurs fois, il y a un risque de voir les données déchiffrées sans même utiliser la clé. Il est facile de s'en convaincre avec le chiffrement par XOR. Considérons deux jeux de données sous forme binaires B_1 et B_2 chiffrés respectivement en $\overline{B_1}$ et $\overline{B_2}$ à l'aide d'une même clé C. On a alors, par propriétés de XOR,

$$\begin{aligned}\overline{B_1} \oplus \overline{B_2} &= (B_1 \oplus C) \oplus (B_2 \oplus C) \\ &= B_1 \oplus B_2 \oplus C \oplus C \\ &= B_1 \oplus B_2 \oplus 0 \\ &= B_1 \oplus B_2.\end{aligned}$$

Si maintenant un des deux jeux données en binaire est connu en clair, alors il est possible de retrouver l'autre sans clé. En effet, disons que l'on connaît par exemple B_2, alors pour retrouver B_1, il suffit de faire :

$$B_1 = \overline{B_1} \oplus \overline{B_2} \oplus B_2.$$

En réalité, sans même connaître l'un des deux jeux de données en clair, il est possible par des moyens plus sophistiqués de déchiffrer si la même clé est utilisée plusieurs fois.

Exercice 11. [Fausse clé] Écrire une fonction générant une clé de chiffrement à partir d'un texte chiffré afin qu'il devienne le texte que vous souhaitez lors du déchiffrement.

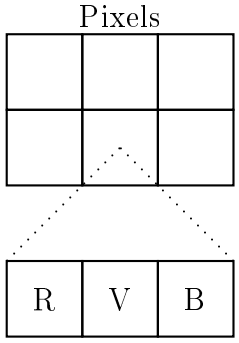
Exercice 12. [Qui a laissé le frigo ouvert ?]

— 16, 46, 60, 100, 43, 124, 40, 44, 37, 55, 57, 33, 117, 46, 63, 110, 55, 43, 62, 54, 32, 119, 46, 51, 61, 63, 40, 60, 121, 116.
— 41, 47, 33, 52, 57, 102, 107, 98, 53, 43, 63, 48, 32, 108, 56, 43, 126, 104, 48, 35, 58, 28, 47, 0, 44, 17, 98, 5.

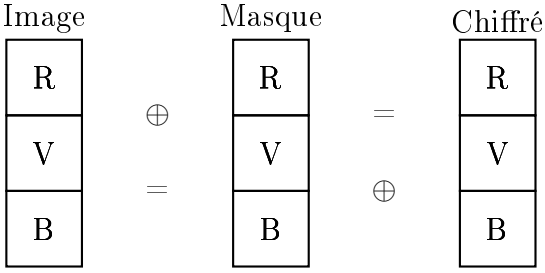
5 Images

Chiffrement d'une image par masque jetable

Une image est composée de pixels. Chaque pixel est composé de trois niveaux des couleurs rouge, vert et bleu dans le standard RGB. Chacun de ces niveaux de couleurs est un nombre codé sur un octet, i.e. une valeur entre 0 et 255.



Il est ainsi possible de chiffrer une image comme on chiffre un texte : à l'aide d'un chiffrement XOR et d'un masque jetable. Le masque jetable peut être une autre image mais pas nécessairement. Dans le cas d'une image, on récupère les niveaux de rouge, de vert et de bleu de chaque pixel de l'image et du masque pour créer une nouvelle image dont les pixels seront le résultat de la fonction XOR appliquée à chacun de ces niveaux.



Comme l'opération XOR vérifie la relation $A = C \oplus B$ si on a $C = A \oplus B$, il est aisé de déchiffrer l'image en réappliquant le masque à l'image chiffrée.

Exercice 13. [Conditions sur le masque]

1. Quelles sont les conditions à respecter par le masque ? Comment cela se traduit-il en terme d'image ?
2. Si le masque n'était pas une image mais un texte par exemple, quelle devrait être sa taille minimale ?

En Python, la bibliothèque PIL permet de manipuler des images, notamment par l'intermédiaire des pixels. Les commandes ci-dessous nous permettront de réaliser notre chiffrement.

```
import PIL
from PIL import Image

# Ouvrir, sauvegarder, afficher une image
image = Image.open("image.extension")
image.save("nom_image.extension","extension")
image.show()

# Récupération de la taille d'une image sous la forme d'un tuple (largeur, hauteur)
taille = image.size

# Création d'une image RGB intégralement noire
image = Image.new("RGB", (largeur, hauteur), (0,0,0))

# Récupération des valeurs RGB d'un pixel sous la forme d'un tuple (r, g, b)
pixel = image.getpixel(coordoonnées_pixel)

# Mise à jour d'un pixel d'une image
image.putpixel(coordoonnées_pixel, pixel)
```

Exercice 14. [Masque jetable d'une image]

1. Programmer une fonction créant une image aléatoire de taille voulue.
2. Programmer une fonction de chiffrement par masque jetable pour image.
3. Déchiffrer l'image secrète présente dans le dossier Sécurisation des Données à l'aide de l'image clé.
4. Que se passe-t-il si vous chiffrer une image avec un masque qui n'a pas été généré aléatoirement ?

Chiffrement par permutation

Il est possible de chiffrer une image en permutant l'ensemble des pixels qui la compose, i.e. d'échanger les places de tous les pixels.

Exercice 15. [Chiffrement par permutation d'une image]

1. Combien de permutations existent-ils de l'ensemble des pixels d'une image ?
2. Avec un processeur cadencé à 1 GHz, combien de temps faudrait-il pour essayer toutes ces possibilités ?
3. Qu'est-ce que le déchiffrement d'une image chiffrée par permutation implique ?
4. Programmer une fonction chiffrant et déchiffrant une image par permutation. *Indication :* il est possible d'éviter de passer par un tableau de permutations en utilisant une permutation simple, comment ?

Stéganographie

La stéganographie est un domaine où l'on cherche à dissimuler une information au sein d'un média qui sert de masque. Cela peut par exemple être un texte ou une image cachés dans une autre image, d'une piste sonore etc. Contrairement à la cryptographie dont l'objectif est de rendre indéchiffrable une information pour ceux à qui elle n'est pas destinée, la stéganographie ne chiffre pas forcément les informations, elle se contente de les cacher dans d'autres informations ; même s'il est possible de combiner les deux méthodes. Toutefois, tout comme la cryptographie, la stéganographie ne se limite pas au monde informatique et était utilisée avant son invention.

On pourra enfin remarquer que la stéganographie n'est pas nécessairement bienveillante. En effet, les chevaux de Troie, logiciels malveillants cachés au sein d'un logiciel tiers, constituent une forme de stéganographie.

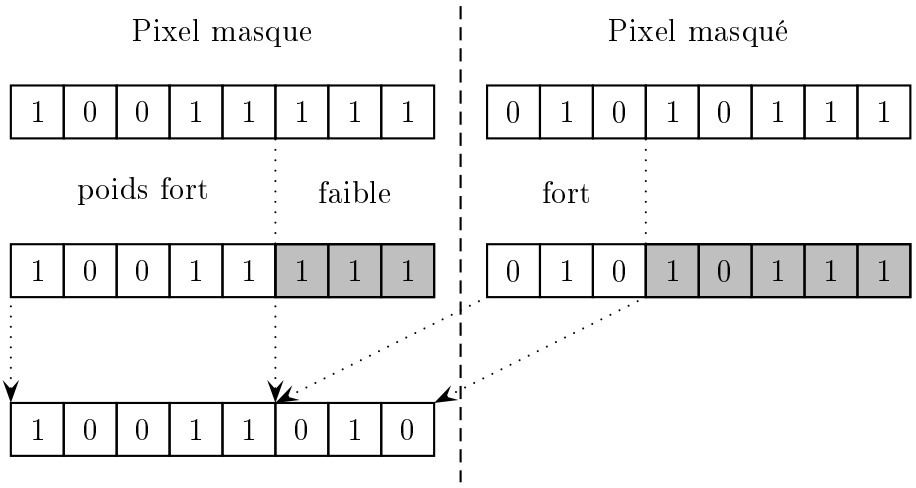
Pour dissimuler une image dans une autre, on travaille comme pour le chiffrement sur les pixels de nos images. Considérons deux images RVB (le principe est le même avec d'autres formats comme le bitmap, etc), chaque pixel de ces deux images contient une valeur de rouge, de vert et de bleu codée sur un octet, soit huit bits. Pour simplifier, considérons uniquement le niveau de rouge d'un pixel de chaque image, par exemple :

Pixel de l'image	1	2
Niveau de Rouge décimal	159	87
Niveau de Rouge décimal	10011111	01010111

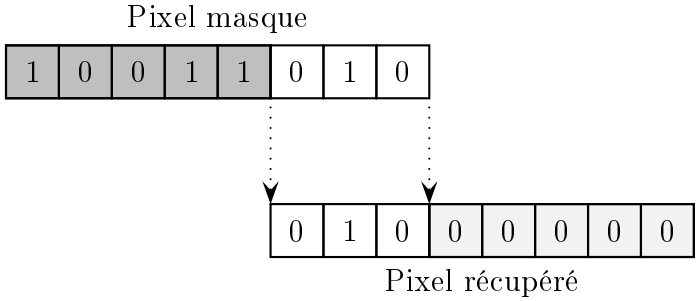
Dans un octet, tous les bits n'ont pas le même poids : le dernier ne change la valeur que de 1, les deux derniers réunis, que de 3, les trois derniers réunis, que de 7, etc. Ainsi, changer la valeur de ces bits ne change que très peu le niveau de rouge. Par exemple, la différence entre 10011000 et 10011111 est seulement de 7 ; différence potentiellement imperceptible à l'œil nu. Bien évidemment, plus l'on change de bits, plus la différence devient visible. On parle de bits de poids faibles pour ceux n'ayant que peu d'impact sur l'image et de bits de poids forts pour ceux ayant un fort impact sur l'image. L'idée de stéganographie d'image informatisée est de remplacer les bits de poids faibles de l'image masque par les bits de poids forts de l'image à masquer.

Masquons par exemple le pixel de l'image 2 dans celui de l'image 1 en reprenant les valeurs données dans le tableau ci-dessus. Pour cela, disons qu'on ne s'autorise qu'à modifier les trois des derniers bits du pixel de masquage : 10011111 ; remplaçons les par les trois plus forts du pixel 2 : 010 ; on obtient alors un nouveau niveau de rouge : 10011010=154.

La différence entre le niveau de rouge 159 et 154 est à peine perceptible, voire imperceptible à l'œil nu. D'autant plus que ce niveau de rouge est aussi mélangé à des niveaux de vert et bleu.



Pour récupérer le niveau de rouge du pixel l'image masquée, il suffit de récupérer les trois derniers bits de celui de l'image masque et de les compléter par 0. Dans le cas de l'exemple précédent, on récupère 010 que l'on complète en 01000000. On remarque que l'on récupère seulement une partie des bits de poids forts de l'image masquée, cette dernière est donc dégradée par rapport à l'originale.



Exercice 16. [Stéganographie] Programmer des fonctions masquant et récupérant des images au format RVB par stéganographie. Le nombre de bits attribués aux masquages devra être un des paramètres de ces fonctions.

Exercice 17. Ik ZV iutzoktz atk lot iginïk. Otjoik : iutyojïxkx rg bgrkax jk rg irï jk inollxksktz.

6 Ressources supplémentaires

— Cours sur le traitement d'images simples et la stéganographie.
— Un autre cours sur la manipulation d'images et la stéganographie en Python.
— Vidéo sur les preuves à divulgation nulle de connaissance de Passe Sciences.
— Vidéo sur l'algorithme de Diffie-Hellman de Science Étonnante.