

La **stéganographie** est l'art de dissimuler un message dans un autre message. En utilisant un algorithme simple de tatouage numérique ou **watermarking**, vous allez apprendre à travers cet exercice à dissimuler une image dans une autre image, puis à retrouver l'image dissimulée.

Brève histoire de la stéganographie

L'apparition de la stéganographie est très ancienne. La première trace écrite se trouve dans les *Histoires* de Hérodote, parues vers 445 av J.-C, à travers deux récits :

- le premier relate l'histoire d'Histiée, qui incite Aristagoras à se révolter contre son roi Darius. Pour ce faire, il eut l'idée de raser la tête de son esclave le plus fidèle, de lui tatouer son message sur le crâne et d'attendre que les cheveux repoussent avant de l'envoyer pour Milet, avec pour consigne de se faire raser les cheveux ;
- le second relate l'histoire de Demarate, ancien roi de Sparte réfugié auprès du roi des Perses, Xerxès Ier, qui a succédé à Darius. Demarate fut mis au courant d'un projet d'invasion de la Grèce. Il décida alors de prévenir Sparte en toute discrétion en utilisant le stratagème suivant : "*Il prit une tablette double, en gratta la cire, puis écrivit sur le bois même les projets de Xerxès ; ensuite il recouvrit de cire son message : ainsi le porteur d'une tablette vierge ne risquait pas d'ennuis.*". Les tablettes étant arrivées à Sparte, la reine Gorgô fit gratter la cire et découvrit ainsi le message de Démarate.

L'un des procédés stéganographiques les plus connus consiste à utiliser de l'encre *sympathique* (par exemple du jus de citron), invisible à l'œil nu, mais révélée par une flamme :

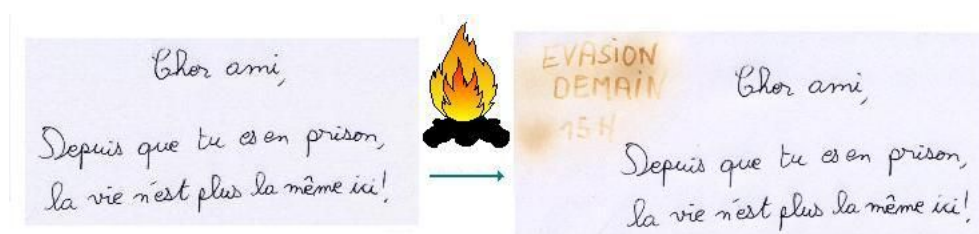


FIGURE 1 – Crédits : <http://www.bibmath.net>

La **stéganographie** est donc un art différent de la **cryptographie**, qui consiste à rendre un message inintelligible. Pour prendre une métaphore, la stéganographie consisterait à enterrer son argent dans son jardin là où la cryptographie consisterait à l'enfermer dans un coffre-fort.

Stéganographie moderne

Une version moderne de la stéganographie consiste à dissimuler une image (que nous appellerons **message**) dans une autre image (que nous appellerons **porteuse**). Cette technique est à rapprocher de celle de *tatouage numérique* ou *watermarking*, utilisée notamment par les agences de presse à des fins de copyright sur leurs photographies.

L'idée consiste à remplacer une partie de l'information véhiculée par la porteuse, de faible importance, par la partie importante de l'information contenue dans le message (cette technique induit donc une *perte d'information*, à la fois dans le message et la porteuse, mais suffisamment faible pour que le message puisse être retrouvé et que les effets sur la porteuse soient à peine visibles).

L'information d'une image est contenue dans ses *pixels* : une image peut être vue comme un tableau de *pixels*, chaque pixel étant un quadruplet de nombres compris entre 0 et 255 (un pour le canal rouge (**R**), un pour le canal vert (**V**), un pour le canal bleu (**B**) et un pour la transparence ou *alpha* (**A**) (0 signifiant totalement transparent, et 255 signifiant totalement opaque)). Ainsi, une image de dimensions $w \times h$ pixels peut être vue comme un tableau de $w \times h \times 4$ nombres entre 0 et 255. La figure ci-dessous illustre cette correspondance image \leftrightarrow tableau de nombres, avec une image en noir et blanc ; avec une image couleur (RVBA), il faut imaginer que chaque case du tableau contient en fait **quatre** nombres :

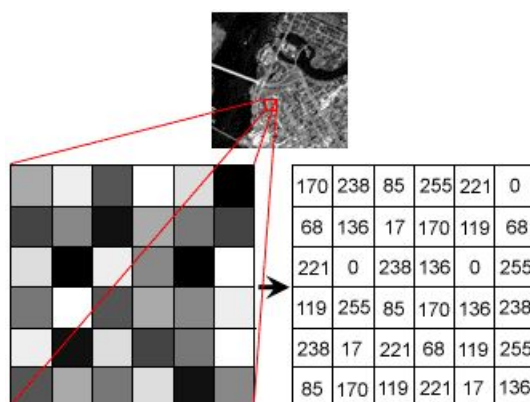





FIGURE 2 – Correspondance image \leftrightarrow tableau de nombres (Crédits : ai.stanford.edu/~syueung/cvweb/tutorial1.html)

Pour dissimuler le message dans la porteuse (**on suppose que les deux images ont les mêmes dimensions**), on choisit un nombre $1 \leq k \leq 8$ et on répète l'algorithme suivant pour chaque pixel :

1. convertir les valeurs RVB en *binnaire* des pixels correspondants dans la porteuse et le message
2. remplacer les k bits de poids *faibles* des valeurs RVB de la porteuse par les k bits de poids *forts* du message
3. donner au canal alpha la valeur 255

Exemple : le i -ème pixel de la porteuse a pour valeurs RVB : (113, 12, 75)  (soit (01110001, 00001100, 01001011) en binaire), et le i -ème pixel du message a pour valeurs RVB : (78, 207, 24) :  (soit (01001110, 11001111, 00011000) en binaire). En considérant (par exemple) que le message sera codé sur les $k = 4$ bits

de poids faible des pixels de la porteuse, le pixel résultat aura pour couleur (116, 12, 65) : 

	R	V	B
Pixel porteuse	01110001	00001100	01001011
Pixel message	01001110	11001111	00011000
Pixel résultat	01110100	00001100	01000001



TABLE 1 – Tableau 1 : principe du codage

💡 Comme seuls les bits de poids *faibles* des pixels de la porteuse sont modifiés, les pixels résultats ont des couleurs très proches de ceux de la porteuse ; ainsi, la différence sera à peine perceptible à l'œil humain et on verra pas qu'il y a un message caché à l'intérieur. D'un autre côté, les bits de poids faibles des pixels

résultats, une fois convertis en bits de poids *forts*, contiennent assez d'information pour retrouver l'image correspondant au message (tous les bits de poids faibles étant comblés par des 0) :

	R	V	B
Pixel résultat	01110100	00001100	01000001
Porteuse décodée	01110000	00000000	01000000
Message décodé	01000000	11000000	00010000

TABLE 2 – Tableau 2 : principe du décodage

Le pixel de la porteuse décodée a ainsi pour couleur (112, 0, 64)  et celui du message décodé a pour couleur (64, 192, 16) . On constate que ces pixels sont très proches des pixels originaux !

Exercice :

Le code HTML d'une interface permettant d'encoder et décoder une image par stéganographie vous est fourni, ainsi qu'un squelette de fichier JavaScript. Votre objectif est d'écrire les fonctions d'encodage et décodage.

Dans un premier temps, vous partirez du principe que le message est codé sur les 4 bits de poids faibles de la porteuse. Si vous avez le temps, ajoutez un slider à la page HTML pour configurer ce nombre de bits ; une modification de la valeur du slider recalculera automatiquement les images encodées et décodées.

💡 Dans ce cas, commencez par créer des fonction “utilitaires” chargées d'effacer les k bits de poids forts, ou faibles, ou d'effacer les autres bits...

💡 Dans cet exercice, vous aurez besoin d'utiliser les *opérateurs binaires* et de *décalage* (cf. https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Opérateurs/Opérateurs_binaires). En JavaScript, ces opérateurs savent travailler directement sur des nombres décimaux, donc vous n'avez pas besoin de convertir les valeurs en binaire.

⚠️ Faites très attention aux règles de précedence entre les opérateurs !