



TP1 « Images et sécurité informatique » / Stéganographie

JOBARD Guillaume – 21/09/2023 - Mewo – 2023/2024 – Inspiré de l'Université de Lille

Présentatrice : Magatte SEYE (BTS SIO Option SLAM)

Introduction

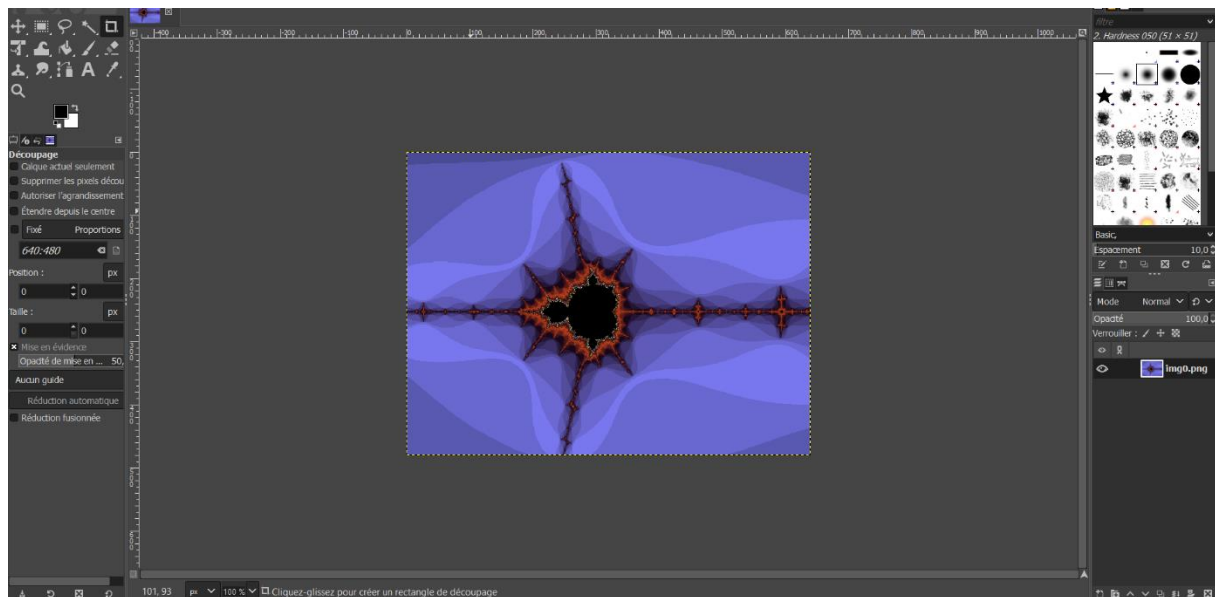
La Stéganographie est une technique de dissimulation d'informations à l'intérieur d'autres données, comme des images, des fichiers audio ou vidéo, sans que cela soit immédiatement perceptibles. Elle permet de cacher des messages ou des données secrètes de manière qu'elles ne soient pas détectées par des observations non autorisées. Cette discipline vise à dissimuler l'existence même de ces données, contrairement à la cryptographie qui se contente de rendre un message illisible. La stéganographie est utilisée dans divers contexte, notamment la sécurité, la confidentialité, la protection des droits d'auteur et de camouflage de donnés sensibles. (Ceci est extraite de google)

Les pixels dans une image

Cette partie nous permet de savoir comment traiter une image numérique à partir du logiciel de traitement d'image Gimp.

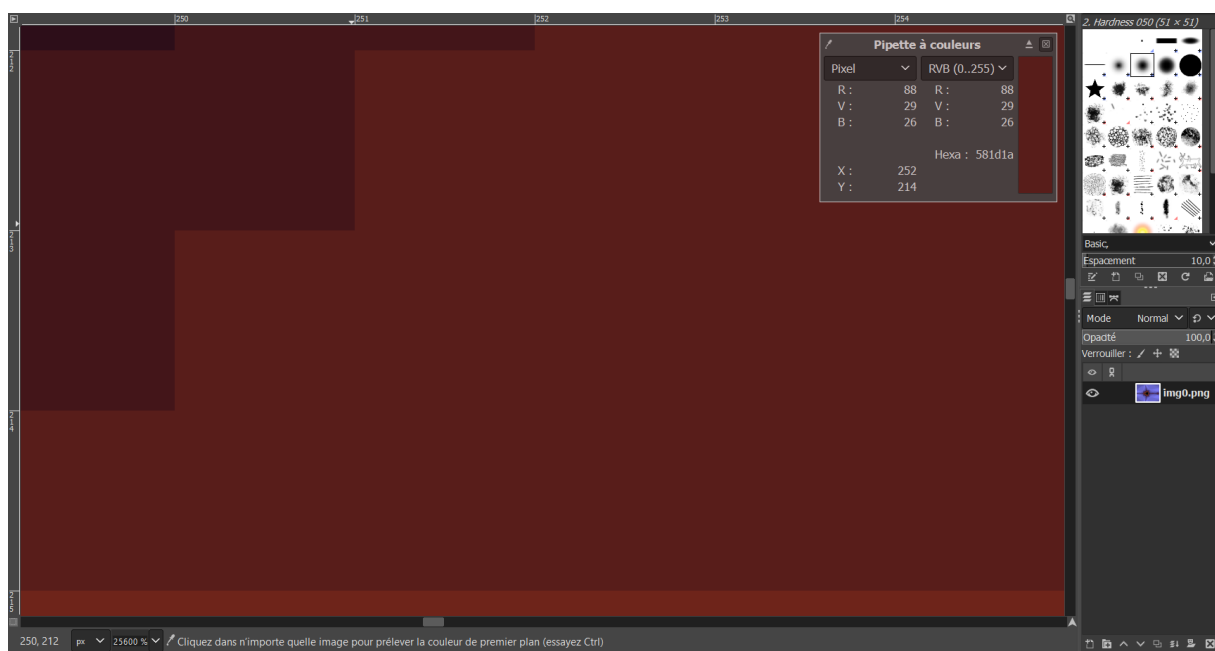
1. Recupérons l'img0.png présent sur le Moodle avec ce TP, et visualisons-la avec le logiciel Gimp.

Après récupération de l'img0.png j'ai procédé à la visualisation de cette image sur le logiciel Gimp en faisant un clic droit sur l'image en plus choisir l'option ouvrir avec Gimp.



2.Effectuons le grossissement maximal de l'image

Après récupération et visualisation de l'img0.png sur mon logiciel Gimp, j'ai effectué le grossissement maximal de 25600 % de coordonnées (252,214) de l'image en utilisant l'outil « loupe » puis appuyer sur la touche « Ctrl » pendant le clic permet un Zoom arrière.



Couleur d'un pixel

Déterminons la couleur du pixel de coordonnées (252,214) de l'image précédente.

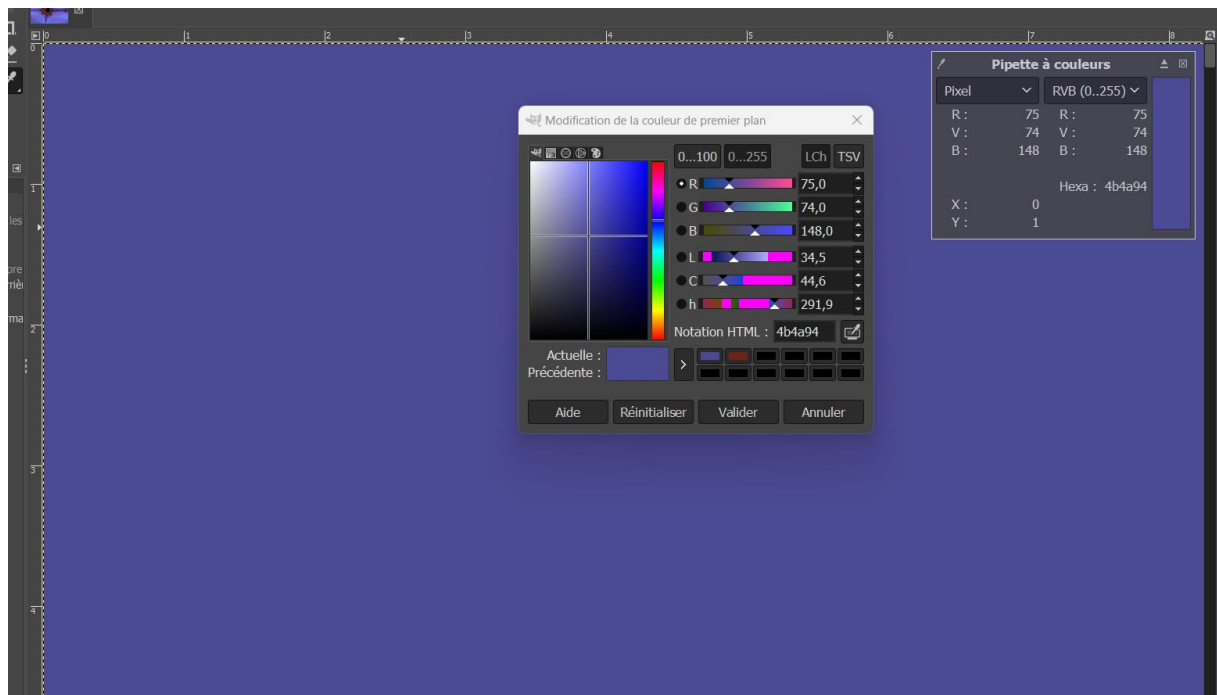
La couleur du pixel est rouge bardeau

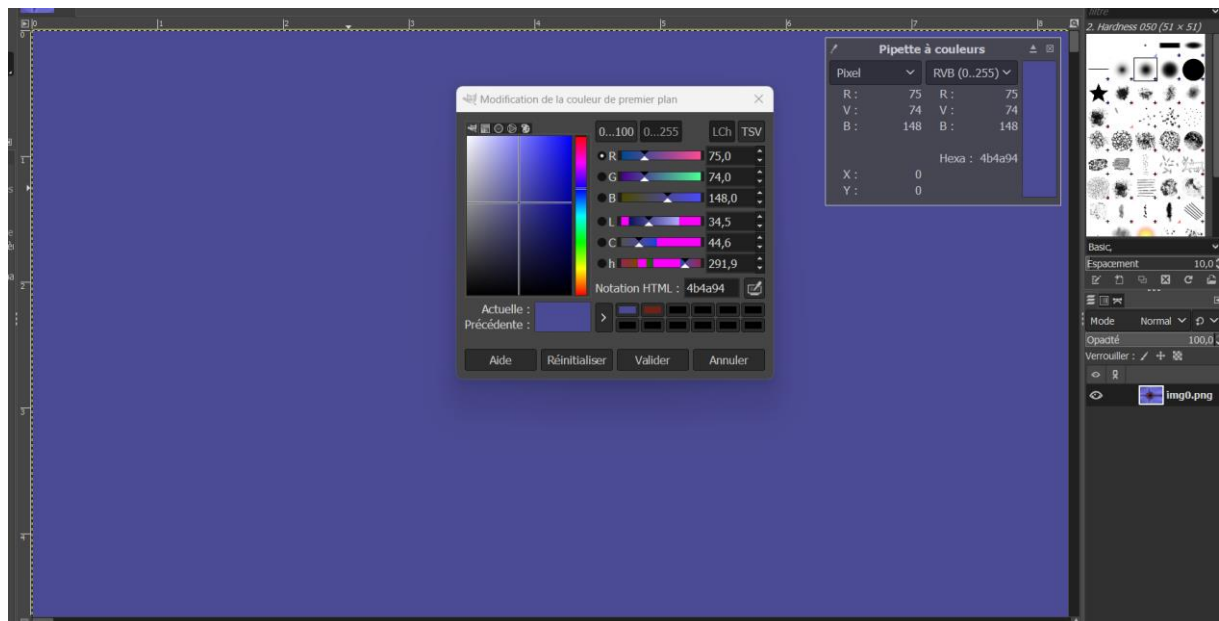
Le code hexadécimal de celui-ci est : 581d1a

Description du procédé stéganographique

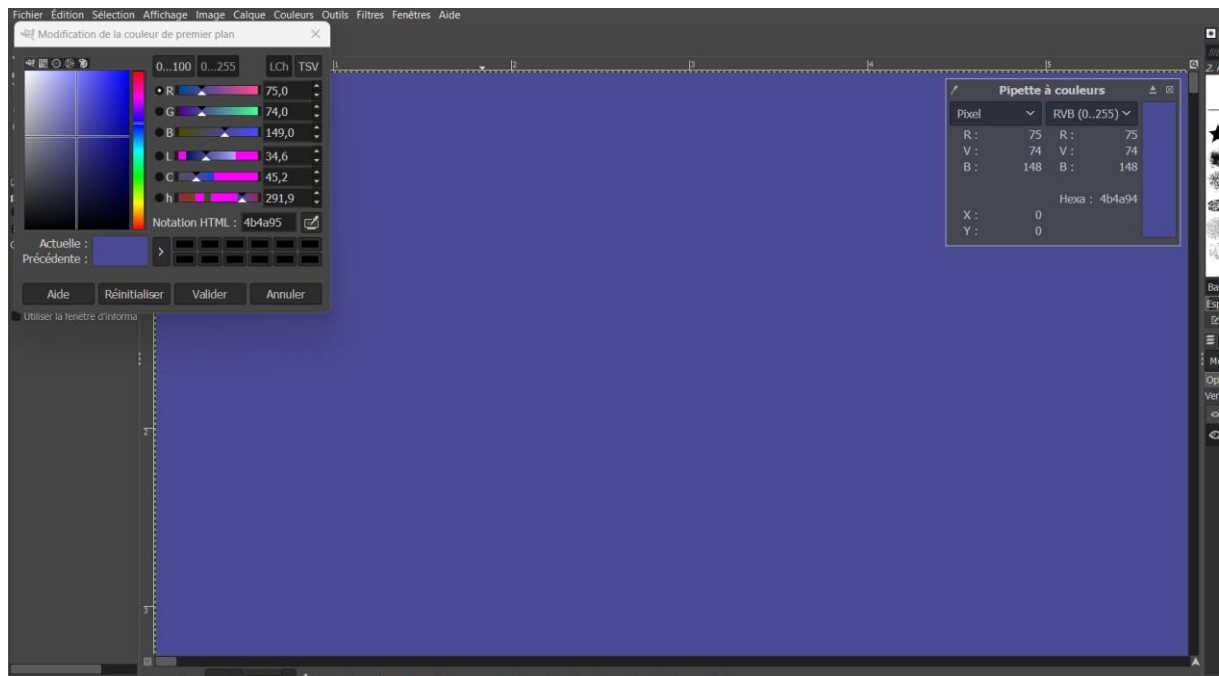
1. Verifions que les deux points de coordonnées (0,0) et (0,1) sont exactement de la même couleur.

Les deux points de coordonnées (0,0) et (1,1) sont exactement de la même couleur car ils ont la même valeur de composant bleue et même code hexadécimal





2. Modifions la couleur du pixel de coordonnées (0,0) en ajoutant 1 à la composante bleue de sa couleur, j'ai procédé comme on l'a bien dit dans l'énoncé du sujet : « dans la "Boîte à outils", après avoir sélectionnée la couleur du pixel avec la pipette cliquez sur la couleur de premier plan, vous pouvez alors modifier la composante bleue puis faites la modification à l'aide de l'outil rayon que vous aurez réglé pour qu'il n'affecte qu'un seul pixel. Attention à rester sur une grille de type « RGB (0...255) » ! »



3. Il n'y a pas de différence avec la pixel voisine

Retrouver un message

Retrouvons le nombre l du message

Pour retrouver ce nombre l je vais commencer à noter les valeurs des composants bleues sur les 8 premiers pixels ou il est codé et leurs bits de poids faible

(0,0) : 148

(1,0) : 148

(2,0) : 148

(3,0) : 148

(4,0) : 148

(5,0) : 149

(6,0) : 148

(7,0) : 148

148 en binaire est égale à 10010100

149 en binaire est égale à 10111111

Donc le bite de poids faible de :

148 est 0

149 est 1

Ceux qui faits que :

(0,0) : 148 ----> 0

(1,0) : 148 ----> 0

(2,0) : 148 ----> 0

(3,0) : 148 ----> 0

(4,0) : 148 ----> 0

(5,0) : 149 ----> 1

(6,0) : 148 ----> 0

(7,0) : 148 ----> 0

00000100

00000100 en décimale donne 4 donc $l=4$

Retrouvons le message dissimulé dans l'image

1) Commençons à noter les valeurs de composantes bleues.

(0,0) :148
(0,1) :148
(1,1) :149
(2,1) :148
(3,1) :149
(4,1) :148
(5,1) :149
(6,1) :148
(7,1) :148
(8,1) :148
(9,1) :149
(10,1) :148
(11,1) :148
(12,1) :148
(13,1) :148
(14,1) :149
(15,1) :148
(16,1) :148
(17,1) :148
(18,1) :149
(19,1) :148
(20,1) :148
(21,1) :148
(22,1) :148
(23,1) :148
(24,1) :148
(25,1) :148
(26,1) :149
(27,1) :148

(28,1) :148

(29,1) :148

(30,1) :148

(31,1) :149

2)Déterminons les valeurs de leur bit de poids faible.

Donnons d'abord en binaire les valeurs de composant bleue

148 = 10010100

149= 10111111

Le bit de poids faible est 0 pour 148

Le bit de poids faible est 1 pour 149

3)Trouvons le code binaire de leur caractère caché

(0,1) :148---->0	}	01010100
(1,1) :149---->1		
(2,1) :148---->0		
(3,1) :149---->1		
(4,1) :148---->0		
(5,1) :149---->1		
(6,1) :148---->0		
(7,1) :148---->0		
(8,1) :148---->0	}	01000010
(9,1) :149---->1		
(10,1) :148---->0		
(11,1) :148---->0		
(12,1) :148---->0		
(13,1) :148---->0		
(14,1) :149---->1		
(15,1) :148---->0		

(16,1) :148 ---->0	}	00100000
(17,1) :148---->0		
(18,1) :149---->1		
(19,1) :148---->0		
(20,1) :148---->0		
(21,1) :148---->0		
(22,1) :148---->0		
(23,1) :148---->0		
(24,1) :148---->0	}	00100001
(25,1) :148---->0		
(26,1) :149---->1		
(27,1) :148---->0		
(28,1) :148---->0		
(29,1) :148---->0		
(30,1) :148---->0		
(31,1) :149---->1		

Le code binaire de leur caractère caché est 01010100010000100010000000100001

NB : J'ai utilisé les couleurs et les accolades pour pas oublier un bit ou se tromper sur le calcul.

4) Révétons le message

Le message est codé sur les 32 bits codant chaque caractère sur 8bits. Pour trouver le message je vais relever le bit de poids faible de chaque pixel puis les convertir en hexadécimal qui à la fin me donnera un caractère sur chaque 8bit d'où le message sera retrouvé

01010100---->T	}	TBespace!
01000010---->B		
00100000---->espace		
00100001---->!		

Le message est : TBespace!

Dissimuler un message

1) Reprenons l'image de laquelle nous avons extrait le message qui était dissimulé et Enregistrons le disque de format jpg.

Voici l'image au format jpg



2) Chargeons cette image et au format jpg avec Gimp et tentons de retrouver l'information dissimulée.

(0,1) :148---->1	}	11111111
(1,1) :149---->1		
(2,1) :148---->1		
(3,1) :149---->1		
(4,1) :148---->1		
(5,1) :149---->1		
(6,1) :148---->1		
(7,1) :148---->1		
(8,1) :148---->1	}	11111111
(9,1) :149---->1		
(10,1) :148---->1		
(11,1) :148---->1		
(12,1) :148---->1		
(13,1) :148---->1		
(14,1) :149---->1		
(15,1) :148---->		
(16,1) :148 ---->1	}	11111111
(17,1) :148---->1		
(18,1) :149---->1		
(19,1) :148---->1		
(20,1) :148---->1		
(21,1) :148---->1		
(22,1) :148---->1		

(23,1) :148---->1

(24,1) :148---->1	}	11111111
(25,1) :148---->1		
(26,1) :149---->1		
(27,1) :148---->1		
(28,1) :148---->1		
(29,1) :148---->1		
(30,1) :148---->1		
(31,1) :149---->1		





Le code binaire de leur caractère caché est 11111111111111111111111111111111

Le message est vide

Je constate que c'est le format de l'image qui définit le nombre de caractère dissimuler dans l'image. C'est pourquoi lorsqu'on change le format de l'image le message est vide.

3. Comparez la taille des deux fichiers aux formats jpg et png. Qu'en pensez-vous ?

La taille des deux fichiers sont différentes : celle au format jpg est plus grande que celle au format png donc on peut dire que la taille du fichier dépend du format et de la valeur du composant bleue car celle plus grande à la plus grande valeur de composant bleue que l'autre.

 stegano-img0 (2)		21/09/2023 14:40	Fichier PNG	48 Ko
 stegano-img0		01/10/2023 20:17	Fichier JPG	54 Ko