Cette tâche a été verrouillée le 29 Juin à 23:59.

Cet examen est à réaliser par groupes de deux étudiants. Dans l’unique cas où le nombre d’étudiants de la promotion est impair, un et un seul groupe de trois est autorisé.

Votre rendu s'effectuera sous la forme d'une archive au format .zip contenant vos codes sources.

Toute forme de plagiat ou utilisation de codes disponibles sur internet ou tout autre support, même de manière partielle, est strictement interdite et se verra sanctionnée d’un 0.

Ce mini-projet donnera lieu à des soutenances, voir la page dédiée pour plus de détails. Vos horaires de passages vous seront communiqués par votre campus.

Le but de ce mini-projet est d'implémenter en Python plusieurs algorithmes de cryptographie et stéganographie ayant pour point commun l'utilisation d'images.

Le traitement de ces images se fera nécessairement en utilisant la librairie [Pillow (Connexions vers un site externe.)](https://pillow.readthedocs.io/en/stable/index.html). Voir ce lien pour l'[installation (Connexions vers un site externe.)](https://pillow.readthedocs.io/en/stable/installation.html) et celui-ci pour la [documentation (Connexions vers un site externe.)](https://pillow.readthedocs.io/en/stable/reference/Image.html). Nul besoin d'en devenir spécialiste, on se servira uniquement de quelques méthodes et fonctions, en particulier 'open', 'new', 'save', 'putpixel', 'getpixel' (liste pas forcément exhaustive) et de l'attribut 'size'.

Les menus de l'interface graphique seront créés en utilisant nécessairement la librairie Tkinter.

Vous adopterez obligatoirement une approche orientée objet.

**1 - Chiffrement d'images**

**1.1 - Principe**

Le but de cette partie est d'implémenter un algorithme symétrique de chiffrement d'images.

La clé sera elle même une image que les correspondants auront échangée en amont.

Pour générer des clés nous procèderons de façon aléatoire, en attribuant une valeur au hasard à chacune des composantes R, G et B d'un pixel.

L'image à chiffrer et la clé devront avoir les mêmes dimensions. On chiffrera un pixel de l'image en utilisant le pixel situé au même emplacement dans la clé.

Pour chiffrer un pixel, on procède comme suit pour chacune des composantes R, G et B :

* on convertit en binaire (sous forme d'un octet) la valeur de la composante de l'image à chiffrer et celle de la clé.
* on effectue un XOR bit par bit.
* on reconvertit le résultat obtenu sous forme d'entier.

L'image chiffrée sera alors constituée de tous ces pixels.

Le déchiffrement se fera exactement de la même façon, en considérant l’image chiffrée et la clé.

**1.2 - Implémentation en Python**

Implémenter une classe en Python permettant de chiffrer et déchiffrer une image avec l'algorithme précédent. Cette classe comportera également une méthode de génération aléatoire d'une clé. Le choix des images se fera en utilisant une interface graphique (utiliser pour cela des widgets 'Button' et 'filedialog', voir cette [documentation (Connexions vers un site externe.)](http://tkinter.fdex.eu/doc/popdial.html?highlight=filedialog) pour ce dernier).

Utiliser cette classe pour déchiffrer l'image [cipher1.bmp](https://canvas.supinfo.com/courses/402/files/76840/download?wrap=1) avec la clé [key1.bmp](https://canvas.supinfo.com/courses/402/files/76842/download?wrap=1).

Chiffrer une image de votre choix avec une clé générée aléatoirement et joignez la clé et l’image chiffrée à votre projet.

Quel format d’image faut-il utiliser ?

Est-il intelligent d’utiliser deux fois la même clé ? Pourquoi ?

**2 - Masquage d'images**

**2.1 - Principe**

Le but de cette partie est d'implémenter un algorithme symétrique de masquage d'images.

La clé sera elle même une image que les correspondants auront échangée en amont.

Cet algorithme fonctionne avec un paramètre LaTeX : kk réel strictement positif.

L'image à masquer et la clé devront avoir les mêmes dimensions. On masquera un pixel de l'image en utilisant le pixel situé au même emplacement dans la clé.

Pour masquer un pixel, on procède comme suit pour chacune des composantes R, G et B :

* à la valeur de la composante de la clé, on ajoute LaTeX : \frac{1}{k}1k multiplié par la valeur de la composante de l'image à masquer.
* si ce résultat dépasse 255 on le tronque à 255.

L'image masquée sera alors constituée de tous ces pixels.

La reconstitution de l’image initiale se fera exactement de la même façon, en considérant l’image masquée et la clé. Au lieu d’effectuer un calcul du type LaTeX : p=n+\frac{1}{k}mp=n+1km, on calculera LaTeX : m=k(pn)m=k(p−n).

**2.2 - Implémentation en Python**

Implémenter une classe en Python permettant de masquer et démasquer une image avec l'algorithme précédent. Le choix des images se fera en utilisant une interface graphique (utiliser pour cela des widgets 'Button' et 'filedialog', voir cette [documentation (Connexions vers un site externe.)](http://tkinter.fdex.eu/doc/popdial.html?highlight=filedialog) pour ce dernier).

Utiliser cette classe pour démasquer l'image [cipher2.bmp](https://canvas.supinfo.com/courses/402/files/76841/download?wrap=1) avec la clé [key2.bmp](https://canvas.supinfo.com/courses/402/files/76843/download?wrap=1) et la valeur LaTeX : k=64k=64.

Masquer une image de votre choix avec une clé de votre choix et joignez la clé et l’image masquée à votre projet. Préciser également la valeur du paramètre LaTeX : kk utilisé.

**3 - Masquage de textes**

**3.1 - Principe**

Le but de cette partie est d'implémenter un algorithme symétrique de masquage de texte derrière une image.

On se donne donc une image. On commence par en effectuer un pré-traitement. Chaque composante R, G, B de l’image est remplacée par 254 si elle vaut 255. Elle est conservée dans le cas contraire.

On va ensuite considérer les composantes R, G et B de chaque pixel les unes à la suite des autres (celles du premier pixel de la première ligne, puis celles du second pixel de la première ligne, etc.).

Un caractère du message d’origine sera manipulé avec sa valeur Unicode représentée sous forme d’octet, et sera dissimulé par une suite de huit de ces composantes R, G, B comme suit :

* On va considérer le premier bit avec la première composante, puis le second bit avec la seconde composante, etc.
* Si la composante est paire, on ne la modifie pas si le bit vaut 0 et on lui ajoute 1 si le bit vaut 1.
* Si la composante est impaire, on ne la modifie pas si le bit vaut 1 et on lui ajoute 1 si le bit vaut 0.

L'image dissimulant le texte sera alors constituée de tous ces pixels.

Pour retrouver un texte dissimulé dans une image selon cet algorithme, il conviendra de prendre les composantes R, G, B huit par huit et de constituer un octet selon la parité de leur valeur. Le nombre entier correspondant à cet octet sera la valeur Unicode d’un caractère. On procèdera ainsi, caractère par caractère.

**3.2 - Implémentation en Python**

Implémenter une classe en Python permettant de masquer et démasquer un texte avec l'algorithme précédent. Le choix de l'image se fera en utilisant une interface graphique (utiliser pour cela des widgets 'Button' et 'filedialog', voir cette [documentation (Connexions vers un site externe.)](http://tkinter.fdex.eu/doc/popdial.html?highlight=filedialog) pour ce dernier).

Utiliser cette classe pour démasquer le texte derrière l'image [cipher3.bmp](https://canvas.supinfo.com/courses/402/files/76845/download?wrap=1).

Masquer un texte de votre choix sous une image de votre choix, et joignez le résultat à votre projet.

**Durée** : 20 minutes de présentation sans interruption puis 10 minutes de questions/réponses.

Il s'agit d'une **soutenance technique** (et non commerciale) qui devra également comporter une **démonstration** du projet.

Vous exposerez votre travail à l'aide d'un document powerpoint (ou équivalent). Il n'est pas nécessaire d'envoyer ce support à votre examinateur en amont.

Il devra comporter les éléments suivants :

* Titre (1 slide)
* Sommaire / plan de la présentation (1 slide)
* Contextualisation / résumé de l'esprit du projet (2 ou 3 slides)
* Présentation et explication des points les plus importants du code (quelques slides). Il convient d'être pertinent dans vos choix.
* Bilan des difficultés rencontrées (1 ou 2 slides)
* Extensions possibles que ce soit du code en lui même, ou de la thématique en général (1 ou 2 slides)

**Barème** :

* Aisance oratoire : 2 points
* Qualité du powerpoint de présentation : 2 points
* Introduction et résumé du projet : 2 points
* Qualité de l'argumentation  : 4 points (à propos du code, mais aussi des difficultés rencontrées et des extensions possibles)
* Choix pertinent des points de codes expliqués : 2 points
* Démonstration du projet : 3 points
* Questions/réponses : 5 points