

# Compte-rendu du projet d'application informatique Cryptographie visuelle

Zacharia Beddalia      Pierre Dubaillay  
Gauthier Girot      Josquin Havard

19 mars 2017

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Principe Algorithmique</b>	<b>3</b>
2.1	Introduction à la cryptographie . . . . .	3
2.2	Principe Algorithmique . . . . .	3
2.2.1	Histoire et fonctionnement . . . . .	3
2.2.2	Application au chiffrement d'images . . . . .	3
2.2.3	La sûreté du One-Time-Pad . . . . .	5
<b>3</b>	<b>Présentation de l'application</b>	<b>6</b>
3.1	Description générale . . . . .	6
3.2	Choix programmatiques . . . . .	6
3.3	Plugins . . . . .	7
3.3.1	ImageFormatter . . . . .	7
3.3.2	Generator . . . . .	8
3.3.3	Cypherer . . . . .	9
<b>4</b>	<b>Difficultés rencontrées</b>	<b>10</b>
<b>5</b>	<b>Conclusion</b>	<b>10</b>

# 1 Introduction

Le but de notre projet est de créer une application permettant de chiffrer des images en utilisant l'algorithme du One-Time-Pad. Elle est également destinée à être utilisée lors de diverses manifestations comme la Fête de la Science ou les Journées Portes Ouvertes. Ce compte rendu a pour but d'expliquer dans un premier temps le principe algorithmique sur lequel repose notre application, puis dans un second temps de l'implémentation retenue pour le-dit principe.

## 2 Principe Algorithmique

### 2.1 Introduction à la cryptographie

La cryptographie est une discipline visant à protéger des messages. Elle prend son origine dans deux mots latin : crypto pour cacher et graphie pour écriture. C'est donc un procédé qui rend un message incompréhensible pour toutes personnes autres que les destinataires, qui eux connaissent le processus inverse permettant de déchiffrer le message.

Une notion fondamentale dans beaucoup de méthodes de cryptographie est la notion de clé de chiffrement. Une clé est un paramètre utilisé lors d'un algorithme de chiffrage. Celle-ci peut être symétrique ou asymétrique : dans le premier cas, la même clé permet de chiffrer et de déchiffrer le message, alors que dans le second cas, elle ne permet uniquement que de chiffrer le message.

Le chiffrement d'un message est le procédé qui permet à l'aide d'une clé de chiffrement de transformer un message compréhensible (message clair) en un message incompréhensible pour les personnes ne connaissant pas la clé (message chiffré).

Le déchiffrement d'un message est le procédé inverse, il permet de passer du message chiffré au message clair grâce à une clé.

Enfin, le décryptage d'un message est un procédé visant à passer du message chiffré au message clair sans connaître la clé de chiffrement. Il est donc important de concevoir des algorithmes de chiffrement rendant le décryptage quasi-impossible.

Utilisées depuis l'Antiquité avec notamment le code de César, les méthodes de cryptographie ont évolué jusqu'à nos jours et ont donné naissance à de nouveaux algorithmes comme le chiffrement de Vigenère, le RSA, les fonctions de hachage ou le One-Time-Pad. Elles sont utilisées aujourd'hui pour chiffrer nos communications, nos transactions, nos mots de passe ...

### 2.2 Principe Algorithmique

#### 2.2.1 Histoire et fonctionnement

L'algorithme du One-Time Pad (ou du masque jetable en français) a été décrit pour la première fois en 1882 par Franck Miller dans le but de sécuriser des messages télégraphiques. Il ne prend réellement forme qu'en 1917 par Gilbert Vernam et est ensuite perfectionné par Joseph Mauborgne qui y ajoute la notion de clé aléatoire. Le principe fondamental du One-Time Pad repose sur 3 principes conditionnant la génération de la clé : celle ci doit être aléatoire, de même taille que le message et ne doit être utilisée qu'une seule fois. Ces 3 principes garantissent que cet algorithme abouti à un cryptosystème parfait. Bien qu'il existe une version du One-Time Pad réalisable à la main (qui est similaire au chiffrement de Vigenère), le One-Time Pad a surtout vocation à chiffrer des données numériques, dans leur représentation binaire. Le One-Time Pad chiffre en effectuant un XOR entre les bits du message à chiffrer et les bits de la clé. C'est un système à clé symétrique.

#### 2.2.2 Application au chiffrement d'images

Une image est composée de pixel. Si dans beaucoup de format d'image, un pixel est représenté par un triplet correspondant à son niveau de rouge, de vert et de bleu, nous utiliserons pour la cryptographie visuelle un format d'image nommé PPM (Portable Mixmap) permettant de représenter les pixels par 2 valeurs : 0 pour le blanc et 1 pour le noir. Nous nous limiterons à ce cas uniquement par souci de clarté, mais il reste envisageable d'appliquer cet algorithme à des images en couleur.

Comme nous essayons de chiffrer une image, la clé devra elle aussi être une image. Sa génération consiste à assembler des blocs carrés de 4 pixels composés de 2 pixels blancs et de 2 pixels noirs. S'il est

alors possible de générer 6 blocs de pixels différents, nous n'en garderons que 2, car il a été observé que les quatres autres pouvaient conduire à des artefacts sur l'image chiffrée.

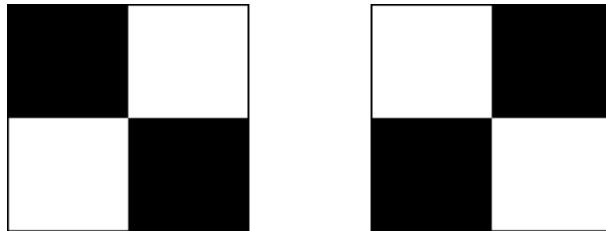


FIGURE 1 – Les 2 blocs qui constituent les clés

Comme nous utilisons des blocs de 4 pixels, l'image à chiffrer doit donc contenir un nombre pair de lignes de colonnes. Comme dans le One Time Pad classique, l'image clé est utilisée lors du chiffrement et du déchiffrement . Il s'agit donc d'un chiffrement à clé symétrique. La Figure 2 illustre le mécanisme de chiffrage et de déchiffrage d'une image. Une étape supplémentaire de conversion a été ajouté afin de transformer n'importe quelle image, en image prête à être chiffrée.

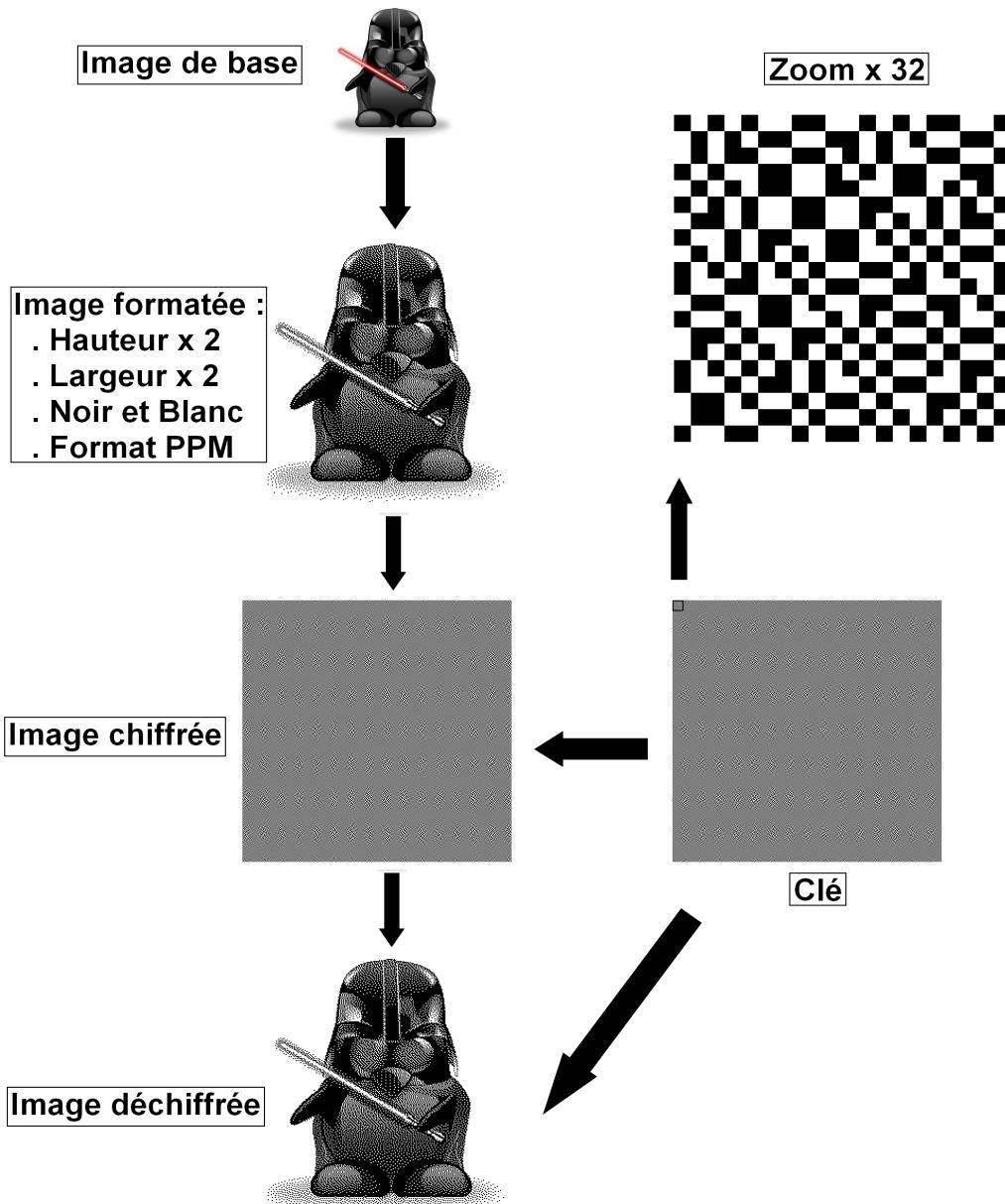


FIGURE 2 – Le processus de chiffrage et déchiffrage

### 2.2.3 La sûreté du One-Time-Pad

Comme il est dit plus haut, la génération de la clé doit suivre trois grands principes. Si il est facile de comprendre pourquoi la clé doit être aléatoire, le fait qu'elle doit avoir la même taille que le message est pour éviter que des motifs se créent sur l'image chiffrée, facilitant ainsi son déchiffrage. Enfin, une question légitime se pose : pourquoi ne pas pouvoir utiliser plusieurs fois la même clé pour chiffrer différents messages ?

Soit  $I_1$  et  $I_2$  deux images prêtes à être chiffrées et  $K$  la clé de chiffrement.

On a l'image chiffrée de  $I_1$  par  $K$  :

$$E_1 = I_1 \oplus K$$

L'image chiffrée de  $I_2$  par  $K$  :

$$E_2 = I_2 \oplus K$$

Si on cherche maintenant à déchiffrer  $E_1$  non pas avec  $K$  mais avec  $E_2$ , voici ce que l'on obtient :

$$E_1 \oplus E_2 = (I_1 \oplus K) \oplus (I_2 \oplus K)$$

$$E_1 \oplus E_2 = I_1 \oplus I_2 \oplus K \oplus K$$

$$E_1 \oplus E_2 = I_1 \oplus I_2$$

Ainsi, en superposant les 2 images chiffrées, nous obtenons simplement une superposition des 2 images en clair, ce qui en terme de sécurité présente quelques problèmes pour le secret de l'image. La Figure 3 illustre ce type de problème : deux images en partie blanches sur deux zones différentes, laissent transparaître les informations sensibles qu'elles contiennent.

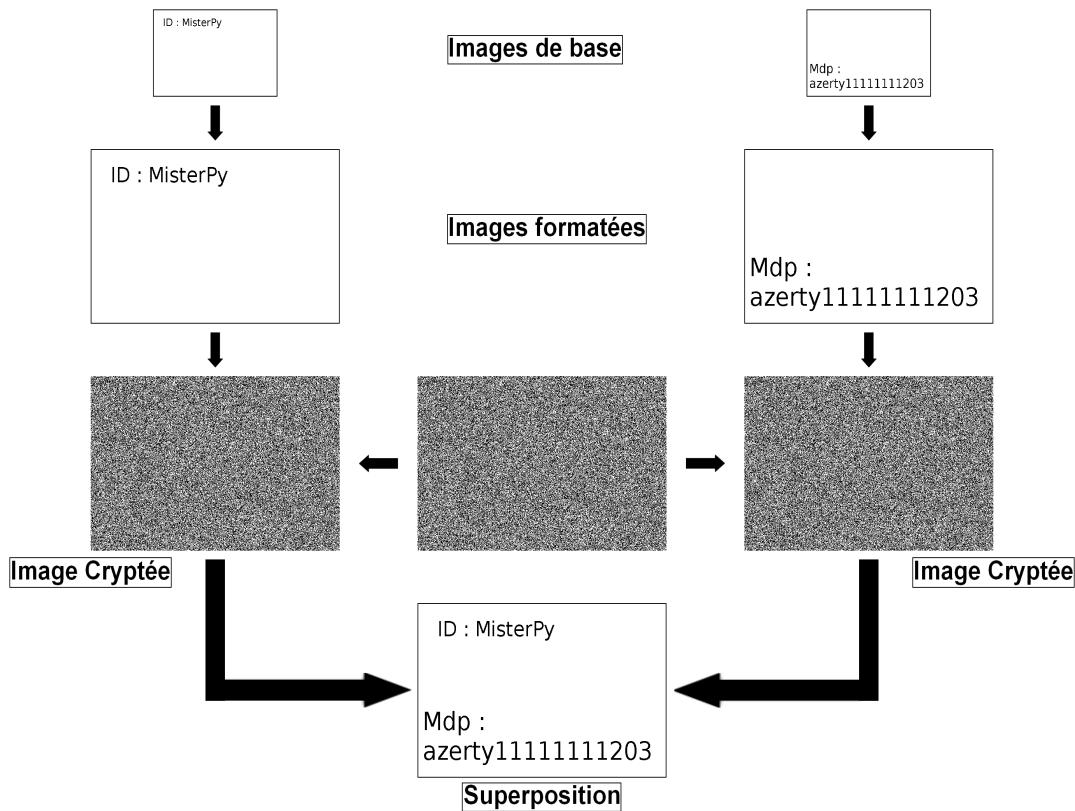


FIGURE 3 – Exemple de d'utilisation double d'une clé

### 3 Présentation de l'application

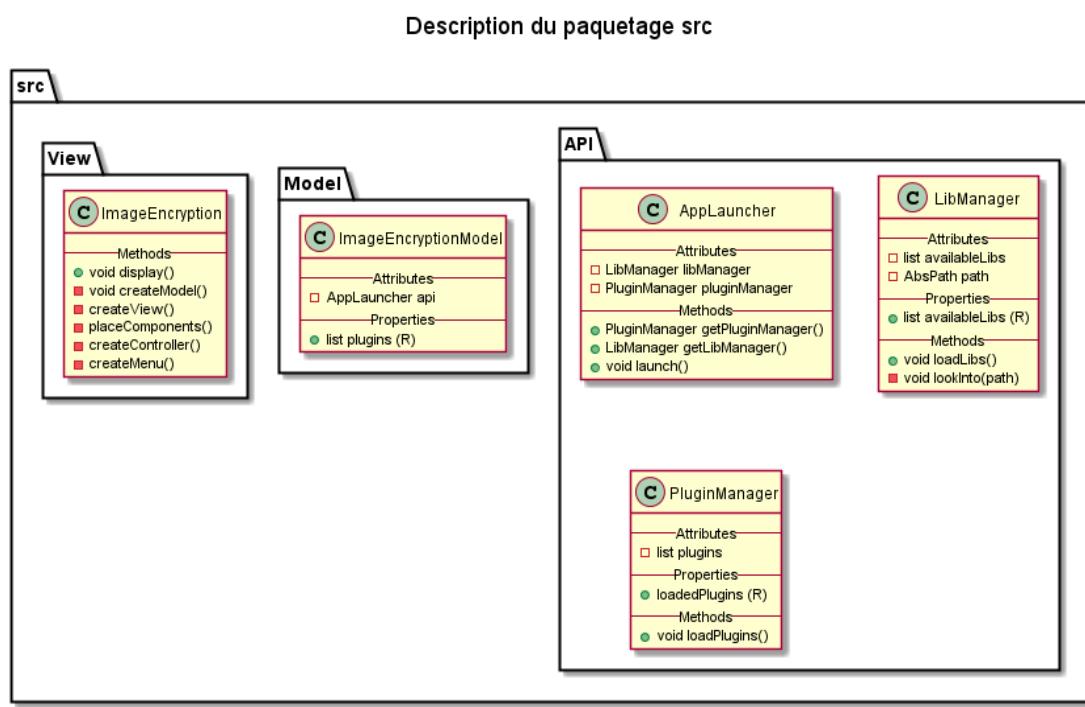
#### 3.1 Description générale

L'application se nomme Image Encryption. Elle repose sur le langage Python et les librairies Tkinter pour la partie graphique et Pillow pour la partie image. Elle propose les fonctions suivantes : conversion des images, génération de clé, chiffrage et déchiffrage d'images.

#### 3.2 Choix programmatiques

Concernant le langage de programmation, notre choix s'est porté su Python en version 3.5. Outre sa disponibilité sur de nombreuses plateformes et son choix de bibliothèques, python est un langage important dans le monde de la sécurité et il nous ait apparu intéressant de le découvrir et de l'utiliser (notamment pour ceux d'entre nous qui souhaitent poursuivre dans cette voie). Concernant les bibliothèques : Tkinter est une bibliothèque très souvent incluse dans les distributions de python, ce qui facilite le déploiement de l'application. Enfin, Pillow propose toutes les fonctionnalités de traitement d'image qui sont requises.

Concernant l'architecture, nous avons voulu rendre notre application la plus modulaire possible. Ainsi celle-ci est constituée d'un modèle principal et d'une vue appelés respectivement ImageEncryptionModel et ImageEncryption. La classe AppLauncher s'occupe d'initialiser l'environement (librairies et plugins) puis de lancer l'application. La force de notre application réside dans son système de plugins, chacun composés d'un modèle et d'une vue. Au moment de l'exécution de AppLauncher, ils sont alors initialisé et leur vue est intégrée sous forme d'onglet dans l'application. Tous nos plugins ainsi que l'application principale fonctionnent selon le modèle MVC : Modèle-Vue-Contrôleur.



Tous les attributs liés à des composants graphiques ne sont pas affichés

FIGURE 4 – Paquetage racine

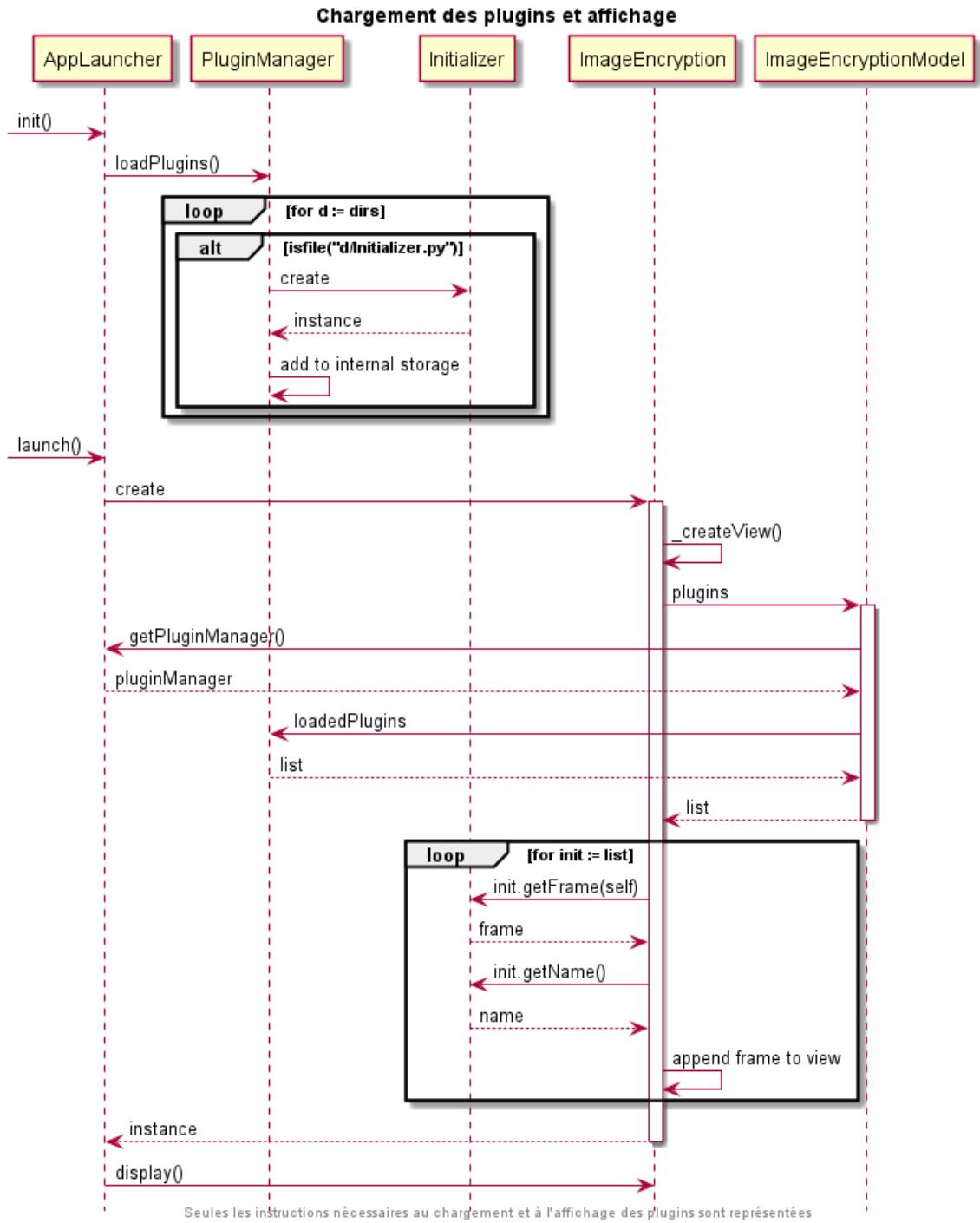


FIGURE 5 – Diagramme de séquence du chargement des plugins

### 3.3 Plugins

Chaque plugin joue un rôle particulier dans l'application, apportant des fonctionnalités qui lui sont propres.

#### 3.3.1 ImageFormatter

ImageFormatter est le plugin qui permet de formater les images. Il transforme une image quelconque en une image prête à être chiffrée. Le principe est le suivant : on multiplie d'abord la hauteur et la largeur de l'image par 2, créant ainsi une image de résolution 4 fois supérieure à celle de base, puis on convertit l'image en PPM. Les formats acceptés en entrée sont : jpg, jpeg, bmp, ps, gif, png, ppm, pgm et pbm.

Concernant la vue, le plugin permet de choisir l'image originale dans le système de fichier ainsi que de choisir l'emplacement de la future image convertie. Un rendue de chaque image est disponible.

## Description du paquetage ImageFormatter

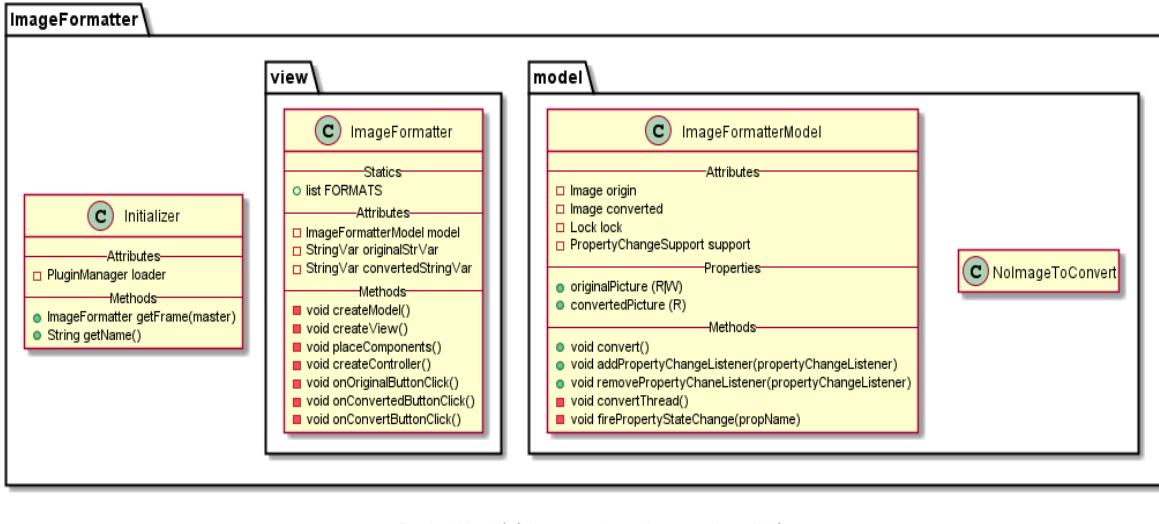


FIGURE 6 – Paquetage ImageFormatter

### 3.3.2 Generator

Generator est le plugin permettant de générer des clés de chiffrement. Il est possible de définir la taille de la clé soi-même ou d'utiliser la taille d'une image déjà existante (par exemple celle obtenue par ImageFormatter). La création de la clé se fait de la manière suivante :

- On crée les deux images qui composent la clé (celles de la Figure 1).
- On crée une nouvelle image vide de la taille de la clé.
- On pave ensuite aléatoirement la clé des 2 images de 4px.

Au niveau de la vue, la démarche est la suivante : on commence par choisir la taille de la clé (par valeurs ou par une image existante). Elle affiche ensuite la progression du calcul et, une fois celui-ci terminé, demande de spécifier l'emplacement où sera enregistré la clé. Un aperçu de celle-ci est disponible une fois la génération terminée.

## Description du paquetage Generator

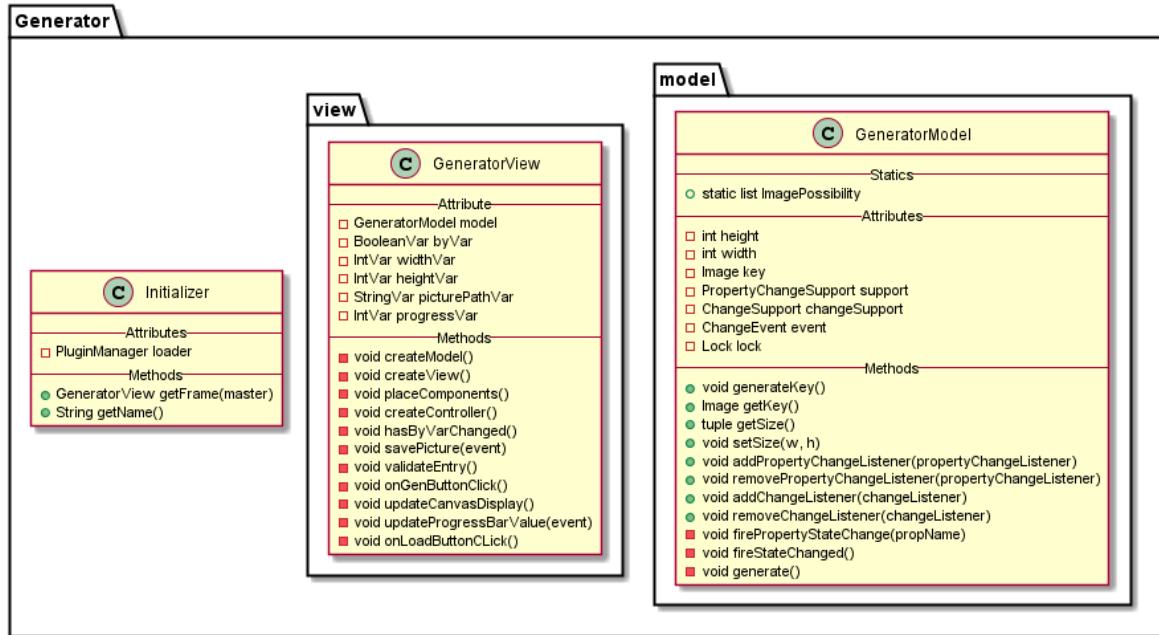


FIGURE 7 – Paquetage Generator

### 3.3.3 Cypherer

Le plugin Cypherer permet de chiffrer et de déchiffrer des images à l'aide d'une clé de chiffrement. Alors qu'à la base les actions de chiffrer et déchiffrer étaient dans 2 plugins différents, nous avons décidé de les réunir en un seul car ils faisaient la même chose.

Le chiffrage/déchiffrage d'une image se fait de la manière suivante :

- On récupère la liste des valeurs des pixels de la clé.
- On fait de même avec l'image à chiffrer/déchiffrer.
- On fait un XOR entre les deux listes et on stocke le résultat dans une liste résultat.
- On crée enfin une image résultat de la taille de la clé et on lui donne la liste résultat comme liste de valeur de pixel.

La vue nous propose de choisir entre chiffrer et déchiffrer une image. Cela aura juste pour conséquence de modifier les différents labels de la fenêtre. Ensuite nous pouvons charger la clé (optionnelle pour chiffrer), l'image à chiffrer/déchiffrer et choisir l'emplacement du résultat. À chaque fois, un aperçu de l'image est disponible. Puis, lors du chiffrage/déchiffrage de l'image, une barre de progression indique l'avancement de la procédure. Si lors du chiffrement aucune clé n'a été spécifiée, une nouvelle clé est générée et enregistré à l'emplacement choisi par l'utilisateur.

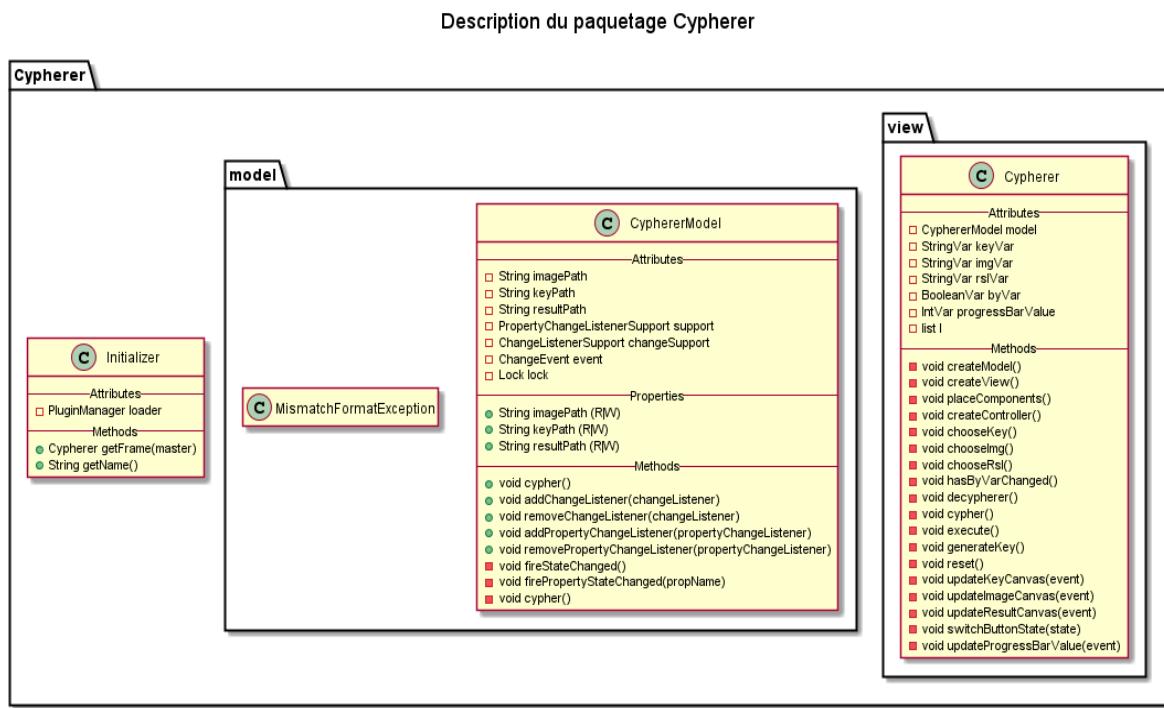


FIGURE 8 – Paquetage Cypherer

## 4 Difficultés rencontrées

La première difficulté rencontrée vient de l'interface graphique de l'application. Python ne prenant pas en charge de base l'architecture MVC, nous avons dû créer de nouvelles classes pour rendre cette architecture possible. La deuxième difficulté rencontrée est elle plus pratique. En effet, bien que l'université disposait d'une version de Pillow sur ses machines, celle-ci n'était pas à jour et il n'était pas possible que de manipuler un nombre restreint de format d'image. Il nous a donc fallu inclure celles-ci à notre projet et faire comprendre à notre application d'utiliser ces bibliothèques. Enfin, la dernière difficulté a été de trouver un moyen de distribuer notre application facilement. Comme celle-ci est destinée à être utilisée lors de différents événements, nous avons donc créé un script permettant de la charger facilement sur différentes machines.

## 5 Conclusion

Ce projet nous a tout d'abord permis de découvrir une nouvelle technique de cryptographie via l'algorithme du One-Time-Pad. Il nous a ensuite permis d'apprendre à travailler en groupe et de découvrir de nouvelles manières de travailler en collaboration, notamment avec l'utilisation de GitHub. Il a enfin permis à certains d'entre nous de découvrir un nouveau langage de programmation : le Python. Nous tenons, pour finir, à remercier Madame Cecile Gonçalves, notre responsable de projet, pour son aide via les documents qu'elle nous a fournis, ses réponses à nos nombreuses questions et la photo prise durant les JPO.

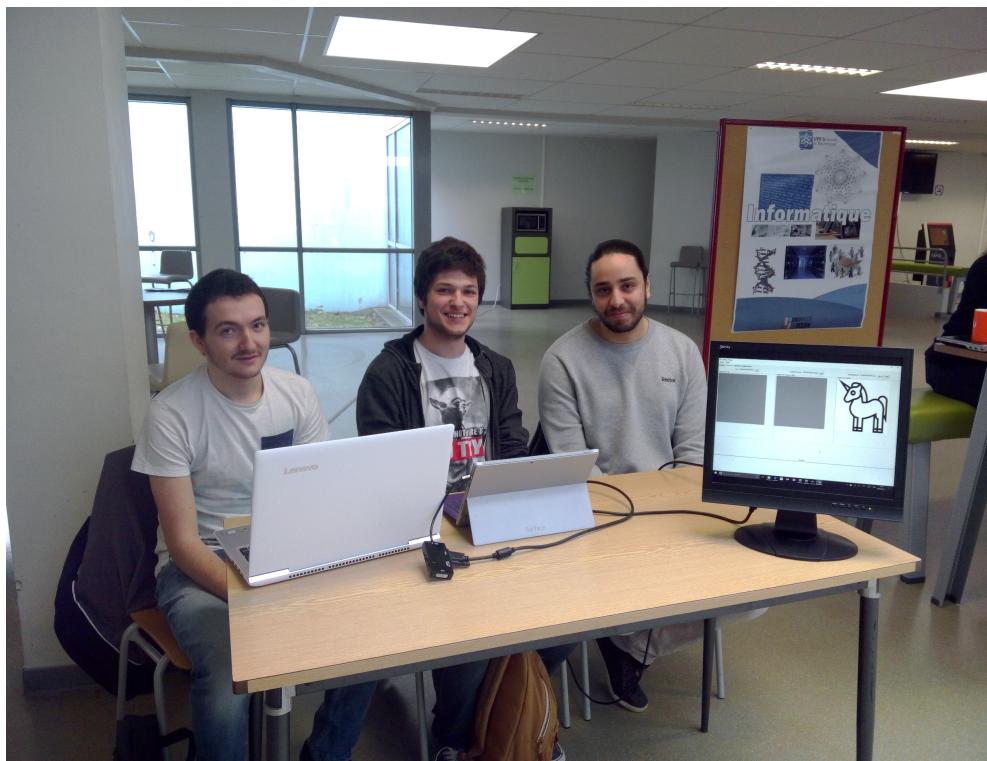


FIGURE 9 – Photo de la Journée Porte Ouverte du 4 Février 2017.