LIRMM

Manuel d'utilisation et d'installation

Cryptosystème de Paillier appliqué à des images

Rédigé par :
Katia Auxilien

Supervisé par William Puech

03.06.2024 (V1.0)

Table des matières

Table des figures

1 Introduction

1.1 Présentation de l'entreprise

Le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) est une unité mixte de recherche (UMR 5506). Il regroupe des informaticiens, des microélectroniciens et des roboticiens et mène des recherches dans les domaines de l'informatique, de la robotique et de la microélectronique.

1.2 Présentation du programme

Ce programme implémente le cryptosystème de Paillier pour des images en niveau de gris avec des options de réduction de la taille des chiffrés basée sur des calculs statistiques. Cette approche permet de faciliter le stockage et la transmission des données chiffrées.

1.3 Contexte et problématique

Le projet s'inscrit dans un cadre de sécurité multimédia et vise à résoudre le problème de la taille des chiffrés obtenus avec le cryptosystème de Paillier, qui est deux fois plus volumineuse que le message initial. Cette situation peut entraîner des difficultés lors du stockage et de la transmission des données chiffrées, en particulier pour les images.

1.4 Équipe de recherche et domaine d'expertise

Le projet de recherche est mené au sein de l'équipe ICAR (Image et Interaction), qui fait partie du LIRMM et réunit des chercheurs des départements Robotique et Informatique. Cette équipe se spécialise dans la thématique de l'image et des données visuelles, et ce projet est principalement concerné par l'axe de Sécurité Multimédia et est supervisé par William PUECH.

2 Programme

2.1 Caractéristiques

Le programme est supporté sur **Linux**. La diffusion est Open-Source, ainsi le code source du programme est gratuit et disponible sur GitHub tout au long de son développement, conformément au principe de science ouverte du LIRMM. Il s'agit d'un programme de chiffrement et déchiffrement par Paillier d'images au format **.pgm** uniquement.

2.2 Fonctionnalités

Le programme implémente de nombreuses fonctionnalités telles que :

- Chiffrement et déchiffrement d'image : Le programme est capable de chiffrer et de déchiffrer des images en nuances de gris au format .pgm en utilisant le cryptosystème de Paillier.
- Compression de la taille du chiffré : Le programme offre différentes méthodes d'optimisation de la taille du chiffré, en optimisant le paramètre aléatoire et en optimisant un paramètre de la clé publique.
- La distribution du pixel chiffré sur deux pixels.
- L'expansion d'histogramme.
- Gestion des clés : Le programme permet de générer, d'enregistrer les clés privées et publiques nécessaires au chiffrement et au déchiffrement des images.
- Gestion des arguments : Le programme offre la possibilité de préciser de nombreux arguments pour son lancement tel que :
  - Le mode chiffrement ou déchiffrement.
  - Le chemin de fichier de l'image en claire ou chiffrée.
  - Si l'on souhaite générer une clé, on pourra préciser les paramètres de sa génération.
  - Si on souhaite utiliser une clé publique ou privée existante, on devra pouvoir préciser le chemin de ce fichier binaire.
  - Si l'on souhaite ou non effectuer une expansion d'histogramme.
  - Si l'on souhaite distribuer le pixel chiffré sur deux pixels.
- Vérification des arguments : Le programme vérifie la validité des arguments entrés par l'utilisateur pour éviter les erreurs et faciliter son utilisation.
- Ergonomie et facilité d'utilisation : Le programme est conçu pour être facile à utiliser par les utilisateurs non techniques, avec des messages d'erreur utiles, des instructions claires et des temps de chargement et de traitement raisonnables.
- Calculs statistiques : La structure du code source possède une partie dédiée aux calculs

statistiques sur les différentes implémentations du cryptosystème de Paillier, permettant ainsi d'effectuer des comparaisons.

— Modularité et extensibilité : Le programme est développé en respectant le principe SOLID, les design patterns et l'architecture Modèle Vue Controleur, afin de garantir sa modularité et son extensibilité pour permettre l'ajout de fonctionnalités supplémentaires de manière simple et l'optimisation de la taille du chiffré en jouant avec tous les paramètres.

2.3 Structure

Le C++ étant un langage orienté objet, nous avons appliqué le principe SOLID, les design patterns et l'architecture Modèle Vue Controleur pour faciliter le développement.
En termes de packages et de classes, le programme est structuré tel que sur la figure 2.1.

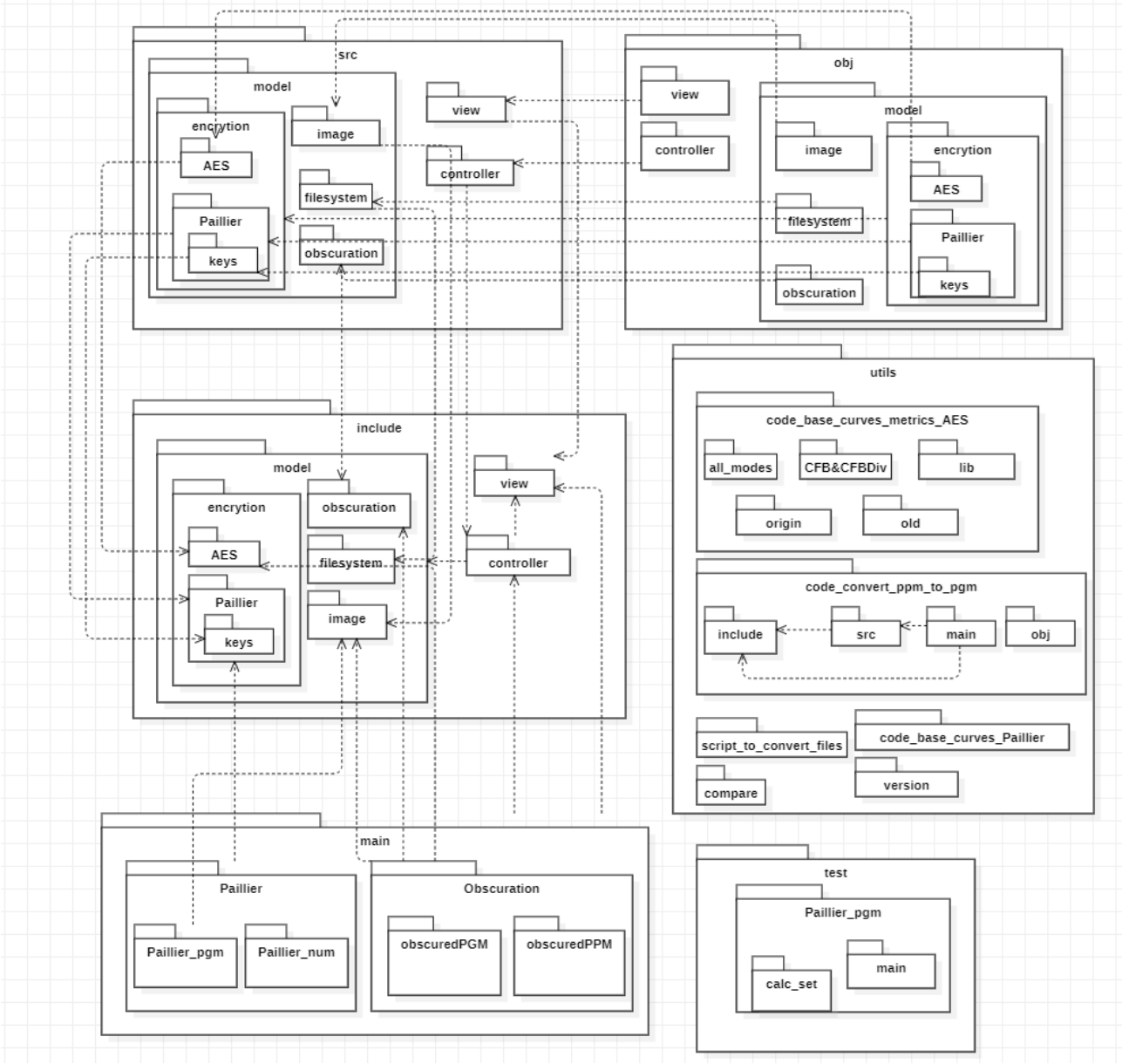

FIGURE 2.1 – Diagramme de package de la structure du code

Les dossiers `include` et `src` contiennent l'essentiel du code. Le dossier `include` contient tout les headers avec de la documentation, le dossier `src` contient l'implémentation des entêtes.
Le dossier `test`, contient les programmes de tests pour vérifier si vos modifications n'ont pas générer des défauts.
Le dossier `utils`, contient des programmes python qui pourraient servir lors du votre utilisation. Le dossier `utils/code_convert_ppm_to_pgm/` contient un programme python vous permettant, comme son nom l'indique, de convertir des images au format `.ppm` au format `.pgm`. Pour pouvoir effectuer d'autres conversions de fichiers en d'autres format, vous avez à votre disposition des programmes dans `utils/script_to_convert_files/`. Ensuite, afin de vérifier

que votre version de `g++` est bien la version 11.4. vous avez à votre disposition un programme dans `utils/version/version.cpp` qu'il faut compiler. Enfin, si vous avez un doute sur l'effet d'un chiffrement ou d'une obscuration, vous pouvez comparer deux images au format `.pgm` dans `utils/compare/`.

Le dossier `obj` contient tout les fichiers au format `.o`.

Pour finir, le dossier `main`, comporte les programmes principaux à lancer. À votre disposition, ils contiennent un fichier `Makefile` pour pouvoir compiler correctement le programme à chaque modification.

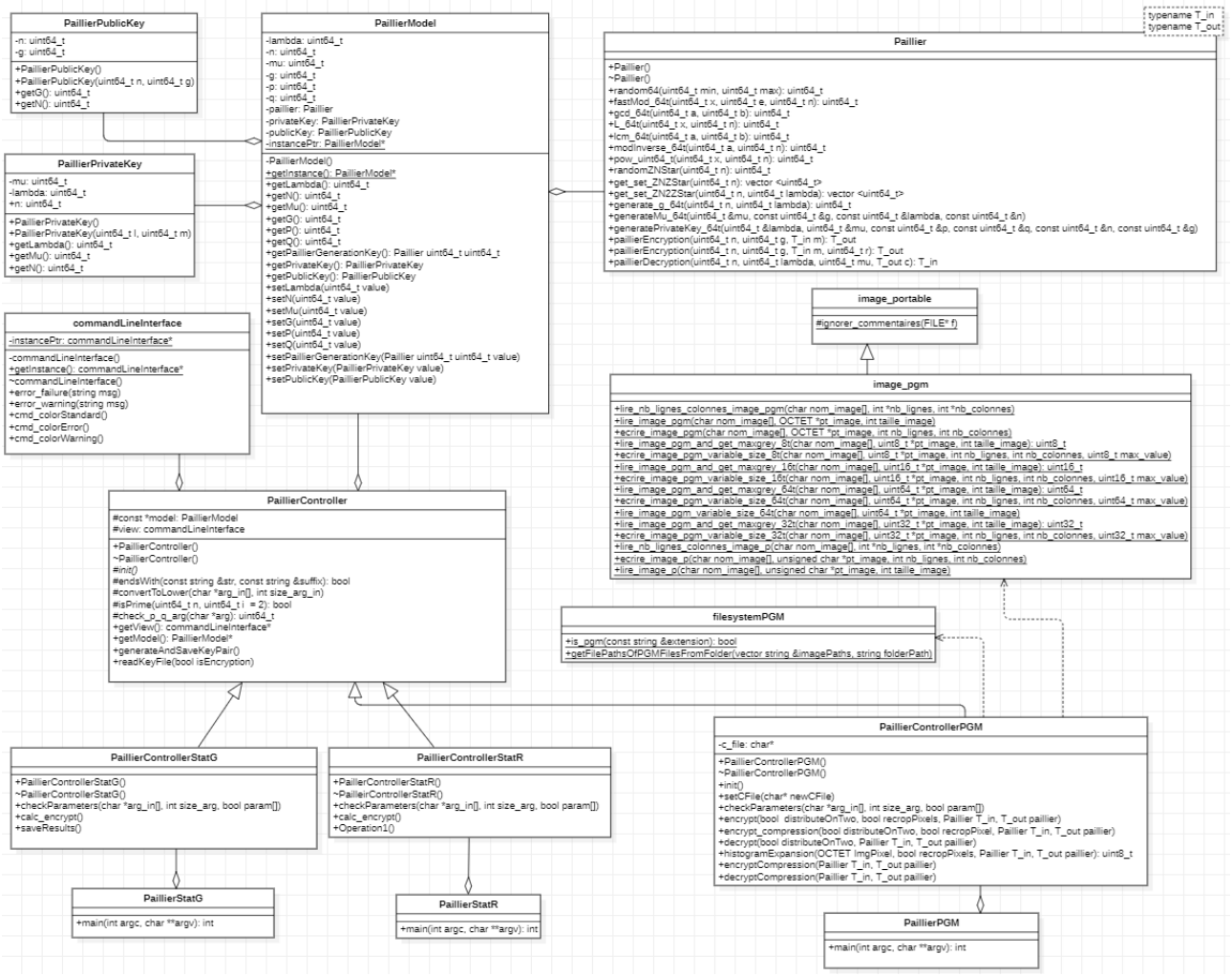

FIGURE 2.2 – Diagramme de classes de l'application

3 Installation

Premièrement, pour installer le programme, rendez-vous sur le répertoire GitHub du projet. Ensuite, clonez le projet ou téléchargez simplement les fichiers. Au fil de votre utilisation, nous vous conseillons de temps à autre d'effectuer la commande `git pull` pour mettre à jour le programme au fil des contributions.
Ensuite, installez `g++` version 11.

```
$ sudo apt install g++-11
```

Vous aurez probablement besoin d'installer `make`, utilisez la commande :

```
$ sudo apt install make
```

Afin de pouvoir compiler à chaque modification, vous avez à votre disposition des programmes `make` dans les dossiers de `main`. Pour cela, utilisez la commande :

```
$ make -f [nom fichier makefile]
```

4 Utilisation

4.1 Name

`$ ./Paillier_pgm_main.out` Lancer le programme.

4.2 Synopsis

```
$ ./Paillier_pgm_main.out [MODE] ... [FILE.PGM] ...
```

4.3 Description

Programme pour chiffrer ou déchiffrer avec le cryptosystème de Paillier.

4.4 Mode

4.4.1 Chiffrement

```
$ ./Paillier_pgm_main.out encryption [ARGUMENTS] [FILE.PGM]
$ ./Paillier_pgm_main.out encrypt [ARGUMENTS] [FILE.PGM]
$ ./Paillier_pgm_main.out enc [ARGUMENTS] [FILE.PGM]
$ ./Paillier_pgm_main.out e [ARGUMENTS] [FILE.PGM]
```

4.4.2 Déchiffrement

```
$ ./Paillier_pgm_main.out decryption [PRIVATE KEY FILE .BIN] [FILE.PGM] [ARGUMENTS]
$ ./Paillier_pgm_main.out decrypt [PRIVATE KEY FILE .BIN] [FILE.PGM] [ARGUMENTS]
$ ./Paillier_pgm_main.out dec [PRIVATE KEY FILE .BIN] [FILE.PGM] [ARGUMENTS]
$ ./Paillier_pgm_main.out d [PRIVATE KEY FILE .BIN] [FILE.PGM] [ARGUMENTS]
```

L'image à chiffrer ou déchiffrer peut être précisée après la clé ou les arguments, ou à la fin.

4.5 Options

```
$ ./Paillier_pgm_main.out encryption [p] [q] [FILE.PGM]
```

Mode chiffrement en précisant les arguments p et q, qui sont des nombres premiers dont

$pgcd(p \times q, p - 1 \times q - 1) = 1$.

4.5.1 Clés

`-k` ou `-key` pour préciser l'utilisation de la clé privée ou publique, suivi de `file.bin` notre fichier.

Mode chiffrement en précisant une clé publique sous forme de fichier `.bin`. `$ ./Paillier_pgm_main.out e`
`$ ./Paillier_pgm_main.out encryption -key [PUBLIC KEY FILE .BIN] [FILE.PGM]`

Mode déchiffrement en précisant une clé privée sous forme de fichier `.bin`. L'option `-k` est facultative, car il est **obligatoire** de préciser une clé privée.
`$ ./Paillier_pgm_main.out decryption -k [PRIVATE KEY FILE .BIN] [FILE.PGM]`

4.5.2 Autres

`-distribution` ou `-distr` ou `-d`

`-histogramexpansion` ou `-hexp` pour préciser lors du **chiffrement** qu'on effectue une expansion d'histogramme avant de chiffrer l'image.

`-optlsbr` ou `-olsbr` pour préciser qu'on souhaite utiliser la "compression" des pixels chiffrés en générant des valeurs aléatoires r favorable.

`-optlsbrcomp` ou `-olsbrc` pour préciser qu'on souhaite utiliser la "compression" des pixels chiffrés en générant des valeurs aléatoires r favorable et en utilisant des compléments de chiffré pour élargir les valeurs.

`-optlsbrg` ou `-olsbrg` pour préciser qu'on souhaite effectuer une "compression" des pixels chiffrés en générant des valeurs aléatoires r favorable et en générant le paramètre g le plus optimisé pour favoriser cette compression.

Bibliographie

[1] ANSSI. (2023). L'ANSSI publie le panorama de la cybermenace 2023. https://cyber.gouv.fr/actualites/lanssi-publie-le-panorama-de-la-cybermenace-2023

[2] LIRMM. (s.d.). Chiffres clés. https://www.lirmm.fr/chiffres-cles/

[3] Putaux, P., Vialle, M., & Puech, W. (2020). Homomorphic Encryption-Based LSB Substitution for High Capacity Data Hiding in the Encrypted Domain. IEEE Access, 8, 111547-111562. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9115625

[4] How to Install G++ (C++ Compiler) on Ubuntu par Lubos Rendek, 15 April 2024. https://linuxconfig.org/how-to-install-g-the-c-compiler-on-ubuntu-20-04-lts-focal-fossa-linux