Sécurité & Cryptographie :

Utilitaire de protection d’un système de fichiers par hash Mac.

Ce projet a été réalisé par Karl Woditsch & Georges Olivarès dans le cadre du cours de sécurité et cryptographie de deuxième année de l’ENSISA. L’ensemble des sources et contributions sont visibles librement sur internet à l’adresse suivante : <http://www.github.com/Rauks/PR-MacProtection>.

Sommaire

[Objectifs, conception et réalisation du projet 2](#_Toc354680889)

[1. Objectifs 2](#_Toc354680890)

[2. Conception 2](#_Toc354680891)

[3. Découpage du projet 2](#_Toc354680892)

[Cœur applicatif : MacProtection (Java 7) 3](#_Toc354680893)

[1. Conception 3](#_Toc354680894)

[2. Outils réalisés 3](#_Toc354680895)

[3. Fichier de validation 3](#_Toc354680896)

[Processus de création d’un fichier de validation 4](#_Toc354680897)

[Processus de lecture & certification d’un fichier de validation 4](#_Toc354680898)

[4. Architecture logicielle 5](#_Toc354680899)

[Interface graphique : MacProtectionGui (Java 7 & Java FX 2.3) 6](#_Toc354680900)

[1. Conception 6](#_Toc354680901)

[2. Vues graphiques 6](#_Toc354680902)

[Fenêtre principale 6](#_Toc354680903)

[Création et chargement de fichiers de validation 7](#_Toc354680904)

[Interface console : MacProtectionCui (Java 7) 8](#_Toc354680905)

[1. Conception 8](#_Toc354680906)

[2. Présentation des paramètres 8](#_Toc354680907)

# Objectifs, conception et réalisation du projet

## Objectifs

Le but de ce projet est de réaliser une application Java permettant d’associer un code d’authentification à un système de fichiers (arborescence de dossiers, sous-dossiers, fichiers) à partir d’un dossier racine.

## Conception

L’authentification des fichiers se fait par des hash Mac. Ceux-ci sont calculés pour chaque fichier suivant un certain algorithme et un mot de passe, connu de l’utilisateur désirant authentifier ses fichiers.

A cette approche simpliste, plusieurs problèmes se sont soulevés :

* Impossible de protéger les dossiers et leur arborescence.
* Impossible de protéger les noms de fichiers et dossiers.
* Impossible de détecter des déplacements de fichiers au sein de l’arborescence de dossiers.

Pour palier à ces problèmes nous avons conçu un système de protection par hash Mac faisant intervenir un fichier de vérification auto-protégé. Ainsi la protection de toute l’arborescence se fait via ce fichier qui nous permet de :

* Détecter tout ajout/suppression/modification de fichiers.
* Détecter tout ajout/suppression/modification de dossiers.
* Détecter tout déplacement de fichiers & dossiers.

Le fichier de vérification est auto protégé. Toute modification sur ce fichier (pour tenter de faire passer inaperçu une modification de l’arborescence à protéger) peut être détectée.

## Découpage du projet

Le projet a été découpé en trois sous-projets modulaires ayant chacun une spécialisation :

* **MacProtection** : Cœur applicatif.
* **MacProtectionGui** : Interface graphique.
* **MacProtectionCui** : Interface console.

MacProtection

Cœur applicatifC:\Users\Karl\Downloads\java_logo1.png

MacProtectionGui

Interface graphiqueC:\Users\Karl\Downloads\Javafx_logo_color.pngC:\Users\Karl\Downloads\java_logo1.png

MacProtectionCui

Interface console

C:\Users\Karl\Downloads\java_logo1.png

# Cœur applicatif : MacProtection (Java 7)

## Conception

Le cœur a été réalisé en Java 7. Il fournit des outils pouvant être utilisés pour réaliser des interfaces utilisateur (console ou graphique).

## Outils réalisés

Ce sous-projet fourni les outils nécessaires aux utilitaires de protection et est découpé en plusieurs parties :

* **Calculs et utilitaires de hash Mac** : Calculs des hash de fichiers à la volée via un FilterInputStream (MacInputStream) et listing des algorithmes de hash disponibles grâce a MacAlgorithm.
* **Représentation de l’arborescence de dossiers avec leur fichiers** : Classes du package core.tree (ParentedTreeElement, HashedFile, Folder). L’ensemble de ces classes implémentent Serializable et sont donc sérialisables.
* **Ecriture, lecture et certification de fichiers de vérification**: Classes du package core.check. Un fichier de vérification peut être généré à partir d’un arbre représentatif par la classe CheckWriter, l’auto certification est automatique. Ce fichier peut être lu (et certifié) par la classe CheckReader qui régénère un arbre représentatif. Le système de fichiers est authentifié par comparaison des arbres.
* **Processeur**: Classes du package core.processor. Construit les arbres représentatifs à partir de la racine du dossier à authentifier. Lors de ce processus, les hash Mac de fichiers sont calculés. Les calculs sont totalement multithreads et se répartissent sur les différents cœurs disponibles de la machines (en en laissant toujours un moins un libre si la machine est multi-cœurs pour le reste des exécutions). L’interruption des calculs inclus la prise en compte de l’interruption propre des threads. L’état de progression du processeur peut être observé via des événements (MacProcessorEvent) et écouteurs (MacProcessorListener).

## Fichier de validation

Le fichier de validation est composé d’un arbre représentation sérialisé et du hash Mac de l’arbre sérialisé. Le hash Mac interne permet de détecter toute modification sur l’arbre sérialisé :

|  |  |
| --- | --- |
| Taille du hash Mac | Hash Mac |
| Arbre sérialisé | |

Le fichier de validation permet de reconstruire l’arbre utilisé pour certifier une arborescence. Toute différence (ajout/suppression/modification) peut ainsi être détectée par comparaison. Que se soit sur les fichiers ou les dossiers.

### Processus de création d’un fichier de validation

ObjectOutputStream

GZipOutputStream

MacInputStream

FileOutputStream

Folder

Racine de l’arbre représentatif

Sérialisation

Lecture de l’arbre sérialisé

Ajout du hash Mac d’auto-validation

Compression

Ecriture

TemporaryFile

Fichier temporaire sans hash

File

Fichier final auto-protégé

### Processus de lecture & certification d’un fichier de validation

ObjectInputStream

GZipInputStream

MacInputStream

FileInputStream

Folder

Racine de l’arbre de validation

Désérialisation

Lecture de l’arbre désérialisé

Certification du fichier de validation

(Equité entre hash embarqué et hash simulé)

Décompression

Lecture

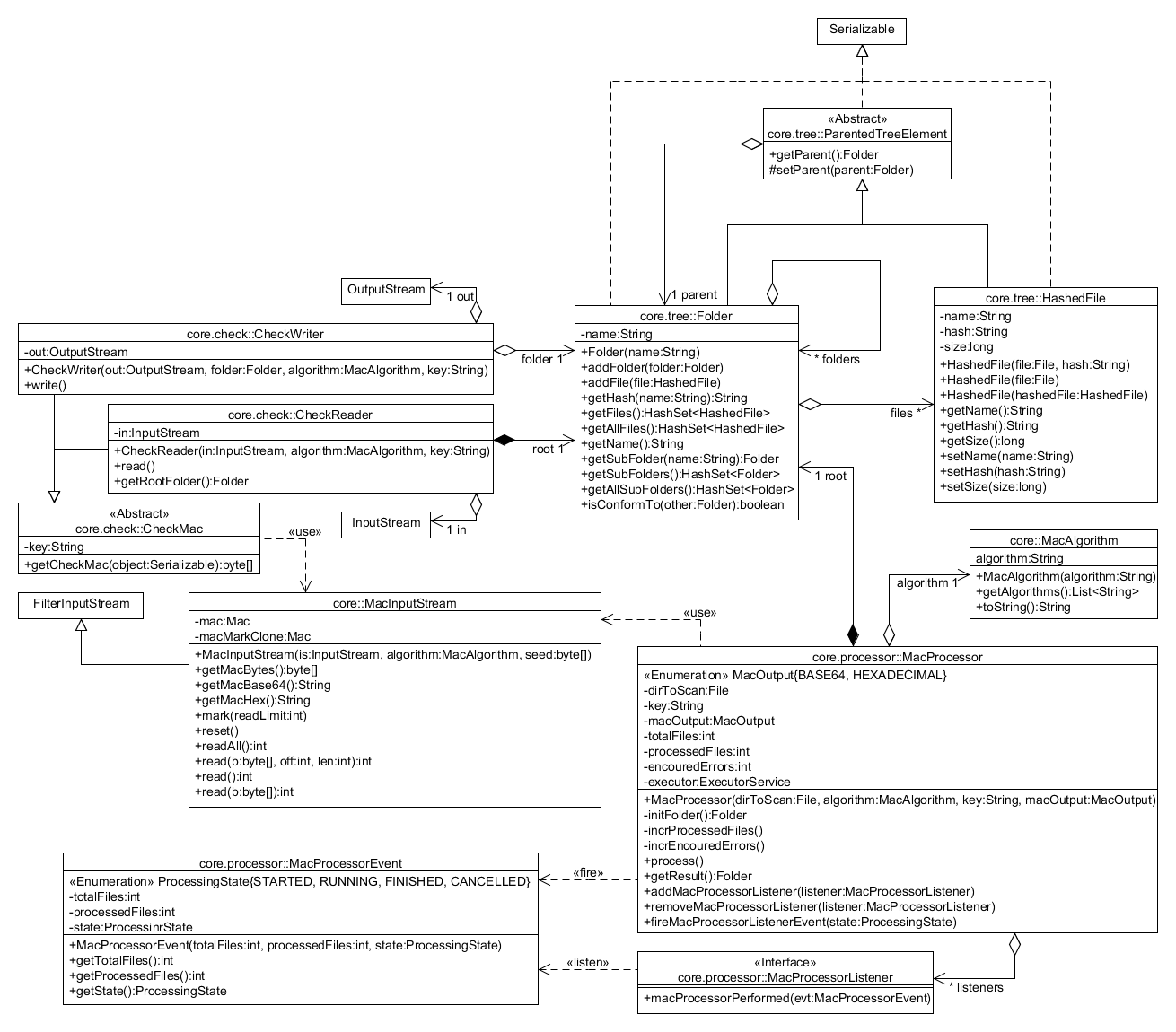
File

Fichier de validation

TemporaryFile

Fichier de validation simulé

## Architecture logicielle

Le diagramme UML suivant présente l’architecture du cœur.

# Interface graphique : MacProtectionGui (Java 7 & Java FX 2.3)

## Conception

Les fenêtres ont été réalisées avec Java 7 et Java FX 2.3 (Runtime intégré à Java 7, conseillé par Oracle comme remplaçant de Swing) et le langage de description FXML intégré à Java FX.

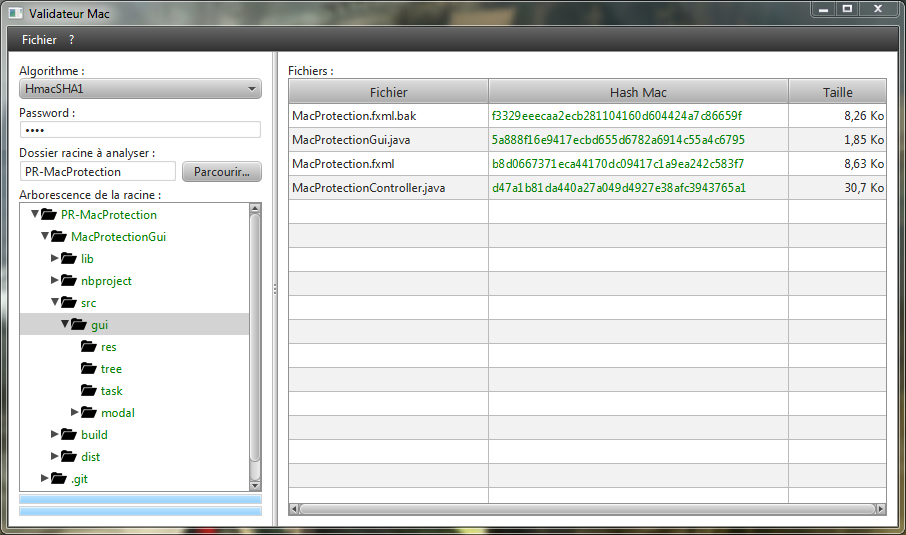
Le design pattern MVC est utilisé :

* **Modèle** : Cœur applicatif présenté précédemment.
* **Vue**: Descriptions FXML.
* **Contrôleur**: Contrôleurs Java / Java FX (Cf. annotations @FXML)

Pour les traitements lourds, des threads séparés sont utilisés (en plus des calculs multithreads apportés par le cœur). Toutes les interfaces utilisent des beans Java FX pour associer les données du modèle aux vues et contrôleurs.

## Vues graphiques

### Fenêtre principale

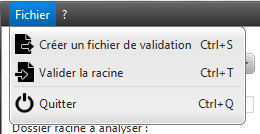


Cette fenêtre permet de choisir l’algorithme Mac utilisé, le mot de passe que l’utilisateur veut utiliser pour valider les fichiers, et la racine du dossier à protéger.

Des barres de progressions sont présentes pour indiquer les avancements de chaque étape des calculs et de construction des arbres. Des fenêtres modulaires indiquent les fins d’opérations.

Il est possible de naviguer dans l’arbre représentatif générer et de voir tous les hash Mac calculés pour tous les fichiers de chaque dossiers.

### Création et chargement de fichiers de validation

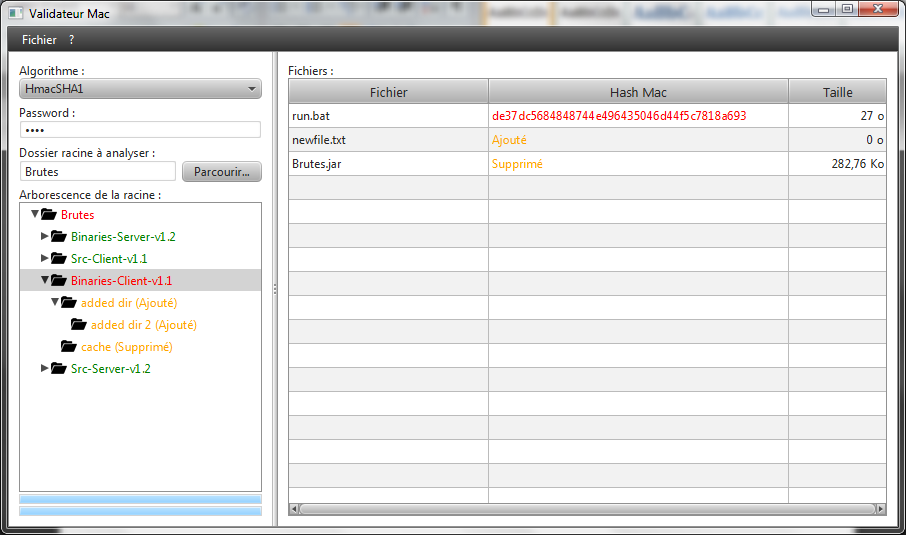
Le menu fichier permet de créer et charger des fichiers de validation.

Le chargement d’un fichier de validation implique la validation du dossier actuellement chargé. Lors de la validation, l’arbre representatif construit a partir du dossier racine sur le disque et l’arbre de validation construit a partir du fichier de validation sont comparés.

Après la validation, il est possible de repérer les fichiers et dossiers modifiés, supprimés ou ajoutés grâce à l’arbre de navigation et la liste des fichiers des dossiers. Un code couleur permet de rapidement repérer les problèmes :

* **Suppression** : Le fichier ou dossier a été supprimé (existant dans le fichier de validation mais pas sur le disque)
* **Ajout** : Le fichier ou dossier a été ajouté (existant sur le disque mais pas dans le dossier de validation)
* **Erreur** : S’il s’agit d’un fichier, cela signifie que son contenu a été altéré. S’il s’agit d’un dossier, c’est alors qu’un de ses fichiers ou sous-dossier comporte un problème (ajout/suppression/altération).
* **Valide**: Le dossier ou fichier est conforme, non modifié, non déplacé, non renommé.

Un déplacement ou le renommage d’un fichier ou dossier est perçu comme une suppression et une addition.



# Interface console : MacProtectionCui (Java 7)

## Conception

La partie interface console est réalisée uniquement en Java 7. Elle suit un modèle de commande Unix. Le programme est en lui-même une commande et ses arguments définissent l’action à réaliser (paramètres).

## Présentation des paramètres

L’interface console se base sur un ensemble de sous-programmes.

Pour cela, nous avons utilisé une librairie externe ([JSAP: Java Simple Argument Parser](http://www.martiansoftware.com/jsap/)[[1]](#footnote-1)) qui permet un contrôle et une utilisation poussée des arguments comme une console UNIX.

**Note :** Tous les paramètres possèdent deux déclarations :

* « -arg value » qui nécessite une déclaration ordonnée
* « --arg value » qui vous permet de les déclarer dans l’ordre choisis

### Affichage de l’aide (help)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cryo help [<command>] | | | | |
| *optional* | *–p* | *String* | *command* | *Affiche l’aide associée à la commande demandée* |

### Affiche la liste des différents algorithmes

|  |
| --- |
| cryo algos |
| *-* |

### Affichage de l’arborescence courante (show)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cryo show -p <password> -a <algorithm> [-s <source>] [-t <showTree>] | | | | |
|  | –p | String | password | Mot de passe |
|  | –a | String | algorithm | Algorithme de cryptage |
| *optional* | *–s* | *File* | *source* | *Dossier de travail* |

### Différence entre l’arborescence courante et un fichier de vérification (diff)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cryo diff -p <password> -a <algorithm> -c <check\_file> [-s <source>] [-t <tree>] | | | | |
|  | –p | String | password | Mot de passe |
|  | –a | String | algorithm | Algorithme de cryptage |
|  | –c | File | check\_file | Nom du fichier de comparaison |
| *optional* | *–s* | *File* | *source* | *Dossier de travail* |
| *optional* | *–t* | *boolean* | *tree* | *Affiche l’arbre récursif des différences ou non (def. true)* |

### Création et export du fichier de vérification

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cryo export -p <password> -a <algorithm> [-s <source>] [-t <showTree>] | | | | |
|  | –p | String | password | Mot de passe |
|  | –a | String | algorithm | Algorithme de cryptage |
|  | –f | File | file\_save | Fichier ou sera sauvegardé les informations |
| *optional* | *–s* | *File* | *source* | *Dossier de travail* |

## Exemples

***Note :*** *Un script bash a été crée dans le dossier cui/examples sous le nom cryo et permet l’exécution directe de l’application. Pour un souci de clarté, on utilisera dans les exemples ce raccourci.*

|  |  |  |
| --- | --- | --- |
| -java -jar ../../dist/MacProtectionCui.jar <command> [<arguments]> | ou | cryo |

### Aide

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **cryo**   |  | | --- | | $> cryo | | **cryo** **help**   |  | | --- | | $> cryo help | | **cryo help <command>**   |  | | --- | | $> cryo help help | |

### Affichage de la liste des algorithmes

|  |
| --- |
| $> cryo algos |

### Affichage de l’arborescence d’un dossier

Il est possible d’afficher l’intégralité des fichiers et dossiers avec leur hash correspondant.

|  |
| --- |
| $> cryo show –p password –a HmacMD5 –s ./monDossier |

### Exporter le résultat dans un fichier

|  |
| --- |
| $> cryo export –p password –a HmacMD5 –f file.check –s ./monDossier |

### Vérifier ou établir la liste des différences entre le dossier courant et le fichier de vérification

|  |
| --- |
| $> cryo diff –p password –a HmacMD5 –c file.check –s ./monDossier  $> cryo diff –p password –a HmacMD5 –c file.check –s ./monDossier –t false |

### Exemple d’arbre généré

|  |
| --- |
| $> cryo diff –p password –a HmacMD5 –c file.check –s ./monDossier  Scan './monDossier' directory  Numbers of files : 23  Scan progres : ||||||||||||||||||||||||  Compare with check file 'file.check' ... Reading ... DONE !  Validation result : false  Detailed tree :  - A DIRECTORY ../monDossier/added\_directory/  - A 6aebb41d4366c2794687e1d9e7aee307 ../monDossier/added\_file.txt  - M 7b83edc8e050292b77ac3fcde2e0075f ../monDossier/edited\_file.txt |

Un arbre contient des nœuds du type :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| <state\_flag E|A|M|D> | | DIRECTORY | | <file uri> | |
| <state\_flag E|A|M|D> | | <hash code> | | <file uri> | |
| *E = equal* | *A = added* | | *M = modified* | | *D = deleted* | |

Lors que l’on affiche l’arbre, on peut aussi voir apparaitre les fichiers inchangés :

|  |
| --- |
| - E a3ae529d74b94241dd6e62508444238c ../examples/file.check |

1. JSAP : <http://www.martiansoftware.com/jsap/> [↑](#footnote-ref-1)