Cahier des charges

PS4

Packup System 4

MAUBANC "Hyperion" Rémi
RIDEAU "Pokkihju" Cyril
CASIER "Zaihner" Frédéric
FARINAZZO "KernelKiller" Cédric

20 février 2019

Table des matières

Première partie

Introduction

Le projet PS4 s'appelle ainsi car nous voulions faire un logiciel de prédiction basée sur des statistiques. Ce projet ayant été refusé, nous avons décidé de faire un logiciel de sauvegarde, permettant la compression/décompression et le chiffrement/déchiffrement des fichiers/dossiers ainsi évidemment qu'un historique des fichiers aux différents états sauvegardés. L'objectif étant de pouvoir restaurer un état antérieur en cas de mauvaise manipulation ou de corruption de certains fichiers. Par exemple, si l'exécution d'un programme corrompt certains fichiers importants et les rend inutilisable ou si une personne tierce a eut accès à des fichiers et les a modifiés, notre logiciel permettra de récupérer un état exploitable dans lequel les fichiers ne sont pas corrompus ou n'ont pas été modifiés. Le problème avec ce genre de sauvegarde étant la place que cela prend sur le disque, nous incluront un algorithme de compression de fichiers afin de limiter la place prise par les différents sauvegardes.
Nous essaierons d'optimiser cet algorithme un maximum afin non pas de limiter le temps pris à la création d'une sauvegarde mais la place que celui-ci prend. L'objectif secondaire de ce projet sera d'inclure un algorithme pour chiffrer les données contenues dans la sauvegarde.

Deuxième partie

Le groupe

2.1 Présentation des membres du groupe

2.1.1 MAUBANC Rémi

Elève standard dans cette école, j'ai été accepté dans ce groupe pour mes réalisations extra-scolaire qui est le développement et la mise en production d'un site d'aide aux révisions. Mais également pour des notes pas trop mauvaise en programmation en Sup et pour ma motivation. Ce projet me permettra d'être plus à l'aise avec le language C que je ne maîtrise que peu.

2.1.2 RIDEAU Cyril

Elève en seconde année à l'EPITA, je suis un grand fan de jeu vidéo, et de programmation. J'ai commencé la programmation en terminale car je souhaitais comprendre comment les programmes que j'appréciais tant fonctionnaient. J'ai donc rejoins l'EPITA. Ce projet m'intéresse car il touche à la gestion I/O [1] et la gestion de fichier. J'aime beaucoup cette partie de la programmation car cela permet de faire beaucoup de choses complexes. J'espère donc que ce projet me permettra de me développer sur ce plan.

2.1.3 CASIER Frédéric

Actuellement en seconde année à EPITA, je suis un passionné d'art et d'informatique, deux domaines qui occupent la majorité de mon temps libre. J'ai commencé à coder en classe de Terminale, en apprenant les bases du langage Python. Dans le domaine de l'informatique, je suis principalement intéressé par le développement d'interface graphique et le traitement d'image. Toutefois, si j'ai rejoint ce projet, ce n'est pas pour retravailler sur une petite interface, mais pour découvrir d'autres domaines et prendre de l'expérience sur une facette de la programmation où je n'excelle pas.

2.1.4 FARINAZZO Cédric

Passionné d'informatique, ce projet m'intéresse énormément. Le fonctionnement de ce projet me semble complet du point de vue algorithmique. Je suis à l'aise dans la plupart des langages de programmation. Je n'aurai pas trop de problème à réaliser ce projet. Ce projet sera donc pour moi un moyen de découvrir en détail le monde de la cryptographie.

1. Input/Output en francais Entrée/Sortie ici du programme

2.2 Répartition des tâches

Voici la répartition des tâches pour la réalisation du projet :

Tâches	Personnes			
	Rémi	Cyril	Frédéric	Cédric
Système de fichier			X	X
Sauvegarde		X	X	
Compression	X	X		
Chiffrement		X		X
Site web	X			X
Interface graphique	X		X	

Troisième partie

Le projet

3.1 Sauvegarde

3.1.1 Système de fichier

Afin de conserver l'organisation des fichiers et les liens entre ceux-ci, nous avons décidé de sauvegarder les fichiers sous forme de graphe, chaque noeud contenant un fichier ou la liste des versions et les dates de sauvegarde de ce fichier avec l'historique. Cette structure de donnée est très adaptée car elle permet de sauvegarder des dossiers entiers facilement.

3.1.2 Type de sauvegarde

Les méthodes de sauvegarde

Il existe 4 méthodes de sauvegardes différentes, toutes ayant leurs avantages et inconvénients.

La première méthode de sauvegarde est la sauvegarde complète. Lorsqu'on utilise cette méthode, chaque sauvegarde copie intégralement tous les fichiers à sauvegarder, cette méthode est très lente et prends beaucoup de place mais évite toute perte de données pour une quelconque raison, elle est la plus sécurisée mais aussi la plus coûteuse. Elle permet également une restauration rapide.

La seconde méthode de sauvegarde est la sauvegarde incrémentale, elle sauvegarde a chaque itération de sauvegarde les modifications par rapport a la dernière sauvegarde. Cela permet d'économiser de la place et du temps a la sauvegarde. Cependant, le temps pris pour la restauration est grandement augmente car il faut rassembler les données.

La troisième méthode de sauvegarde est la sauvegarde différentielle. Elle suit les mêmes principes que la sauvegarde incrémentale a ceci près qu'elle sauvegarde les modifications depuis la dernière sauvegarde complète au lieu de la dernière sauvegarde. Ceci permet une restauration un peu plus rapide que la méthode incrémentale mais plus de temps est perdu lors de la sauvegarde et cette méthode demande plus de place. Cette méthode permet un compromis entre la sauvegarde complète et la sauvegarde incrémentale.

La dernière méthode est la méthode miroir. Celle conserve une seule sauvegarde des fichiers et écrase l'ancienne sauvegarde avec les nouvelles données. Cela permet d'économiser de la place par rapport aux autres méthodes, mais elle est beaucoup plus risquée car si une sauvegarde est faite par inadvertance avec une mémoire corrompue, toute la sauvegarde est corrompue. C'est donc une méthode a utiliser en cas de manque de place.

Notre choix

Afin de ne pas prendre trop de place ni de temps à sauvegarder, nous avons décidé de choisir la méthode incrémentale. Elle est intéressante à implémenter notamment pour la partie restauration. Elle permet également de faire des tests rapide en faisant plusieurs sauvegardes à la suite. De plus, cette méthode est rapide pour l'utilisateur la plupart du temps car un logiciel de sauvegarde sauvegarde plus qu'il ne restaure. Si nous arrivons à l'implémenter, nous souhaiterions proposer plusieurs méthodes de sauvegarde à l'utilisateur. Cependant, ces autres méthodes resteront des bonus si le projet est assez avancé.

3.1.3 Restauration

Une fois les fichiers sauvegardés, il est nécessaire de pouvoir les restaurer. Pour cela, lié à chaque fichier il y aura un fichier contenant le chemin menant à son emplacement original. Ce chemin permettra d'écraser les nouvelles modifications apportées et de restaurer l'état sauvegardé du fichier. Lorsque l'utilisateur souhaitera restaurer ses fichiers, le chemin d'accès sera lu et le fichier restauré à l'état sauvegardé. Pour cela nous utiliserons les fonctions de la librairie standard en C, notamment fopen, qui permet d'ouvrir un fichier, mais aussi fprintf (ou fwrite) qui permettra d'écrire directement dans le fichier. Bien évidemment, si le fichier en question a été chiffré et/ou compressé, cela sera indiqué dans le fichier associé avec la méthode à utiliser pour décompresser. De même, la méthode de chiffrement, si plusieurs sont utilisées, sera indiquée dans ce fichier. Cela permettra de déchiffrer et de décompresser le fichier avec la bonne méthode avant de le réécrire.

3.2 Compression

Avant de chiffrer les données et de réduire l'espace occupé, il est essentiel de les compresser auparavant.

3.2.1 Algorithmes de compression de données

Actuellement, il existe pléthore d'algorithme de compression (LZW, LZ77, PPM, Deflate, Arithmetical, FLAC, JPEG,...). Chaque algorithme de compression peut être plus ou moins spécialisé. C'est-à-dire qu'il sera plus ou moins performant selon le type de fichier à compresser. Par exemple, le Code Baudot est efficaces sur de la compression de fichier texte intelligible. En effet, sur des fichiers binaire contenant beaucoup de caractère spéciaux (non alpha-numérique), le fichier compressé serait plus lourd que le fichier d'origine. Ou comme le format FLAC (respectivement MP4) qui est spécialisé pour de la compression de fichier audio (respectivement de format vidéo).
Cependant, d'autres formats de compression sont assez efficace quelque soit le contenu du fichier à compresser (autrement dit, quelque soit le format et le type du fichier).
D'autres critères existent pour trier les algorithmes de compression, comme s'il est asymétrique : c'est-à-dire que l'algorithme de décompression est différent de l'algorithme de compression (on retrouve cette caractéristique dans les algorithmes de chiffrement entre autres) ou symétrique. Ou encore la méthode de compression : Codage par répétition, Codage entropique, Codage par dictionnaire, codage par modélisation de contexte... Tous les algorithmes de compression cités précédemment sont des algorithmes sans perte de données, c'est-à-dire qu'aucun bit de donnée n'est perdu après un cycle de compression-décompression. Cependant, il existe des algorithmes de compression provoquant une perte de données. Ils sont notamment utilisé pour les fichiers vidéo, audio et pour les images. Cela permet d'augmenter le taux de compression tout en restant acceptable pour le spectateur humain.

3.2.2 Notre choix

Pour notre projet, plusieurs pré-conditions interdisent l'utilisation d'une partie des algorithmes de compression. La première est que l'archive créée par la compression sera ensuite chiffrée, ce qui exclut les algorithmes de compression asymétriques. En effet, leur utilisation serait redondante et plus gourmande en ressources et en espace disque. Une sauvegarde est, par définition, censé permettre une restauration intégrale des données sauvegardées. Un critère qui proscrit l'utilisation d'un algorithme de compression avec perte de donnée. Autre points, une sauvegarde peut être consistuée de n'importe quel type de fichier, il nous faut donc un algorithme de compression dont la performance ne dépends pas ou très peu du type de fichier traité. Il ne nous reste plus qu'une fraction des algorithmes de départ. Notre choix s'est fait sur deux d'entre eux : Le codage de

Huffman et Lempel-Ziv78 (LZ78). Le premier est basé sur le codage entropique des données ; Le nombre de bit de codage d'une lettre est inversement proportionnelle à sa probabilité d'apparition. Le deuxième algorithme est basé sur la construction d'un dictionnaire pour la compression. Il demande en moyenne plus de temps et de ressources que l'algorithme de Huffman, mais il est globalement plus performant que ce dernier.

3.3 Chiffrement

Notre outil proposera la possibilité de chiffrer l'archive crée pour sécuriser la sauvegarde. Nous devons donc utiliser des algorithmes de chiffrement.

Algorithmes de chiffrement de données

Ils existent 2 types d'algorithmes de chiffrement : les algorithmes de chiffrement symétrique et les algorithmes de chiffrement asymétrique.

Nous utiliserons dans notre projet les algorithmes suivants :

Les algorithmes de chiffrement asymétrique

RSA

Le chiffrement RSA est un algorithme de cryptographie asymétrique, très utilisé dans le commerce électronique, et plus généralement pour échanger des données confidentielles sur Internet. Cet algorithme a été décrit en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman.

Le chiffrement RSA est asymétrique : il utilise une paire de clés composée d'une clé publique pour chiffrer et d'une clé privée pour déchiffrer des données. Les deux clés sont créées au même moment. La clé publique est utilisée pour chiffrer les données. La clé privée permet de déchiffrer ces données. La clé privée peut aussi être utilisée pour signer une donnée, la clé publique permettant à n'importe qui de vérifier la signature.

Une condition indispensable est qu'il soit mathématiquement impossible de déchiffrer à l'aide de la seule clé publique, en particulier de reconstituer la clé privée à partir de la clé publique.
RSA utilise les congruences (deux entiers relatifs sont congrus modulo n s'ils ont le même reste dans la division euclidienne par n) sur les entiers et le petit théorème de Fermat.
Cet algorithme de chiffrement est sûr seulement si la clé est assez longues ce qui rend une attaque par factorisation impossible.

Cryptosystème de ElGamal

Le cryptosystème d'ElGamal est un protocole de cryptographie asymétrique inventé par Taher Elgamalen 1984 et construit à partir du problème du logarithme discret.

Ce protocole est utilisé par le logiciel libre GNU Privacy Guard dont les versions récentes implantent jusque sa version sur les courbes elliptiques.

C'est un algorithme de chiffrement asymétrique. Le cryptosystème est composé de trois algorithmes (il suit le même principe que RSA) : génération des clés, Chiffrer et Déchiffrer.

Le problème de décision de Diffie-Hellmann (ou DDH) garantit la sécurité sémantique [1] du cryptosystème.

Les algorithmes de chiffrement symétrique

Advanced Encryption Standard (AES)

Advanced Encryption Standard ou AES un algorithme de chiffrement symétrique. Il remporta en octobre 2000 le concours AES, lancé en 1997 par le NIST et devint le nouveau standard de chiffrement pour les organisations du gouvernement des États-Unis. Il est actuellement le plus utilisé et le plus sûr.

Cet algorithme prend en entrée un bloc de 128 bits et la clé de 128, 192 ou 256 bits.

Les bits du bloc sont chiffrés via des permutations sur les bits du bloc et grâce à la fonction XOR [2] répété sur plusieurs tours.

1. La sécurité sémantique regarde quelles sont les informations que l'on peut obtenir à partir du fichier chiffré à propos du fichier original sans le détenir.
2. Fonction ou-exclusive

3.4 L'interface graphique

Un logiciel de sauvegarde peut être compliquer à utiliser pour l'utilisateur. C'est pour cela qu'une interface graphique est plus que nécessaire.
Bien qu'elle ne doit pas être excessivement complète et qu'elle peut reste simple d'utilisation, l'interface devra tout de même fournir tous les outils nécessaire pour compresser et chiffrer les document désires.
De plus incorporer une barre de progression a notre interface pour informer l'utilisateur du temps restant est aussi implémentable, pour avoir une interface agréable a regarder et ne pas être vide durant l'exécution du programme. Les objectifs actuellement fixe pour une interface sont donc les suivant : Une interface simple et efficace réalisé grâce à la librairie GTK-3, donner accès a tous les outils nécessaire à l'utilisateur pour effectuer des sauvegardes, permettre à l'utilisateur de sélectionner les fichiers et document à compresser/chiffrer et créer une interface appréciable à regarder, pouvant informer l'avancement du programme après son exécution.

3.5 Site web

Le site web permettra à l'utilisateur de stocker des archives d'une taille limité ainsi que les clés de chiffrement si ce dernier le souhaite.
Une api sera donc réalisé afin de lier le site web à l'exécutable.

3.5.1 Développement

Le site web sera développé en python avec le framework Django. Il utilisera le duo nginx + uwsgi pour le côté serveur. Il comportera une page où seront disponible les rapports de soutenance ainsi que le cahier des charges, une page permettant de télécharger le projet, une module de gestion de compte, une page permettant d'uploader une archive ou de gérer les clés de chiffrement.

3.5.2 Hébergement

Le site web du projet sera hébergé sur le serveur d'Hyperion. En effet, son serveur hébergeant déjà un site en Django, il possède déjà la configuration côté serveur ; c'est-à-dire que le reverse proxy nginx et le serveur d'application uwsgi sont déjà configurés pour un site web développé sur Django.

3.6 Avancement prévu pour les prochaines soutenances

Voici un tableau d'avancement des tâches prévu pour les prochaines soutenances :

Tâches	Soutenance		
	n° 1	n° 2	n° 3
Système de fichier	75%	100%	100%
Sauvegarde	50%	75%	100%
Compression	30%	60%	100%
Chiffrement	30%	80%	100%
Site web	50%	80%	100%
Interface graphique	25%	60%	100%