GostCrypt

Problématique : Pourquoi et comment chiffrer vos données sensibles?

Rapport Projet Fin d’Etude (PFE)

Table des matières

[Table des matières 1](#_Toc473234335)

[Introduction 2](#_Toc473234336)

[État de l’art 3](#_Toc473234337)

[Solutions intégrées au système d'exploitation 3](#_Toc473234338)

[Solutions logicielles 4](#_Toc473234339)

[GostCrypt 5](#_Toc473234340)

[L’algorithme GOST 5](#_Toc473234341)

[Le mode XTS 6](#_Toc473234342)

[Réalisations du projet de fin d’étude 7](#_Toc473234343)

[Compréhension du code 7](#_Toc473234344)

[Doxygen: Ajout de commentaires 7](#_Toc473234345)

[Changer l'interface en Qt 9](#_Toc473234346)

[L'intégrité de GostCrypt 9](#_Toc473234347)

[L'intégrité de Veracrypt 11](#_Toc473234348)

[Conclusion 12](#_Toc473234349)

[Annexe 1 13](#_Toc473234350)

# Introduction

De nos jours, les systèmes d'information sont de plus en plus la cible d'attaques informatisées. Ces attaques ne sont pas bénignes, elles ont un but précis, voler des données, détruire des données et obtenir des rançons. Les entreprises sont les plus visées, mais il faut sensibiliser les utilisateurs ainsi que le nombre d'outils accessibles au public pour protéger leurs données.

La sécurisation des données doit donc se démocratiser, TrueCrypt est un projet libre qui avait ce but. TrueCrypt visait à donner au public une solution fiable, non contrôlé par un État, de chiffrement de disque dur et de données. GostCrypt, un fork de TrueCrypt a commencé en 2013. Le sujet de notre projet de fin d'études était d'améliorer le logiciel GostCrypt selon différents angles.

# État de l’art

Le 28 mai 2014, une annonce sur le site officiel annonce la fin du développement en prétextant la fin du support Windows XP. Par ailleurs, un autre message en rouge apparait sur le site signalant aux utilisateurs que TrueCrypt pourrait contenir des failles de sécurité. Il leur conseille alors de se tourner vers d'autres logiciels et des indications pour migrer ses données vers BitLocker, le logiciel de chiffrement de la firme de Redmond. Malheureusement, BitLocker était réputé pour avoir des portes dérobées pour les services de renseignements.

Ce logiciel de chiffrement open source, largement utilisé, faisait partie des outils considérés comme les plus solides en matière de cryptographie. Les développeurs du projet étaient restés anonymes, mais ceux-ci ont disparu presque du jour au lendemain sans laisser de traces, avec une dernière version publiée avant leur départ, généralement considérée comme douteuse. Depuis, aucune nouvelle ou explication n’a été donnée, mais beaucoup soupçonnent l’action d’un gouvernement pour contraindre les développeurs à saboter ou abandonner le projet.

## Solutions intégrées au système d'exploitation

Plusieurs solutions sont directement intégrées au système d’exploitation, nous vous présenterons BitLocker, FileVault et LUKS.

BitLocker est un système de sécurité intégré à Windows et qui permet de chiffrer les données stockées sur votre ordinateur, ce qui permet d’assurer la confidentialité de vos données personnelles en cas de vol. Par défaut, l’AES 128 bits est utilisé, sans possibilité de le modifier lors de l’activation de BitLocker sur un lecteur. Il est possible d'utiliser l'AES 256bits.

FileVault est un système de protection des fichiers sur les ordinateurs Macintosh (Apple). Il utilise un système de fichiers chiffré qui est monté et démonté à la connexion et déconnexion de l'utilisateur au système. Le répertoire personnel de l'utilisateur est chiffré/déchiffré en utilisant l'algorithme AES. Le chiffrement se fait au niveau hardware dans un chipset propriétaire d'Apple.

LUKS (Linux Unified Key Setup) est le standard associé au noyau Linux pour le chiffrement de disque créé par Clemens Fruhwirth.Il permet de chiffrer l'intégralité d'un disque de telle sorte que celui-ci soit utilisable sur d'autres plates-formes et distributions de Linux (voire d'autres systèmes d'exploitation). Il supporte des mots de passe multiples, afin que plusieurs utilisateurs soient en mesure de déchiffrer le même volume sans partager leur mot de passe. Il est possible de choisir l'algorithme de chiffrement (AES - Serpent - Cast5/6 - Twofish) ainsi que le padding XTS en fin de bloc.

## Solutions logicielles

VeraCrypt est un logiciel utilitaire sous licence libre utilisé pour le chiffrement à la volée (OTFE). Il est développé par la société française IDRIX2 et permet de créer un disque virtuel chiffré dans un fichier ou une partition.

Il est devenu le logiciel open source de chiffrement le plus populaire depuis l’arrêt du support officiel de TrueCrypt.

De même, un audit de sécurité a aussi débuté après la sortie de sa dernière édition (1.18), L’OSTIF (Open Source Technology Improvement Fund) a pu embaucher des chercheurs de chez QuarksLab pour débusquer d’éventuelles vulnérabilités ou faiblesse au sein du code source de VeraCrypt. Lors de cet audit, il a été détecté un problème d'implémentation de l'algorithme GOST. Plus tard VeraCrypt redéploiera cet algorithme modifié sous le nom de Kuznyechik.

Pour son étude, Quarkslab est parti de l'hypothèse d'un ordinateur portable perdu ou volé, dont un attaquant voudrait déchiffrer le disque dur. Au total, 26 failles ou soucis ont été trouvés, dont huit vulnérabilités critiques, qui demandaient à être corrigées avant d'être révélées. La vaste majorité d'entre eux ont été éliminés avec la version 1.19. Voici les algorithmes de chiffrement qu’il utilise:

* AES
* Serpent
* Twofish (Évolution du Blowfish)
* AES – Twofish
* AES – Twofish – Serpent
* Serpent – AES
* Serpent – Twofish – AES
* Twofish – Serpent

# GostCrypt

Développé par l’Université des Sciences Appliquées d’Amsterdam, le Bauman Moscow State Technical University et le Laboratoire de Cryptologie et de Virologie Opérationnelles ESIEA à Laval, GostCrypt reprend les sources de TrueCrypt, supprime ses implémentations des algorithmes de chiffrement et implémente l’algorithme de chiffrement russe : GOST.

L'algorithme GOST, plus exactement GOST 28147-89, l'équivalent russe du standard américain DES. De même, les fonctions de hachage russes GOST R34.11-2012 et GOST R34.11-94 sont disponibles (et toujours Whirlpool, d'origine européenne).

C’est sous l’impulsion d’Eric Filiol que le développement de GostCrypt commence en 2013. En voulant rendre hommage à TrueCrypt, Eric Filiol reprend le code source de la version 7.1a, mais change une partie majeure : les algorithmes de chiffrement.

## L’algorithme GOST

Il existe donc plusieurs solutions pour chiffrer des données personnelles à caractère sensible. Ce ne sont pas les moyens qui manquent. Des solutions gratuites ou payantes sont disponibles, mais ce qu'il faut se demander c'est plutôt quelles solutions sont le plus à l’abri d'erreurs, de failles, de backdoors et où iront les données si ces données sont déchiffrables.

GostCrypt implémente un algorithme russe du nom de GOST 28147-89. Cet algorithme par chiffrement par bloc. Il a été créé par et pour la Russie ce qui, pour Eric Filiol, est un signe que cet algorithme est plus sûr qu’AES. En effet, on voit mal un gouvernement faire un algorithme de chiffrement pour leur propre sécurité et inclure des backdoors mathématiques.

GOST 28147-89 est un algorithme de chiffrement équivalent au DES Américains cependant GostCrypt implémenterait une version plus robuste de l’algorithme russe.

Contrairement à DES, GOST 28147 n’a pas subi beaucoup de cryptanalyse. Il n’a donc été que très peu éprouvé.

L'algorithme de chiffrement par blocs GOST 28147-89 utilise une clé de 256 bits. En interne, cette clé est divisée en sous-clés de 32 bits. Lors du processus de mise à la clé, la S-Box interne est modifiée à l'aide de ces sous-clés, apportant un premier degré de diversification de l'algorithme par l'utilisateur (et donc une résistance accrue face aux attaques connues qui supposent un algorithme fixe).

Pour cela, le paramètre S-box 'GOST R 34.11-94 CryptoProParamSet' du RFC 4357 est utilisé comme S-Box initiale.

En cryptographie, une S-box (substitution-box) est un composant de base des algorithmes de clé symétrique qui effectue la substitution. Dans les chiffres de bloc, ils sont généralement utilisés pour masquer la relation entre la clé et le texte chiffré.

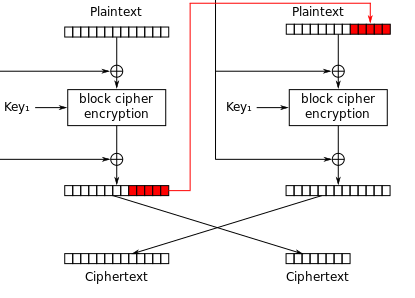
En général, un S-box prend un certain nombre de bits d'entrée, m, et les transforme en un certain nombre de bits de sortie, n, où n n'est pas nécessairement égal à m [1] Un m × n S-box peut être implémenté comme une table de consultation avec 2m mots de n bits chacun. Les tables fixes sont normalement utilisées, comme dans le Data Encryption Standard (DES), mais dans certains chiffrages les tables sont générées dynamiquement à partir de la clé comme pour GostCrypt

Sa clé secrète de 256 bits est d'abord hachée à l'aide de la fonction de hachage GOST R 34.11-2012, laquelle produit une valeur de 512 bits. Comme la S-Box initiale et cette valeur sont de la même taille (512 bits), on opère une addition bit à bit modulo 2 sur les entrées de 4 bits de la S-Box (voir le schéma ci-après). La S-Box résultante, laquelle est donc clé-dépendante, est utilisée pour le chiffrement et le déchiffrement. Durant les opérations de chiffrement/déchiffrement, GOST 28147-89 est utilisé en mode XTS. L'identifiant d'unité de donnée (un offset disque dans le mode XTS) est combiné bit à bit modulo 2, avec la clé GOST 28147-89. Cela assure que la clé réelle (clé initiale et marquant de données) est différente pour chaque section de 512 octets du disque, assurant ainsi une résistance contre toutes les attaques connues, lesquelles nécessitent de grands volumes de données produites avec la même clé.

## Le mode XTS

XTS est une méthode de substitution lorsque la taille du secteur qui doit être chiffré n’est pas divisible par la taille du bloc de chiffrement. Elle fonctionne comme suivant en prenant l’exemple d’un secteur de 24 octets et un bloc de chiffrement de 16 octets:

On chiffre les 16 premiers octets. On coupe à ces 16 octets maintenant chiffrés les 8 derniers. On ajoute au deuxième bloc que l’on doit chiffrer ces 8 octets coupés. On chiffre les 16 octets que l’on vient de fabriquer. On les place au début du chiffré final.  
On place ensuite les 8 octets du premier bloc chiffré restant que l’on place à la fin du chiffré final.  
Le processus est résumé par le schéma suivant :

*Figure 1 : Fonctionnement XTS (Wikipédia)*

# Réalisations du projet de fin d’étude

## Compréhension du code

La première étape importante dans ce projet était de comprendre l'intégralité du code source de GostCrypt le plus rapidement possible. Cette étape doit être appliquée par chacun des membres du groupe. Guillaume Swaenepoel a travaillé sur GostCrypt avant nous, et a déclaré que l'étape d'appréhension du logiciel et de sa structure peut s'avérer très longue, mais est essentiel. Nous avons donc commencé par analyser le code à deux pour que nous ayons chacun cette connaissance.

Le projet GostCrypt est découpé en 6 sous projet: Boot, Crypto, Driver, Format, Mount et Setup. On retrouve également un dossier Common rassemblant des sources nécessaires à chaque sous-projet.

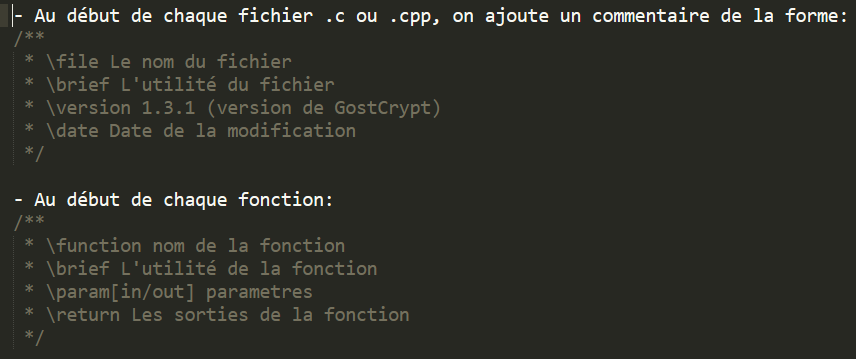
Ce n'est qu'au bout d'une semaine de recherche que nous avons commencé à comprendre son fonctionnement : le sous-projet Setup permet de créer une archive exécutable où sera exécuté le reste du code.

Nous avons alors décidé de nous séparer pour comprendre le plus rapidement possible le fonctionnement de chaque sous-projet. Le but de cette étape était schématiser le fonctionnement de chaque sous-projet: On regarde ce que fait la fonction main () du projet et on dessine l'appel aux fonctions. Vous retrouverez en annexe 1 ces schémas que nous avons pu mettre en place.

## Doxygen: Ajout de commentaires

L'un des objectifs du PFE était de commenter entièrement le code source tout en respectant le standard de Doxygen. C'est un générateur de documentation sous licence libre capable de produire une documentation logicielle à partir du code source d'un programme. Pour cela, il tient compte de la grammaire du langage dans lequel est écrit le code source, ainsi que des commentaires qui sont écrits dans un format particulier.

Voici la forme des commentaires qui ont été ajoutés:



*Figure 2: Commentaires selon la structure de Doxygen*

Il faut savoir que cette partie a été la plus longue dans ce PFE, il y a au total plus de 65000 lignes de code à comprendre pour commenter correctement. Ce n’est pas toujours évident de comprendre le code source ayant été écrit par plusieurs personnes et ne respectant donc pas les mêmes conventions partout.

Il est important d’avoir du code bien commenté. Les intérêts sont multiples et variés, mais on s’intéressera ici principalement à la documentation d’une application.

- Commenter pour comprendre ce qu’on a fait : Les personnes qui travaillent sur GostCrypt ont souvent tendance à changer, et il est important pour chacun de pouvoir participer au projet le plus rapidement possible.

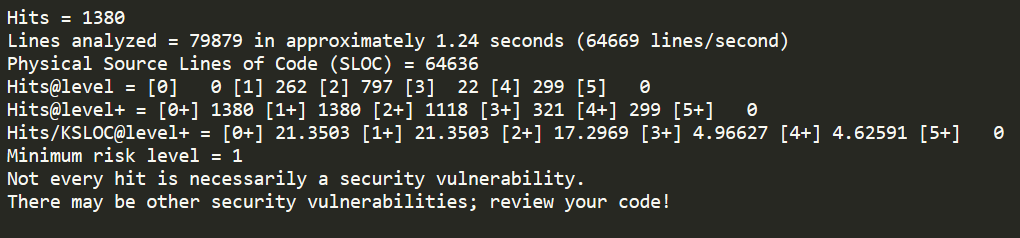
- Commenter pour se faciliter la tâche : Doxygen est capable de s’en servir pour créer une documentation ou afficher la cartographie du projet. Dans le cadre du développement de GostCrypt, les commentaires révèlent toute leur puissance en vous évitant de replonger dans le code d’une fonction pour retrouver quels arguments lui passer ou encore ce qu’elle retourne.

Doxygen peut utiliser Graphviz pour afficher tous les appels aux fonctions : il permet d’afficher sous forme de graphe les appels des fonctions, mais aussi le nom des fonctions qui les appels. En voici un exemple simple :



*Analyse statique et dynamique du code (effacer le code inutile)*

Nous avons réalisé une analyse statique de l’ensemble des .c et .cpp des sources. Il y a un total de 1380 failles (dont 299 failles de niveau 5). Voici le résumé du rapport de l’analyse statique par FlawFinder :



*Figure 3 : Résumé du rapport de FlawFinder*

## Changer l'interface en Qt

Le changement de l'interface en Qt n'a pas pu être réalisé comprenant deux obstacles majeurs. Premièrement il nous aurait fallu comprendre comment fonctionne l'interface utilisée par Windows en afin de ne rien oublier. Le deuxième étant le temps qu'il aurait fallu pour s'initier au Qt afin de produire une interface stable. Nous aurions pu pallier à ce problème en utilisant l'interface produite pour GostCrypt sous Linux par Guillaume Swaenepoel.

## L'intégrité de GostCrypt

Au cours de notre progression sur la compréhension globale du projet, nous avons été amenés à nous questionner sur l'intégrité dans le logiciel. Nous ne nous attaquerons pas au chiffrement puisque celui-ci est vérifié grâce à un checksum et intègre. Voici ce qu'il se passe lors du montage d'un volume :  
  


*Figure 4 : Comment GostCrypt monte un volume*

En effet, lorsque l'on monte un volume, la première chose demandée est le mot de passe. Le logiciel ne sait pas quel algorithme a été utilisé pour chiffrer le volume. Il déchiffre alors une partie du header en utilisant chaque algorithme avec le mot de passe pour clé. Si le déchiffrement fonctionne sur cette partie du header, il déchiffre le volume en entier. Si aucun des déchiffrements ne fonctionne, soit le mot de passe est erroné, soit le header a été modifié et ne peut plus être déchiffré correctement.



*Figure 5 : Message d'erreur ouverture de volume*

Dans le cas où l'on entre le bon mot de passe, mais qu'on a modifié une partie chiffrée du volume qui ne correspond pas au header, GostCrypt va déchiffrer le volume sans regarder l'intégrité du volume. Les données de l'utilisateur ont alors été modifiées sans même qu'il ne puisse s'en rendre compte.  
  


*Figure 6 : Non-intégrité de GostCrypt*

Voici jusqu'où peut aller le problème : Une personne qui a connaissance de la cartographie du volume, peut, cibler un fichier, et, sans ouvrir le volume détruire ou modifier partiellement ce fichier. Il ne pourra pas savoir quel impact a eu son attaque sur ce fichier, mais il pourrait dans le cas d'une signature sur un document l'effacer. Si l'on imagine que cette personne a changé de travail. Elle donne l'accès au volume après vérification des données et changement du mot de passe pour le prochain utilisateur. Ensuite excusée puisqu'elle n'a pu accès au volume, attaque la cible et modifie le volume sans en avoir le mot de passe. Il détruit des preuves tout en étant excusé.

Afin de remédier à ce problème, nous avons proposé une solution permettant d'avertir l'utilisateur en cas de modification de données. Lorsque le volume est chiffré, il faut ajouter le hash du volume (Grâce au hachage fourni par GOST par exemple) dans le header. Si le header est modifié ou si un élément du conteneur est modifié, le hash ne sera alors pas le même. Il faut ensuite ajouter une boite de dialogue pour prévenir l'utilisateur de données modifiées tout en lui laissant le choix de récupérer les données dans le volume.

## L'intégrité de Veracrypt

VeraCrypt étant aussi d'une base venant de TrueCrypt nous avons voulu rejouer cette attaque sur un volume chiffré par VeraCrypt. Le résultat est le même : il y a possibilité de modifier partiellement (sans savoir l'impact réel) une ou des données du volume. VeraCrypt est beaucoup plus avancé dans son fonctionnement et dans l'implémentation de nouveaux algorithmes et fonctionnalité, mais des problèmes persistent.  
  
 Il serait intéressant pour GostCrypt de travailler sur cette faille: le logiciel cherche souvent à s’affranchir définitivement de la licence TrueCrypt et VeraCrypt par la même occasion, résoudre ce problème contribuerait à les différencier un peu plus

# Conclusion

Pour chiffrer des données personnelles il y a un problème pour le moment, à qui faire confiance, entre le chiffrement AES que le gouvernement américain n'utilise pas forcément, mais essaye d'en faire un standard. BitLocker de Windows qui est susceptible de contenir des backdoors pour les services de renseignement. Il existe de bonnes alternatives gratuites et libres comme VeraCrypt qui est, pour le moment, le logiciel libre le plus avancé après la disparition de TrueCrypt. Mais nous pouvons voir qu'ils sont toujours très attachés à cette base de TrueCrypt qui comprend quelques faiblesses. GostCrypt pourrait devenir un bon logiciel du même genre si ces petits défauts sont corrigés, si la licence se sépare de VeraCrypt et prend son propre chemin dans une optique multiplateforme utilisant un algorithme qui n'est pas développé par les Américains.

# Annexe 1

Voici les schémas que nous avons réalisés pour expliquer les différents sous-projets à partir de leur main ().

