



Institut National
Universitaire
Champollion

Projet de Cryptographie

Compte rendu

Gaël Burguès
Laurian Dufrechou
Lucàs Vabre

Proposé par: Monsieur POUIT

Table des matières

1. Introduction	3
2. Situation du projet	3
2.1. Version à 16 boucles	3
2.2. Des simple	4
2.3. Triple Des	5
2.4. Interface Graphique	5
2.4.1. Menu	5
2.4.2. Fenêtre Des	5
2.4.3. Fenêtre Triple Des	6
3. Conclusion	6

1. Introduction

Au cours de notre 5ème semestre de Licence Informatique, Monsieur POUIT nous a proposé un projet lors de ses cours de réseau. Celui-ci avait pour but de nous initier à la cryptographie avec notamment l'algorithme du DES (Data Encryption Standard). Nous devons recréer cet algorithme sous Java.

2. Situation du projet

Notre projet a compté 4 versions différentes. Nous avons commencé par une version effectuant 16 boucles avec 16 clés (voir algorithme du DES). Beaucoup de problèmes se sont accumulés sur cette version, ce qui nous a poussé à la mettre en suspend pendant que nous essayons de réaliser un version du Des en une seule boucle et clé. La 3ème partie de ce projet est la version `TripleDes` qui utilise trois instances différentes de la classe DES pour crypter et décrypter. Et enfin la 4eme version consistait à réaliser une interface graphique afin de rendre accessible aux utilisateurs les classes programmées.

2.1. Version à 16 boucles

Pour gagner du temps nous avons décidé de partir sur la deuxième version du Des demandé sur le TP. Notre objectif était de créer les classes du Des avec une méthode `cryptage` qui génère 16 clés dans un tableau puis effectuait les permutations en fonction des clés. Seulement, nous nous sommes heurtés à plusieurs problèmes que nous n'avions pas prévus.

Premièrement notre classe `decoupage` ne prenait en compte que des tableaux de taille 64, de ce fait seuls les messages de 64 bits étaient autorisés, or tous les messages ne font pas la même taille. Nous avons donc ajouté des bits 0 à la fin des messages une fois convertis en binaire. Malheureusement, lorsque le message faisait enfin 64 bits, nous avons découvert un nouveau problème: notre algorithme ajoutait 8 entrées au tableau, rempli de 0.

Ensuite, notre `fonction_F` ne fonctionnait pas correctement. Car bien que tout au long de notre projet nous ayons utilisé JUnit pour tester nos méthodes de manière unitaires, la `fonction_F` dépendait beaucoup du hasard pour être testé

automatiquement, c'est pourquoi il a été fastidieux de corriger les différents bugs liés à cette classe.

Enfin la fonction `recollage` permettait de recoller uniquement le 1er tableau, découpé dans le message, à ce même tableau. Ce qui nous donnait comme résultat des message comme : `a@--§-ta@--§-ta@---ta@--§-t`

Face à tous ces problèmes nous nous sommes rendu compte de notre arrogance à vouloir sauter les étapes, et avons repris le projet de zéro, en commençant par le simple `Des` (comportant une seule boucle), que nous pourrions améliorer dans un second temps avec les 16 boucles.

2.2. Des simple

Dans notre version du `Des` simple. Nous nous sommes concentré sur les points qui ont péchés dans notre version précédente. Nous avons modifié la fonction `decoupage` pour qu'elle découpe n'importe quel tableau en un nombre de blocs donnés. Si la taille du tableau n'est pas un multiple du nombre de blocs voulu, nous calculons le nombre de 0 à rajouter à la fin de celui-ci pour qu'il soit multiple. Le tableau et le potentiel surplus de 0 à rajouter sont d'abord découpés puis stockés dans un nouveaux tableaux à double entrée.

Nous avons ensuite modifié la fameuse fonction `F`. Dans la version précédente nous n'avons pas utilisé de tableau de 48 bits. Par la suite, nous avons pu grâce à ce tableau et à la nouvelle fonction `decoupage` pu se s'intéresser à la fonction `S`. Cette fonction ne s'applique qu'une seule fois, elle ne fait qu'une seule boucle qui est générée aléatoirement.

Grâce à cela nous avons une fonction `cryptage` qui fonctionne. Puis nous avons découvert quelques soucis mineurs lors des tests, mais sans erreurs critiques. Et c'est ainsi que nous nous sommes intéressés à la fonction `decrypte` faisant tout naturellement, le travail inverse de la fonction `crypte`.

C'est au moment où la fonction `decrypte` ne fonctionne pas que l'on a conclu que `crypte` possédait aussi un défaut. Nos soupçons se sont portés sur les fonctions auxiliaires. Nous avons donc dû re-vérifier chaque fonction en détails jusqu'à ce que l'on tombe sur la fonction `stringToBits`. Celle-ci permet de convertir une chaîne de caractères en un tableau de bits en utilisant l'encodage ASCII. Normalement, un caractère ASCII n'as besoin que de 7 bits pour être représenter, seulement nous avons ajouté un 8ème bits de parité. Ce bit de parité était surtout utile dans le cas d'un

transfert de données, seulement dans notre cas il ne nous semblait pas utile. Nos suppositions étaient fausses, utiliser 8 bits permettait de normaliser l'écriture des caractères au lieu de prendre le minimum de bits. En effet, normalement les bits 0 en tête ne sont pas obligatoires. Nous nous sommes rendu compte plus tard que les chiffres en ASCII n'avaient besoin que de 6 bits pour être représenté. Au final, une fois cette méthode corrigée, les méthodes crypte et decrypte fonctionnaient parfaitement.

2.3. Triple Des

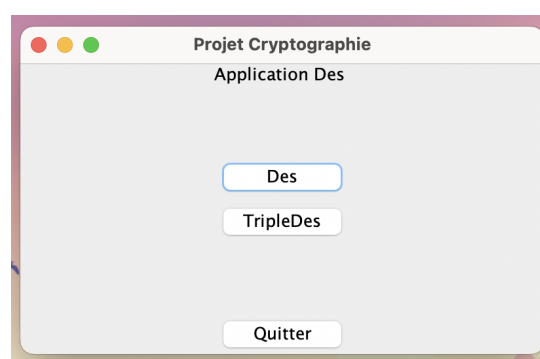
Concernant la classe `TripleDes` nous lui avons simplement attribué les méthodes `crypte` et `decrypte` puis nous avons utilisé trois instances de `Des` pour les utiliser. Dans `crypte` nous appelons la fonction `crypte` de chaque `Des` de la classe (3 en l'occurrence). Et de la même manière pour `decrypte`, on vient appeler la fonction `decrypte` le nombre de `Des` en commençant par le dernier.

2.4. Interface Graphique

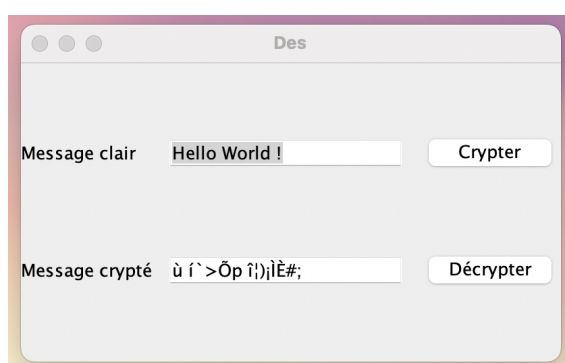
Pour compléter ce projet, nous avons réalisé une interface graphique, composée de 3 fenêtres.

2.4.1. Menu

La première est un menu, elle permet de choisir quel type de chiffrement nous souhaitons utiliser, `Des` ou `Triple Des`. Elle possède aussi un bouton pour quitter l'application.



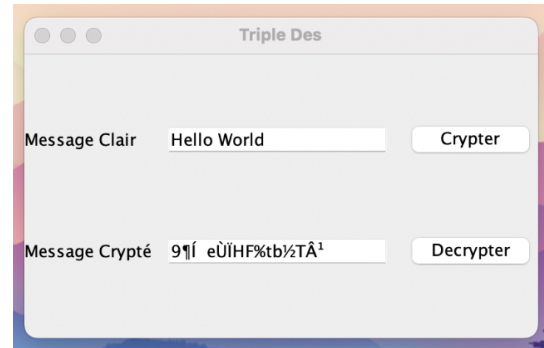
2.4.2. Fenêtre Des



Dans le cas où notre choix de chiffrement est le `Des`, une nouvelle fenêtre apparaît et nous permet de saisir un message, le crypter puis le décrypter.

2.4.3. Fenêtre Triple Des

Si l'utilisateur choisit le triple Des, alors une fenêtre semblable à la précédente apparaît et permet de crypter et décrypter un message avec le Triple Des



3. Conclusion

Ce projet a été complet et très formateur. Il nous a permis de nous familiariser avec la cryptographie et plus particulièrement le chiffrement DES. Celui-ci nous a montré les différents tenants et aboutissants d'un algorithme de chiffrement et les difficultés que l'on peut rencontrer comme par exemple les tests unitaires des fonctions aléatoires.