[Titre du document]

[Attirez votre lecteur avec un résumé attrayant. Il s’agit généralement d’une brève synthèse du document. Lorsque vous êtes prêt à ajouter votre contenu, cliquez ici et commencez à taper.]

[Sous-titre du document]

Tables des matières

[I. Introduction 2](#_Toc128576948)

[1. Objectif du projet 2](#_Toc128576949)

[2. Description de la méthode de cryptage Solitaire 2](#_Toc128576950)

[II. Analyse et conception 3](#_Toc128576951)

[1. Analyse des besoins 3](#_Toc128576952)

[a. Fonctionnalités attendues 3](#_Toc128576953)

[b. Utilisateurs de l’application 3](#_Toc128576954)

[c. Sécurité 3](#_Toc128576955)

[d. Performance 3](#_Toc128576956)

[2. Conception de l’application 4](#_Toc128576957)

[a. Diagramme de classe et structure de données 4](#_Toc128576958)

[b. Conception de l’algorithme 5](#_Toc128576959)

[c. Conception de l’interface utilisateur\* 6](#_Toc128576960)

[d. Tests 8](#_Toc128576961)

[III. Conclusion 8](#_Toc128576962)

[1. Bilan du projet 8](#_Toc128576963)

[2. Difficultés rencontré 8](#_Toc128576964)

[3. Perspectives d'amélioration 8](#_Toc128576965)

# Introduction

## Objectif du projet

Le projet consiste à utiliser la méthode de cryptographie Solitaire, conçue par le spécialiste de cryptographie Bruce Schneier, qui permet de coder des messages à l'aide d'un jeu de 54 cartes. L'objectif de ce projet est de proposer une méthode de cryptage fiable, équivalente aux meilleurs algorithmes utilisés aujourd'hui, et de concevoir un procédé de création de suites pseudo-aléatoires afin de produire une clé pseudo-aléatoire en continu. Cette méthode de cryptage par addition d'un message avec une clé est considérée comme la seule méthode de cryptographie mathématique prouvée sûre d'une manière absolue. Le projet a tout de même une contrainte, la clé générée doit être de la même longueur que le message à envoyer. Ainsi, la méthode de Schneier peut être utilisée pour crypter des messages avec une sécurité optimale. Il en va de même que si nous pouvons crypter un message, nous avons également la possibilité de décrypter un message en utilisant non plus une addition mais une soustraction du message avec la clé de cryptage.

## Description de la méthode de cryptage Solitaire

La méthode de cryptage Solitaire est basée sur l'utilisation d'un jeu de cartes pour générer une suite pseudo-aléatoire, qui sera utilisée comme clé de chiffrement. Cette suite est générée en utilisant des mouvements de cartes spécifiques, que nous détaillerons plus tard, qui sont déterminés par un algorithme précis. Cette suite est utilisée pour chiffrer le message en additionnant chaque caractère du message avec la valeur numérique des cartes présente dans la clé de chiffrement.

Pour déchiffrer le message, il suffit d'utiliser la même suite de clés, c’est-à-dire la clé de cryptage pour soustraire chaque caractère chiffré. La sécurité de cette méthode repose sur la complexité de la génération de la suite de clés pseudo-aléatoires, qui rend difficile la détermination de la clé par une personne qui ne possède pas l’ordre du paquet initial. En d’autres termes, l’état du paquet juste avant de commencer à réaliser les manipulations. De plus, cette méthode est particulièrement robuste face aux attaques de type brute force, qui consiste à essayer toutes les combinaisons possibles de clés pour déchiffrer le message.

# Analyse et conception

## Analyse des besoins

### Fonctionnalités attendues

Pour les fonctionnalités attendues de l'application, il est nécessaire de pouvoir générer une clé pseudo-aléatoire de la même longueur que le message à crypter, ainsi que de pouvoir crypter et décrypter des messages à l'aide de cette clé. Nous disposerons également d'une interface utilisateur conviviale pour faciliter l'utilisation de l'application avec notamment la possibilité d’importer et exporter ses clés de base.

### Utilisateurs de l’application

Pour simuler l'utilisation de notre application, nous avons supposé qu'elle serait utile à toute personne souhaitant chiffrer ou déchiffrer un message. La seule condition est que les deux personnes doivent disposer de l'application sur leur ordinateur.

### Sécurité

La sécurité est un élément clé de ce projet. Il est crucial de garantir la confidentialité des messages échangés en utilisant une méthode de cryptage fiable et éprouvée, ainsi qu'en veillant à la sécurité des clés de cryptage générées par l'application.

Étant donné que nous avons choisi de ne pas utiliser de communication réseau, les problèmes de sécurité devraient être limités. Cependant, il est important de s'assurer que la clé de base de la communication entre les deux personnes reste confidentielle. Si quelqu'un découvre l'ordre initial des cartes, la sécurité des messages cryptés pourrait être compromise. Notre application propose la possibilité d’exporter et d’importer l’ordre de son paquet, nous invitons nos utilisateurs a se communiqué cet ordre face à face afin de limiter les risques d’interception de celui-ci.

### Performance

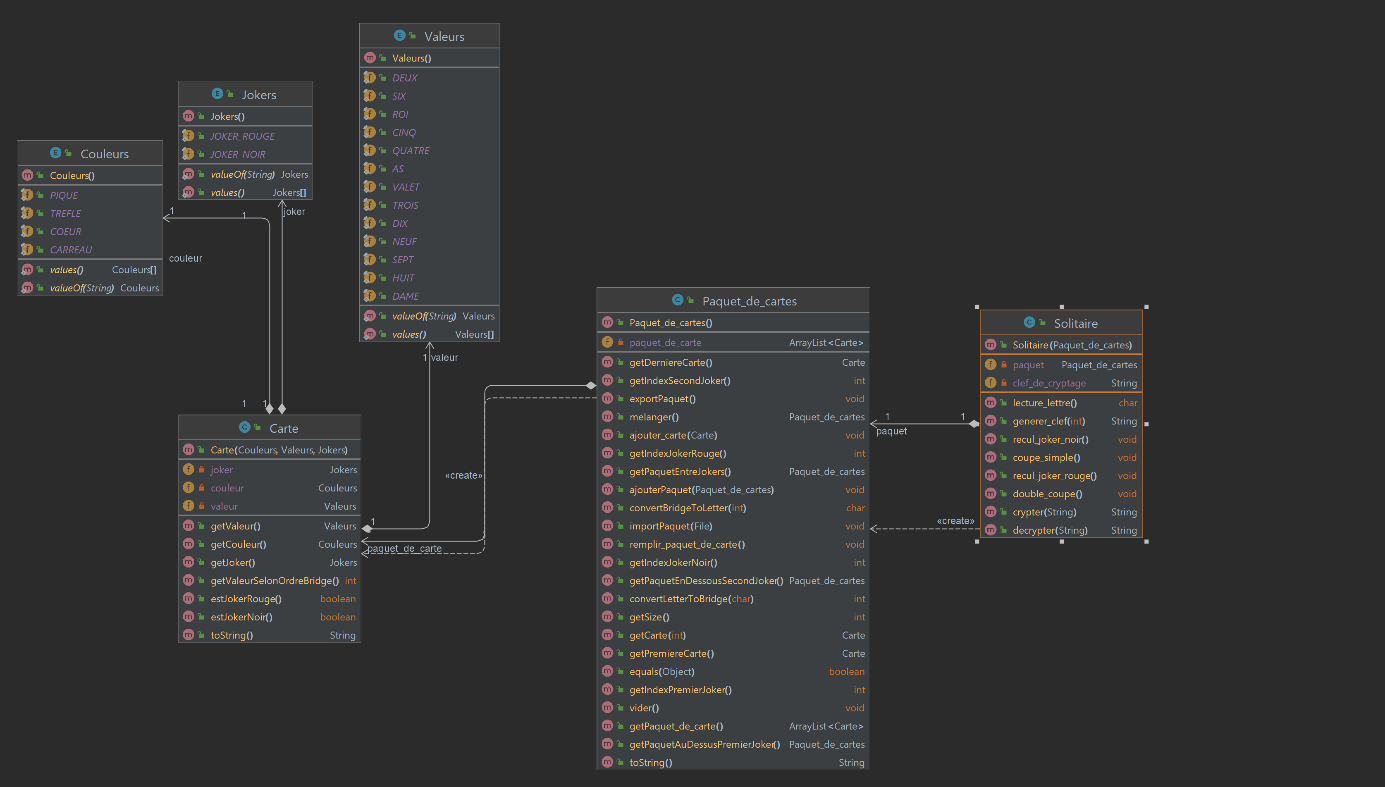
En ce qui concerne les performances, l'application devra être en mesure de générer rapidement des clés pseudo-aléatoires de grande longueur, tout en assurant un temps de cryptage et de décryptage raisonnable pour les messages à échanger.

## 

## Conception de l’application

### Diagramme de classe et structure de données

Avant de traiter l’algorithme, regardons rapidement notre diagramme de classe afin de comprendre la structure de notre projet.



: Diagramme de classe

Nous précisons, que pour une question de lisibilité, sur ce diagramme il n’y a pas les classes concernant l’interface de l’application.

Comment avons-nous donc structuré nos données ?

Nous avons choisi de structurer nos données à l'aide d'énumérations pour représenter une carte, plutôt que d'utiliser une valeur entière selon l'ordre du bridge. Cette approche offre plusieurs avantages. Tout d'abord, elle permet une meilleure lisibilité et compréhension du code en utilisant des noms significatifs pour les valeurs et les couleurs des cartes, tels que "ROI" et "PIQUE". Ensuite, l'utilisation d'énumérations facilite la maintenance du code en évitant les erreurs liées aux valeurs numériques ou chaînes de caractères non valides. Enfin, cette approche offre une plus grande flexibilité pour gérer des cartes spéciales comme les Jokers, qui peuvent être représentés explicitement comme «JOKER\_ROUGE » et « JOKER\_NOIR »

Une fois que nous avons établi la structure de données pour représenter une carte, la question se pose de savoir quelle structure utiliser pour représenter un paquet de cartes. Nous avons choisi d'utiliser un ArrayList de carte pour plusieurs raisons. Tout d'abord, cette structure de données est dynamique, ce qui signifie que nous pouvons facilement ajouter ou supprimer des cartes du paquet en fonction des besoins de l'algorithme de cryptage. De plus, un ArrayList offre une grande flexibilité pour manipuler les éléments du paquet, tels que trier les cartes, les mélanger ou les distribuer. En outre, cette structure de données est largement utilisée dans la communauté Java et bénéficie d'un large support de la part de la documentation et des forums en ligne.

Comparé à d'autres structures de données qui pourraient également représenter un paquet de cartes, telles que les tableaux ou les listes chaînées, un ArrayList offre des performances élevées en termes de temps d'accès et de manipulation des éléments. De plus, l'ArrayList offre une capacité de stockage dynamique, ce qui signifie que la taille du paquet peut être modifiée à tout moment, ce qui peut être très utile dans le cadre de l'algorithme de chiffrement solitaire de Schneier. Enfin, l'utilisation d'un ArrayList facilite la modularité et la réutilisabilité du code en permettant une séparation claire entre la représentation des cartes et celle du paquet.

Enfin, nous avons décidé de séparer la logique de l'algorithme de chiffrement solitaire de Schneier de l'implémentation des cartes et du paquet. Pour ce faire, nous avons créé une troisième classe appelée "Solitaire" qui instancie un paquet de cartes et effectue toutes les étapes de l'algorithme sur celui-ci. Cette séparation des responsabilités entre les classes offre plusieurs avantages. Tout d'abord, elle permet une meilleure organisation et structuration du code en divisant la complexité de l'algorithme en plusieurs classes distinctes. Ensuite, elle facilite la maintenance du code en évitant les effets de bord lors de la modification d'une classe spécifique. Enfin, elle offre une plus grande flexibilité pour modifier ou remplacer la logique de l'algorithme sans affecter la représentation des cartes et du paquet.

### Conception de l’algorithme

Nous allons maintenant expliquer le fonctionnement de notre algorithme de cryptage et de décryptage. Nous avons besoin de trois fonctions :

* Générer une clé de cryptage
* Crypter un message
* Décrypter un message.

Comme la clé doit avoir la même longueur que le message envoyé, les étapes de l'algorithme doivent être présentes dans notre fonction de génération de clé, appelée "générer\_clef(int)", qui prend en paramètre la taille du message. Les cinq étapes pour générer le flux de clé de cryptage sont détaillées dans la partie 3 de l'énoncé[[1]](#footnote-1). Elles consistent à prendre le paquet de cartes face à vous avec l'ordre initial convenu avec votre correspondant, à reculer le joker noir d'une position, le joker rouge de deux positions, à effectuer une double coupe par rapport aux jokers, une coupe simple déterminée par la dernière carte et enfin à lire une lettre pseudo-aléatoire.

Une fois que nous avons notre clef de cryptage, nous pouvons crypter notre message, comment cela marche ?

* Le message est converti en minuscules et tous les espaces et caractères spéciaux sont supprimés.
* Une clé de chiffrement est générée à partir de la longueur du message.
* Une chaîne de caractères vide est initialisée pour stocker le message crypté.
* Pour chaque lettre du message, sa valeur est convertie en un nombre correspondant à sa position dans un jeu de cartes.
* La même conversion est appliquée à chaque lettre de la clé de cryptage.
* Les deux valeurs numériques sont ensuite ajoutées, et si le résultat est supérieur à 26, la soustraction de 26 est effectuée.
* La valeur numérique résultante est convertie en lettre correspondant à sa position dans le jeu de cartes à jouer.
* Cette lettre est ajoutée à la chaîne de caractères stockant le message crypté.
* Enfin, la chaîne de caractères contenant le message crypté est renvoyée.

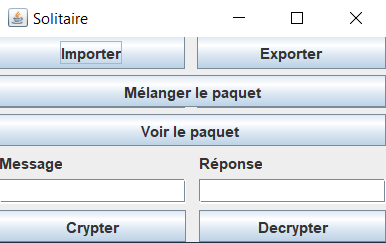
Pour décrypter le message c’est le même principe :

* Générer une clé de décryptage de la même longueur que le message à décrypter. Comme les deux utilisateurs auront le même ordre de départ, la même clé sera retrouvée.
* Parcourir chaque lettre du message à décrypter.
* Convertir la lettre du message et la lettre de la clé correspondante en leur valeur numérique correspondante.
* Soustraire la valeur numérique de la lettre de la clé à la valeur numérique de la lettre du message.
* Si le résultat est inférieur à 1, ajouter 26 pour revenir à la plage de valeurs 1-26.
* Convertir la valeur numérique résultante en la lettre correspondante.
* Ajouter cette lettre au message décrypté.
* Répéter pour chaque lettre du message.
* Retourner le message décrypté.

Ainsi nous avons la possibilité de pouvoir envoyer a notre partenaire un message crypté mas également de pouvoir décrypter ses messages, nous pouvons donc avoir une conversation en totale sécurité.

### Conception de l’interface utilisateur\*

Concernant la conception de notre interface utilisateur, nous avons voulu faire quelque chose de simple mais conviviale à utiliser.



: Vue principale de l'application

Nous avons plusieurs possibilités :

* Importer le paquet de son partenaire afin de pouvoir décrypter ses messages.
* Exporter son paquet, afin de donner la possibilité a votre partenaire de décrypter vos messages.
* Mélanger le paquet, afin d’avoir un ordre du paquet aléatoire que vous pourrez par la suite, s’il vous convient, exporter.
* Voir le paquet
* Crypter un message.
* Décrypter un message.

Notez que l’importation et l’exportation se fait via des fichiers « .txt » formaté de cette façon :

Une image contenant texte

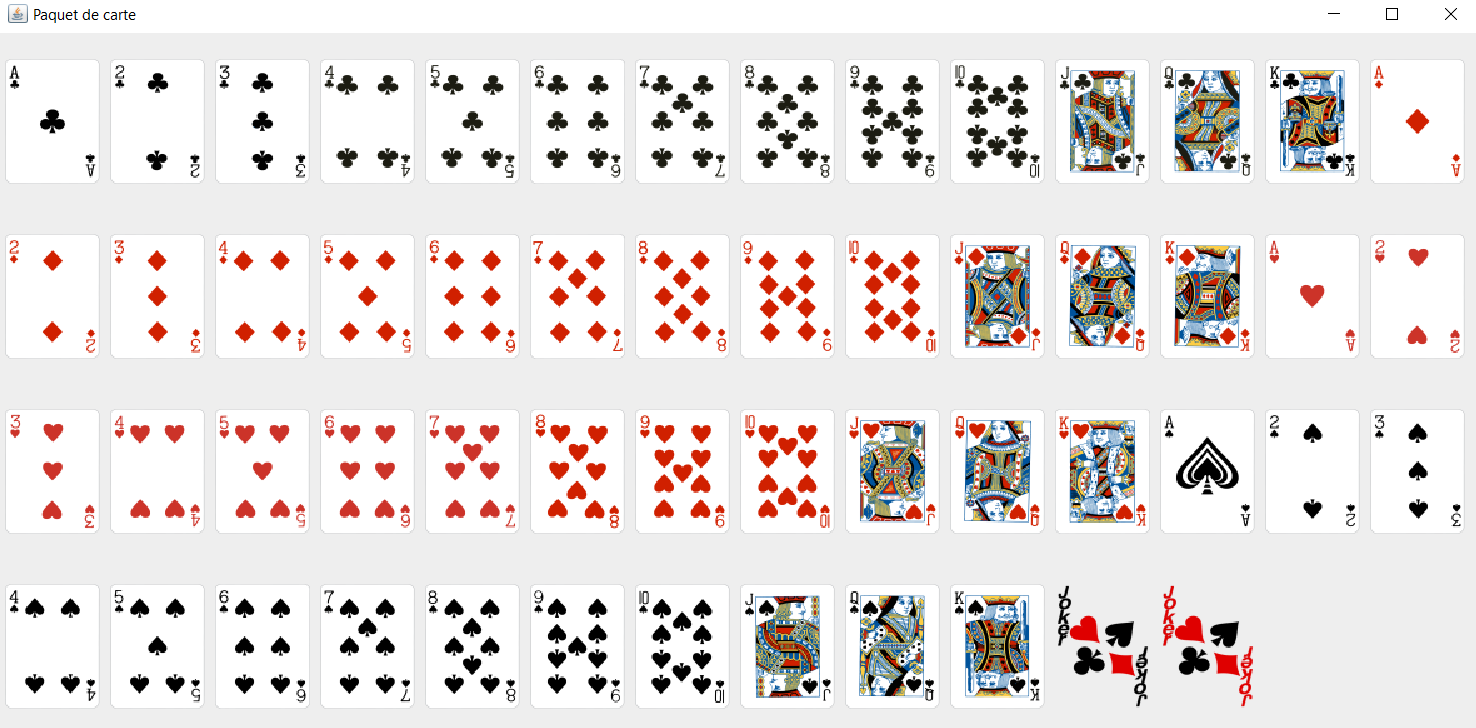
Description générée automatiquementPour structurer votre fichier de cartes, vous devez utiliser le format suivant : « VALEUR » de « COULEUR » pour toutes les cartes à l'exception des jokers.

Les jokers doivent être désignés par "JOKER\_NOIR" et "JOKER\_ROUGE".

Si votre fichier ne contient pas exactement 54 cartes ou si une carte est mal orthographiée, une erreur sera générée et le programme ne pourra pas fonctionner correctement.

3 : Exemple de fichier stockant un paquet

Enfin la vue du paquet de carte ressemble à ceci :



: Vue d'un paquet de carte

### Tests

Pour assurer la fiabilité de notre projet, nous avons réalisé des tests unitaires pour chaque étape de l'algorithme. Ainsi, dès que nous codions une étape, nous écrivions également sa méthode de test. Nous avons également comparé nos résultats avec ceux du site https://www.dcode.fr/chiffre-solitaire-schneier pour nous assurer de leur exactitude. Cela nous a permis d'avancer en toute sécurité et d'avoir la certitude que nos résultats étaient corrects.

# Conclusion

## Bilan du projet

Nous sommes convaincus que ce projet est une réussite. En effet, nous avons réussi à programmer la méthode de chiffrement Solitaire, et même à aller au-delà de ce qui était initialement demandé en permettant à deux personnes de communiquer de manière cryptée. En outre, l'interface utilisateur est conviviale et permet une utilisation facile de l'application.

## Difficultés rencontrées

Lors de la réalisation de ce projet, nous n'avons pas rencontré de difficultés majeures. Bien que quelques erreurs aient été commises lors de l'implémentation de l'algorithme, nous avons pu les corriger en utilisant un paquet de cartes. C'est la raison pour laquelle nous avons décidé de programmer des fonctionnalités supplémentaires, telles que l'importation et l'exportation de son paquet de cartes pour communiquer avec d'autres personnes, ainsi qu'une interface visuelle conviviale.

## Perspectives d'amélioration

Bien que l'application ait été conçue pour répondre aux besoins spécifiques du projet, elle peut être améliorée de plusieurs manières. Par exemple, nous pourrions mettre en place un système de communication en temps réel avec des sockets réseau.

1. Énoncé du projet : <http://v.vincent.u-bourgogne.fr/0ENS/THEO-INFO/projet.pdf> [↑](#footnote-ref-1)