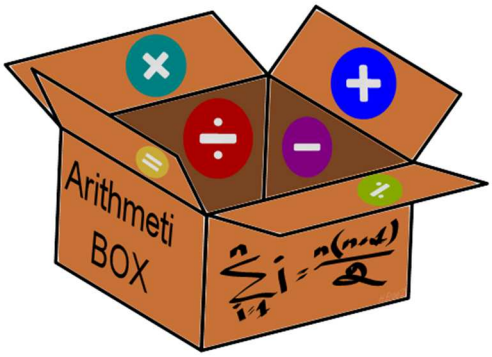

Rapport de projet
Arithmétibox

Quentin RAT
Adrien CAVALIERI
Jack KAING
Senaroith NOR
Fahath MOUGAMMADOU

Ce projet est encadré par David Hébert

Avant-propos et remerciement

Le projet assigné pour notre dernier semestre est « Arithmétibox ». Projet pensé, médité, imaginé et conçu par M. David Hébert. Toutefois, Pour conquérir le continent d'Arithmétibox des Impériaux, une première équipe a été formé. Quentin, Jason, Jack, Jeremy et Fahath étaient les élus pour rendre ce continent aux Nordiens. Mais une guerre civile éclata entre les Nordiens et les Impériaux et nos cinq héros se dirigèrent vers la capitale du continent et siège des Impériaux, Solitude, pour négocier une trêve.
Insulté par l'absence de Nordien lors de cette négociation, les Impériaux décidèrent d'attaquer nos cinq héros. Blessés, nos cinq héros réussirent à prendre la fuite et tuèrent, par la même occasion, le capitaine César et ses sous-capitaines Affine et RSA. Néanmoins, deux de nos héros, Jason et Jeremy reçurent une flèche dans le genou et les rendirent invalides pour continuer cette guerre.
Nos héros perdirent tout espoir pour conquérir à la suite de l'invalidité de Jason et Jeremy. C'est alors, venu de Sovngarde, deux anciens âmes, Adrien et Senaroith pour prêter main forte à nos héros. Ils repartirent, en laissant Jason et Jeremy en arrière, donc en combat mais ont cette fois pour objectif de tuer le Capitaine Hill dimension N, Substitution et Vigenere pour rendre le continent aux Nordiens.

Nous remercions M. David Hébert pour l'aide qu'il nous a apporté tout au long du projet, que ce soit en S3 ou en S4.

Nous remercions, également, M. Lucian Finta qui a confirmé la continuité du projet pour ce S4.

Table des matières

Introduction

Le projet Arithmétibox a pour finalité d'être sur le site de M. David Hébert. Ce site aura pour vocation d'aider les futurs étudiants de S3 de DUT Informatique qui étudieront le module de cryptanalyse.
Pour le projet, nous continuerons à utiliser les langages de programmation web tels que le PHP, le HTML, le CSS et le JavaScript. Ainsi que le langage de compilation mathématiques LaTeX. Ce projet sera encadré par notre enseignant-chercheur et client M. David Hébert. De ce fait, nous serons cinq à développer ce site.
Quentin sera le chef de projet et se chargera de rendre les documents à notre client, M. David Hébert.

1. Description du projet

Nous reprenons la base du site qui a été créé en S3. Toutefois, il n'y aura aucune modification sur les fichiers déjà créés. Sauf, si une de nos missions nous le demande. Dans ce cas, les modifications auront lieu à cet effet. Nous avons donc juste à implémenter de nouvelle fonctionnalité sans toucher à cette base.

Lors de notre premier rendez-vous avec M. Hébert. Nous avons eu trois directives pour réaliser le projet.

- **Le chiffrement par substitution :**

  Continuer le développement de ce dernier et ajouter à la fonction de substitution la possibilité de changer le résultat obtenue à l'aide d'un bouton changer. De plus, il faut afficher l'analyse fréquentielle dans un cadre à côté du message à décoder.

- **Le chiffrement de Hill :**

  Terminer la méthode de chiffrement de Hill et implémenter le cryptage et décryptage de Hill à dimension n.

- **Le chiffrement de Vigenere :**

  Implémenter une fonction de cryptage et décryptage du chiffrement de Vigenere.

2. Solutions mises en œuvre

2.1 Définition des exigences et des contraintes

Pour mener à bien ce projet, nous avons dû déterminer les exigences fonctionnelles et non fonctionnelles. Les exigences fonctionnelles décrivent les caractéristiques du système ou des processus que le système doit exécuter. Tandis que, les exigences non fonctionnelles décrivent les propriétés que le système doit avoir

- **Exigences fonctionnelles :**
  - Codé et décodé par la méthode de substitution
    Continuer l'implémentation de la méthode de substitution
  - Ajout d'un bouton « changer » dans substitution
    Le bouton « changer » nous permettra de permuter les lettres en fonction de leur fréquence d'apparition dans le message.
  - Crypter et décrypter par la méthode de Hill à dimension N
    Implémenter la fonction qui permettra de résoudre un déterminant d'une matrice de dimension N
  - Codé et décodé par la méthode de Vigenere
    Implémenter la fonction de codage et de décodage pour le codage de Vigenere

- **Exigences non fonctionnelles :**
  - Amélioration du design du site
    Rendre le site plus « user friendly »
  - Rendre la page dynamique
    Implémentation de code JavaScript dans le site web

Les exigences définies, il nous reste à caractériser les contraintes que les développeurs peuvent rencontrer.

- **Contraintes :**
  - Logiciel WAMP ou MAMP
    L'utilisation du logiciel WAMP ou MAMP est essentielle pour avancer dans le projet. Ces logiciels nous permettent d'exécuter du code PHP en créant un serveur local.
  - GitHub
    La plateforme GitHub est, également, crucial pour le projet. Cette plateforme nous permet de partager le code avec le reste de l'équipe.
  - WhatsApp
    WhatsApp est le groupe de messagerie qu'on utilise pour communiquer entre nous.

2.2 Les missions et la répartition des tâches

À partir des directives données par M. Hébert. Nous avons pu dégager des missions pour mieux organiser le déroulement du projet. Voici la liste des missions :

- **1.** Ajouter en JS un bouton qui va changer le résultat du décryptage par substitution
- **2.** Modifier le design de la page web
- **3.** Créer une fonction Chiffrement de Vigenere
- **4.** Ajouter dans un cadre à côté du message à coder l'analyse fréquentielle de chaque caractère
- **5.** Crypter et décrypter un message avec une matrice de dimension N

Nous pouvons maintenant estimer la difficulté de ces missions, ainsi que le temps nécessaire à les réaliser.

N° des stories	Facile	Moyen	Difficile
1	4h		
2	8h		
3		10h	
4		15h	
5			15h

Les tâches sont classées suivant leur niveau de difficulté (facile, moyen ou compliqué) cela permet d'estimer approximativement la durée de la tâche dans les cases. Le numéro des stories correspond au numéro des stories listé plus haut (dans la partie “c – Stories”).

Total= 52 points
Nous avons une équipe de 5 développeurs
1 développeur peut faire 15 points 5*15=75 points par mois
52/75= 0,7 mois

La répartition des tâches entre les membres de l’équipe de projet est :

- Fonction de codage de Vigenère *réalisé par* Adrien et Senaroith
- Fonction Hill dimension N *réalisé par* Fahath
- Fonction de substitution *réalisé par* Quentin et Jack

3. Problèmes rencontrés et leurs résolutions

3.1 Les problèmes rencontrés

Nous avons rencontré des problèmes lors de l'implémentation de la fonction de substitution. Nous avons pensé, au début, qu'il fallait garder en mémoire toutes les possibilités de l'alphabet, soit 26 ! alphabets à mettre dans un tableau. Nous nous sommes vite rendu compte qu'à partir de 8 !, nos machines ne pouvaient supporter cette masse de donnée. La génération de l'alphabet devait donc se faire seulement au moment de l'appui du bouton changer.

Nous avons, également, eu des problèmes pendant l'implémentation de la fonction de Vigenere. Effectivement, comme écrit plus tôt dans le rapport, c'est un codage qui n'a pas été vu en cours de cryptanalyse. Nous avons donc eu besoin de faire des recherches ou de demander à des redoublants qui ont peut-être fait Vigenere en cryptanalyse.

3.2 Les résolutions des problèmes

Pour résoudre le problème d'alphabet de substitution, nous avons dû faire de longue et fastidieuse recherche sur un possible algorithme qui génère un alphabet lorsque l'utilisateur clique sur le bouton changer. Après plusieurs heures de recherche et plusieurs implémentation, nous avons, finalement, réussit à générer un alphabet à chaque clique du bouton changer.

Quant à la fonction de Vigenere, nous avons eu de la chance d'avoir un redoublant qui a fait de la cryptanalyse l'année d'avant et plus important il avait vu la fonction de codage de Vigenere. De plus, il s'était désigné à implémentation la fonction avec l'aide d'Adrien.

4. Explication des algorithmes

4.1 Codage par substitution

Dans ce crypto système, l'algorithme est une substitution de caractères, la clé étant la liste de substitution de l'alphabet. En d'autres termes, un chiffrement simple par substitution est défini par une application A→ A0. Où, nous notons l'alphabet du message par A et l'alphabet de texte chiffré par A0.

Supposons que nous codions d'abord un message en éliminant tous les caractères non alphabétique (par exemple nombres, espaces, et ponctuation) et en changeant tous les caractères en majuscule. Alors la taille de la clé, qui borne la sécurité du système, est 26 lettres. Par conséquent le nombre de clés est 26! = 403291461126605635584000000 = 4.10^26 = 2^88 qui est un nombre énorme.

Donnons un exemple pour crypter un mot:

Alphabet du message	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Alphabet de chiffrement	A	Z	E	R	T	Y	U	I	O	P	Q	S	D	F	G	H	J	K	L	M	W	X	C	V	B	N

le message SUBSTITUTION devient donc LWZLMOMWMOGF car le S devient un L, le U un W...

Pour attaquer substitution nous allons d'abord décrypter suivant la fréquence d'apparition des caractères et ensuite nous permuterons les caractères en appuyant sur un bouton « changer », une permutation consiste à intervertir des valeurs entre elle.

4.2 Chiffrement de Vigenere

Pour crypter/décrypter/attaquer avec la méthode de chiffrement de Hill nous avons besoin d'une matrice (qui sera ici de dimension NxN). On récupère chaque élément de la matrice et on l'insère dans une case du tableau mais comme pour toute méthode de cryptage, nous avons besoin d'une clé et d'un modulo qu'on calculera en fonction de l'alphabet du message ($alphabet). Avec les éléments récupérés dans le tableau on peut calculer le déterminant de la matrice.
Une fois le calcul fait on récupère l'équivalent modulaire en fonction du modulo calculé précédemment ($mod), ces étapes finies on affiche ces résultats grâce au LateX à l'utilisateur pour lui permettre de comprendre les étapes et ainsi avoir une meilleure compréhension de ce qu'il se passe (affichage de la matrice et le détail du calcul euclidien).

Nous devons maintenant vérifier que la clé est valide pour cela il faut calculer le PGCD entre le déterminant et le modulo et avoir un PGCD qui vaut 1. Toutefois, si le PGCD est différent de 1 on affichera un message disant que la clé rentrée n'est pas valide. Avant d'afficher la clé valide si l'utilisateur voulait décrypter le message on devrait récupérer l'inverse modulaire de la matrice grâce à la fonction créer par mon collègue, tout cela possible si bien sûr le PGCD vaut 1, on pourra donc commencer le décryptage une fois la matrice récupérée et le message affiché. La clé, prête, on peut passer au processus de conversion du message en chiffre qu'on utilisera pour le produit de matrice.
En fonction du choix cryptage/décryptage, on utilisera la matrice qu'il faudra et par simple produit de la matrice et de l'alphabet codé on obtient le nouveau message chiffré, que l'on convertira en lettre pour pouvoir récupérer le message crypter/décrypter. Mais le plus dur, ici, est de pouvoir calculer le déterminant d'une matrice qui est de dimension N, pour cela nous avons demandé l'aide de notre professeur de cryptanalyse M. Hébert qui a pu nous expliquer les étapes du processus du calcul. Prenons par exemple cette matrice de taille n :

$$A = \begin{pmatrix} a_{1;1} & \cdots & a_{1;n} \\ \vdots & \ddots & \vdots \\ a_{n;1} & \cdots & a_{n;n} \end{pmatrix}$$

Pour calculer le déterminant de cette matrice on devra le calculer à l'aide de N déterminant de matrice de taille N-1 qu'on obtiendra à la matrice de départ une ligne et une colonne nous allons donc utiliser une fonction récursive qui nous facilitera la tâche, la nouvelle matrice d'A avec la première ligne et colonne est :

$$A_{i,j} = \begin{pmatrix} a_{1,1} & \cdots & a_{1,j-1} & a_{1,j+1} & \cdots & a_{1,n} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{i-1,1} & \cdots & a_{i-1,j-1} & a_{i-1,j+1} & \cdots & a_{i-1,n} \\ a_{i+1,1} & \cdots & a_{i+1,j-1} & a_{i+1,j+1} & \cdots & a_{i+1,n} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,j-1} & a_{n,j+1} & \cdots & a_{n,n} \end{pmatrix}$$

Puis avec la formule donnée on pourra alors le calculer :

$$\det(A) = \sum_{j=1}^{n} a_{i;j}(-1)^{i+j} \det(A_{i,j})$$

On a donc utilisé la formule de Laplace. $(-1)^{i+j} \det(A_{i,j})$ est appelé cofacteur du terme et $a_{i,j}$ est $\det(A_{i,j})$ appelé le mineur du terme $a_{i,j}$

Voici le code en PHP de la formule de Laplace :

```php
$d = $d + (pow(-1, $k) * $matNN[0][$k] * det($smat,$taillem-1));
// $k=i+j $smat= Matrice N-1
```

4.3 Codage de Vigenere

Depuis le début du projet, tous les codages et chiffrements donnés par M. Hébert ont été vus en cours. Toutefois, le codage de Vigenere nous a été proposé par M. Hébert comme étant une ouverture sur la cryptanalyse vue en S3.
Ce codage nous a été inconnu et nous avons donc fais des recherches pour comprendre le principe de Vigenere et user de notre imagination pour implémenter ce codage.
Le principe du codage de Vigenere se base sur le codage de César (on utilise un alphabet allant de a à z et on utilise une clé pour codé le message). Sauf qu'avec Vigenere, la clé n'est pas un alphabet décalé mais un mot. Autrement dit, on utilise un mot clé et on effectue des translations de lettres par rapport au mot clé.
Prenons un exemple :

- On veut coder le mot « toilette »
- avec le mot clé : « code »

On répète le mot clé tout le long du message, le texte codé est obtenu en additionnant les deux nombres.
Codage du mot « toilette » :

Mot clé	c	o	d	e	c	o	d	e
Position dans l'alphabet	2	14	3	4	2	14	3	4
Mot à coder	t	o	i	l	e	t	t	e
Position dans l'alphabet	19	14	8	11	4	19	19	4
Addition	21	2	11	15	6	7	22	8
Mot codé	v	c	l	p	g	h	w	i

Le message codé de « toilette » avec la clé « code » est « vclpghwi ».
Le décodage suit le même principe.

Décodage du mot « vclpghwi » :

Mot à décoder	v	c	l	p	g	h	w	i
Position dans l'alphabet	21	2	11	15	6	7	22	8
Mot clé	c	o	d	e	c	o	d	e
Position dans l'alphabet	2	14	3	4	2	14	3	4
Soustraction	19	14	8	11	4	19	19	4
Mot décodé	t	o	i	l	e	t	t	e

Le message décodé de « vclpghwi » avec la clé « code » est « toilette ».

À noter que Vigenere ne peut être attaqué par force brute ou par analyse fréquentielle. Cependant, pour casser ce code, on peut utiliser le test de Kasiski qui consiste à repérer des répétitions de lettres dans un message codé.

Algorithme : Algorithme du codage de Vigenere

Entrées : Message à coder (*msgAC*), clé (*cle*) et alphabet (*Alphabet*)
Sorties : Message codé (*msgC)*
fonction : *indexAlpha*, retourne la position d'une lettre dans l'alphabet

début

Pour chaque lettre de *msgAC* et de *cle* **faire**
motClair <- *indexAlpha* de *msgAC*
cleCode <- *indexAlpha* de *cle*
code <- *motClair* + *cleCode*
code modulo taille *Alphabet*
msgC <- *code*

retourner message codé (*msgC)*

fin

Algorithme : Algorithme du décodage de Vigenere

Entrées : Message à décoder (*msgDC*), clé (*cle*) et alphabet (*Alphabet*)
Sorties : Message clair (*msgC)*
fonction : *indexAlpha*, retourne la position d'une lettre dans l'alphabet

début

Pour chaque lettre de *msgAC* et de *cle* **faire**
motClair <- *indexAlpha* de *msgDC*
cleDeCode <- *indexAlpha* de *cle*
de*code* <- *motClair* - *cleCode*
de*code* modulo taille *Alphabet*
msgC <- *decode*

retourner message clair (*msgC)*

fin

Conclusion

Ce projet a été pour nous une expérience très enrichissante, aussi bien au niveau scolaire qu'au niveau professionnel. En effet, ce projet nous a permis, tout d'abord de mettre en application les différentes compétences et connaissances acquises durant notre formation. Mais, il nous a également permis d'en apprendre plus sur les différentes démarches à prendre avant d'entreprendre un projet et les erreurs à éviter. Nous avons appris à travailler en autonomie, mais aussi en équipe avec des personnes qu'on ne connaissait pas forcément auparavant, nous avons également appris à respecter des délais . Enfin, ce projet tuteuré, nous a apporté un esprit de groupe et d'adaptation, qui nous sera utile lors de projets futurs dans notre vie professionnelle.