Cahier des charges

Outil automatique de décryptage

Ce projet a été réalisé par :
Abdelaziz Ameurlain, Redha Dali, Mamadou Mouctar Barry, Allaye Afo Diallo, Rachid El Badaoui, Sofiane Hamad, Raphael Marouani, Linda Bedjaoui

Chef du projet fonctionnel :
Linda Bedjaoui

Sous la direction de : Madame Leila Kloul

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

université PARIS-SACLAY

Table des matières

Lignes directrices du projet

1. Finalité du projet

1.a Contexte du projet

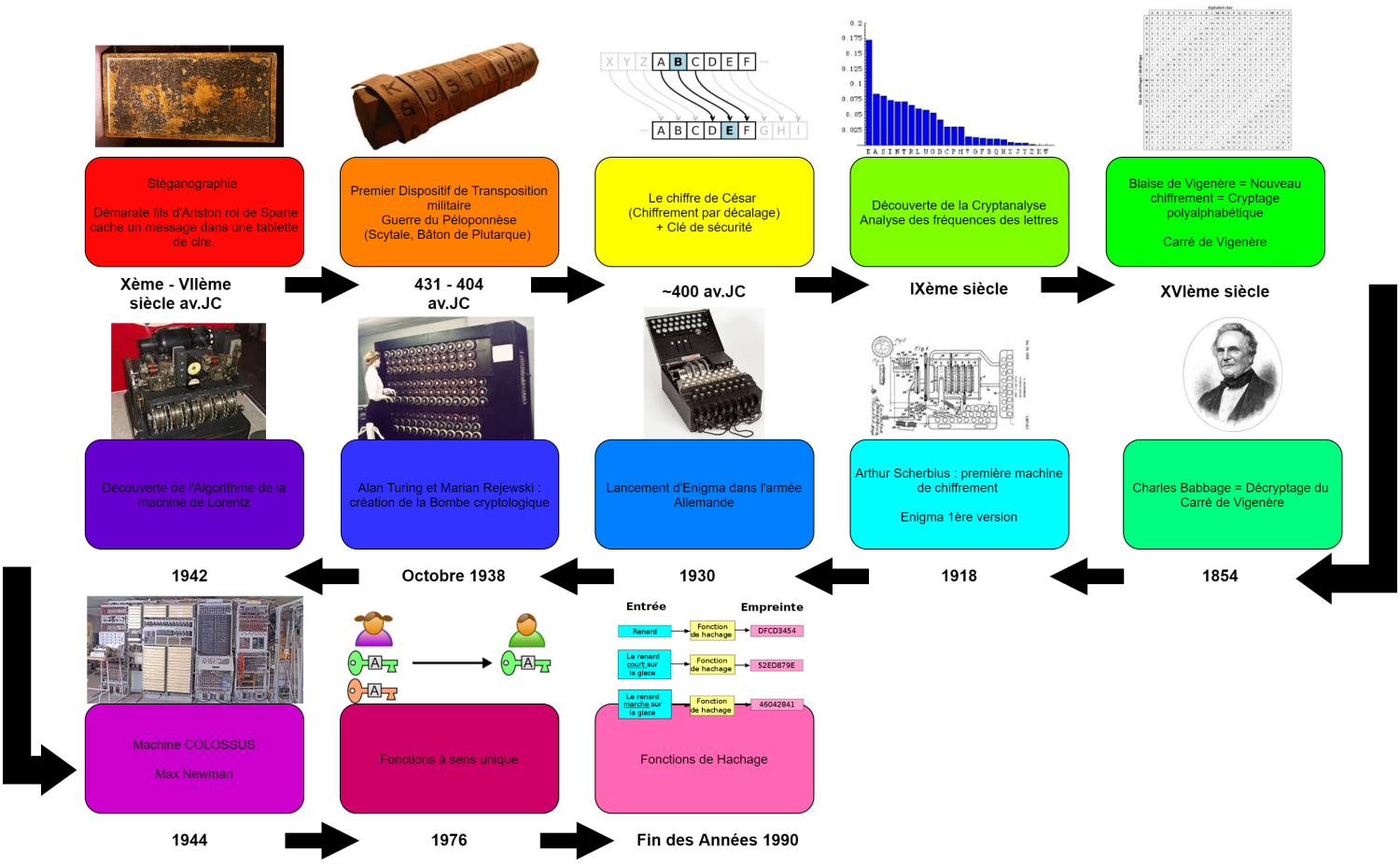
1.a.1 *Préambule*

Depuis toujours l'homme a ténu à garder certaines informations secrètes, dans ce but, il n'a cessé de chercher des solutions. La notion de cryptologie, aussi appelée «science du secret» a longtemps été considérée comme un ''art'', ce n'est qu'à partir du XIXe siècle que cet art de la dissimulation est devenue une discipline scientifique, regroupant ainsi la cryptographie et la cryptanalyse.

La cryptographie définit et étudie les systèmes de chiffrement utilisés pour dissimuler les messages.

La cryptanalyse quant à elle cherche à casser et déduire un message en clair d'un messgae crypté.

1.a.2 *Historique*



La Cryptographie est connue depuis des centaines d'années, mais reste encore étrangère au grand public. Avant de parler de Cryptographie, elle est intimement liée avec la Stéganographie, qui représente l'art de dissimuler un message.
Tout commence entre le Xème et le VIIème siècle av.JC, avec Démarate fils d'Ariston roi de Sparte, vivant chez l'ennemi Xerxès roi de Perse. Il décide de prévenir son peuple en dissimulant un message dans une tablette de cire.
Il grave le bois de la tablette et y fait couler de la cire par-dessus afin d'éviter que l'on remarque sa trahison.
Le tout premier dispositif de transposition militaire connu apparaît durant la guerre de Péloponnèse (431-404 av.JC) , avec la Scytale ou Bâton de Plutarque.
Dans les années qui suivent, le chiffrement par substitution voit alors le jour avec le chiffre de César (chiffrement par décalage). L'ajout d'une clé va venir alors renforcer la sécurité des messages. Cette méthode aura beaucoup de succès et sera utilisée durant tout le premier millénaire.
A partir du IXème siècle les Arabes de l'Islam des Lumières découvrent le domaine de la cryptanalyse, l'analyse des fréquences des lettres leur permettra de surpasser la sécurité du chiffrement par décalage.
Au XVIème siècle, Blaise de Vigenère, diplomate français propose un nouveau chiffrement soi-disant incassable. Le cryptage polyalphabétique grâce au carré de Vigenère, efface les possibilités d'analyse de fréquence des lettres.
On attendra alors près de 300 ans, car en 1854 Charles Babbage, précurseur anglais de l'informatique arrivera à décrypter un message par le repérage de motifs identiques.
En 1918, après la Première Guerre Mondiale et les avancées technologiques, les communications par télégraphe et par radio deviennent banales. Arthur Scherbius, un ingénieur allemand invente alors une machine de chiffrement (Enigma sous sa première version) qu'il n'arrivera pas à vendre à la marine allemande.
En 1930, après 12 ans de travail et de modifications, la marine est convaincue et commande 30000 Enigma, qui est une machine électromagnétique qui procède à un chiffrement par triple substitution renforcé par une triple transposition.
Octore 1938 : Alan Turing va alors se lancer dans l'amélioration de la bombe de Marian Rejewski, un mathématicien polonais, qui pourra faire un nombre de calcul immense et de combinaison de l'ordre du milliard par heure afin de décrypter les messages allemands utilisant Enigma.
Hitler et ses généraux utilisaient la machine de Lorenz, encore plus sécurisée car elle chiffrait chaque caractère en code binaire qui était lui-même rechiffré aléatoirement, c'est pour la première fois dans l'histoire de la cryptologie qu'un chiffrement pseudo-aléatoire est utilisé.
En 1942 un officier anglais trouve l'algorithme de la machine de Lorenz, car un opérateur allemand avait envoyé deux messages sans changer de clé.
Max Newman mathématicien et informaticien britannique avec l'aide de Tommy Flowers terminent les plans du premier calculateur électronique au monde : "La Machine Colossus".
Inspirée de la bombe de Turing, elle peut alors décrypter tous les messages de la machine de Lorenz, grâce à sa force d'opération de 5000/s.
C'est le début de l'industrialisation du cryptage.
En 1976, la découverte du secret des fonctions à sens unique secoue le domaine de la cryptographie.
Dès la fin des années 1990, certaines fonctions de hachage populaires ont été analysées puis « cassées ».
On peut citer les fonctions suivantes :
-IDEA, RSA, 1977
-MD5 (message digest 5) inventée en 1991 par Ronald Rivest -SHA-1 (Secure Hash Algorithm) et SHA-0 en 1993 par le NSA
Suite à ces recherches fructueuses, on peut se poser des questions sur la sécurité offerte par les algorithmes de hachage fréquemment utilisés dans les applications. Il est désormais clair que MD5 et SHA-1 ne doivent pas être incorporés dans les nouveaux logiciels.
La Cryptographie est un domaine très vaste qui s'est vu enrichir depuis des millénaires, l'industrialisation des machines de cryptage a été le tournant dans la démocratisation des techniques de chiffrement.
L'utilisation d'algorithmes de plus en plus puissant qui augmentent de façon considérable le nombre de possibilités de combinaisons, rendent de plus en plus difficile la résolution des secrets d'encodage.

1.a.3 *Les différents outils automatiques de décryptages*

Aujourd'hui il existe plusieurs systèmes de chiffrement, les plus connus étant entre autres:

Chiffrement par décalage	Chiffrement RSA
Chiffrement par permutation	Chiffrement par substitution
Chiffrement en chaine	Chiffrement de Vigenère
Chiffrement affine	

Dans notre projet nous nous intéresseront aux deux derniers. Ce sont des systèmes de chiffrement symétriques. Contrairement au RSA par exemple qui lui est asymétrique, c'est à dire qu'on utilise une paire de clé, publique pour chiffrer et privée pour déchiffrer. Ici on utilisera donc une même clé pour chiffrer et déchiffrer.

1.a.4 *Explication des outils : Chiffrement par Substitution et Chiffrement de Vigenère*

- **Chiffrement par Substitution**

ce chiffrement consiste à effectuer des permutations aléatoires sur l'ensemble des lettres alphabétiques, le chiffrement par décalage en est un cas particulier. La clé dans le chiffrement par substitution est tout simplement la permutation des 26 lettres comme chaque lettre peut avoir 26 permutations possibles, le nombre de clés possibles est donc 26!. Ce qui empêche la recherche exhaustive même pour un ordinateur.
Par exemple :

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
x	n	y	a	h	p	o	g	z	q	w	b	t	s	f	l	r	c	v	m	u	e	k	j	d	i

Le mot à chiffrer est : **BONJOUR** Le mot crypté est : **nfsqfuc**

- **Déchiffrement par Substitution**

Pour déchiffrer il suffit de récupérer la fonction de déchiffrement qui s'obtient en inversant la deuxième ligne de la table avec la premiere (dans l'exemple ci-dessus x->A n->B ....)

Le mot crypté est : **yxex** Le mot clair est : **CAVA**

Cependant il existe une autre manière pour cryptanalyser ce chiffrement qui s'appelle l'analyse des fréquences des lettres de l'alphabet.

- **Analyse des fréquences**

Elle est basée sur le fait que chaque langue contient des lettres, digrammes ou trigrammes qui apparaissent dans un message assez long avec une certaine fréquence. Par exemple en français la lettre E est la plus fréquente avec un pourcentage de 15,87.

- **Décryptage par Substitution**

Pour décrypter un message chiffré par substitution, on effectue l'analyse fréquentielle des lettres du message chiffré ainsi la lettre la plus fréquente si le texte est en français peut être la lettre E et on raisonne de la même manière avec le reste du message.

Par exemple si on prend le message chiffré :
Segelazewaopqjlnkfapzajyuyhklazeacnwpqepaaynepaykklanwperaiajp

Lettres	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	Total
Occurrences	12	12	12	0	7	1	1	1	1	3	4	4	0	4	1	7	2	1	1	0	1	0	3	0	4	3	62
Fréquences	**19,4**	**0**	**1,6**	**0**	**11,3**	**1,6**	**1,6**	**1,6**	**1,6**	**4,8**	**6,5**	**6,5**	**0**	**6,5**	**1,6**	**11,3**	**3,2**	**1,6**	**1,6**	**0**	**1,6**	**0**	**4,8**	**0**	**6,5**	**4,8**	**100**

On remarque que la lettre la plus fréquente est la lettre A on peut dire alors que A représente la lettre E en message clair, de même les lettres E/P sont fréquentes dans le message elles peuvent représenter les lettres I ou A en message clair, en suivant ce raisonnement on obtient notre message clair qui est : Wikipédia est un projet d'encyclopédie gratuite écrite coopérativement.

- **Chiffrement de Vigenère**

Le chiffrement de Vigenère est un système de substitution poly-alphabétique qui ressemble beaucoup au chiffrement de Cesar, sauf qu'il utilise une clé sous forme de mot ou de phrase, ce qui permet a une lettre d'être chiffrée de plusieurs façons d'ou le fait qu'il soit poly-alphabétique et rend la cryptanalyse plus difficile.
Pour chiffrer en Vigenère on procède comme cela :
- Chaque lettre de l'alphabet possède son indice (A = 0 , B = 1, ... , Z = 25)
- On converti le texte clair en utilisant les indices ci dessus
- On fait de meme pour le mot clé qui va se répéter pour être de la meme taille du texte clair.
- On additionne les indices du texte clair avec celle du mot clé modulo 26, et reconvertit les valeurs obtenues en lettres pour avoir notre mot chiffré.

Par exemple :

Le mot clé est : **CIPHER** (2,8,15,7,4,17)
Le mot clair est : **CRYPTOSYSTEM** (2,17,24,15,19,14,18,24,18,19,4,12)

2. 17 24 15 19 14 18 24 18 19 4 12
2 8 15 7 4 17 2 8 15 7 4 17
4 25 13 22 23 5 20 6 7 0 8 3
Le mot chiffré est : **EZNWXFUGHAID (E = 4 , ….)**

- **Déchiffrement de Vigenère**

Pour déchiffrer on utilise le meme mot clé mais en soustraction modulo 26.

- **Décryptage de Vigenère**

On remarque que dans le chiffrement de Vigenère, le nombre de mots clés possible est de 26 puissance m (taille de la clé). Cela rend la recherche exhaustive de clé très longue. On propose donc une autre méthode de cryptanalyse qui se déroule en 2 étapes :
La première consiste a retrouver la longueur de la clé en employant le test de Kasiski (blocs de repetition) et l'utilisation de l'indice de coïncidence ainsi que l'analyse et comparaison des fréquences pour rechercher les lettres qui composent notre clé. Après avoir récupérer la clé il ne manque plus qu'a faire le déchiffrement de Vigenère pour trouver le message.

1.a.5 *L'importance de la cryptographie*

Les domaines d'utilisations de la cryptographie sont vastes et vont du domaine militaire, au commercial, en passant par la protection de la vie privée.
Confidentialité, protection de l'information :
L'objectif fondamental de la cryptographie est de permettre à 2 personnes de communiquer au travers d'un canal de telle sorte qu'un opposant ne puisse pas comprendre ce qui est échangé. Le canal peut être par exemple une ligne téléphonique ou tout autre réseau de communication. On sait que de nos jours la cryptographie est indispensable, car le monde entier communique à travers les réseaux, on a ainsi besoin d'une certaine sécurité et une garantie que les messages envoyés soient totalement confidentiels. La cryptographie sert également a protéger des données tels que les mot de passe les comptes bancaires ou les codes nucléaires. La cryptographie a donc une grande importance dans le monde actuel.

1.b Objectifs du projet

Aider à la réalisation d'une application de chiffrement/déchiffrement/décryptage par substitution et Vigenère

Enjeu

L'application devra permettre de chiffrer un texte en utilisant le système de Vigenère et/ou le système de chiffrement par substitution, elle pourra aussi permettre de faire la cryptanalyse (décryptage) d'un texte chiffré, c'est à dire trouver le message caché sans avoir connaissance de la clé de chiffrement. L'application sera articulée autour d'une interface graphique qui permettra à l'utilisateur de naviguer entre les différentes fonctionnalités et d'en sélectionner une.

Mesure de la performance

Satisfaction du client qui s'agit ici de notre enseignante Madame Kloul.

Apprendre le travail d'équipe

Enjeu

Ce projet est constitué d'une équipe de 8 personnes, leurs objectif est donc de travailler harmonieusement ensemble et connaitre parfaitement leurs taches en suivant un planning semainier pour que l'application soit efficace et puisse satisfaire la cliente.

Mesure de la performance

Évolutivité de l'application selon les taches accomplies.

Contraintes

1. Contraintes liées a l'environnement

1.a Contraintes sur la solution

1.a.1 *Contraintes techniques*

L'application doit pouvoir offrir la possibilité de saisie du texte sur écran tactile pour les matériels mobiles suivant :

- Ordinateurs / PC portable

Le système doit être capable de chiffrer/déchiffrer/décrypter en utilisant l'outil de chiffrement par substitution et/ou Vigenère.

1.b Organisation du service informatique

1.b.1 *Contraintes de calendrier*

Il s'agit d'un projet dans le cadre d'une licence. Les engagements de l'équipe sont les suivants :

- Le projet est divisé en 5 Phases , chaque phase doit être livrée a une date précise :
  - Remise du Cahier de charge : 16/03/2021
  - Presentation orales : 23/03/2021
  - Remise des spécifications : 20/04/2021
  - Remise du compte rendu : 25/04/2021
  - Soutenance et démonstration : 01/06/2021
- L'ensemble de l'application doit être remis le 01/06/2021

1.b.2 *Conventions de dénomination et définitions des modules*

Ce projet est divisé en 4 modules :

- **Chiffrement :** Ce module regroupe 2 sous-modules :
  - **Chiffrement par substitution :**

Générer une clé qui va permettre de chiffrer le message en substituant chaque lettre alphabétique par une autre lettre alphabétique.

  - **Chiffrement de Vigenère :**

Chiffrer le message avec le mot clé fourni par l'utilisateur en remplaçant chaque lettre par son indice puis additionner ces indices avec ceux du texte en entrée en modulo 26.

- **Déchiffrement :** Ce module regroupe 2 sous-modules :
  - **Déchiffrement par substitution :**

Déchiffrer le message chiffré avec la clé que l'on connait.

  - **Déchiffrement de Vigenère :**

Déchiffrer le message chiffré avec la clé que l'on connait.

- **Décryptage :** Ce module regroupe 3 sous-modules :
  - **Décryptage de substitution :**

Décrypter le message chiffré en essayant d'abord de trouver la clé de chiffrement avec une attaque statistique par analyse de fréquence, et après avoir trouvé la clé on pourra déchiffrer le message.

  - **Décryptage de Vigenère :**

Décrypter le message chiffré en utilisant l'indice de coïncidence et le test de Kasiski qui consiste à chercher la longueur du mot-clé de chiffrement, puis faire une attaque statistique par analyse de fréquence, et après avoir trouvé le mot-clé on pourra déchiffrer le message.

  - **Analyse des fréquences :**

Calcul de la fréquence d'apparition des lettres de l'alphabet dans le texte d'entrée.

• Interface Graphique :

L'interface offre la possibilité d'écrire du texte en clair ou chiffré, puis de saisir la clé dans le cas d'un chiffrement ou d'un déchiffrement.
L'interface sera dotée des boutons :

1. Chiffrement par substitution
2. Chiffrement de Vigenère
3. Déchiffrement par substitution
4. Déchiffrement de Vigenère
5. Décryptage substitution
6. Décryptage Vigenère
7. Analyse fréquentielle
8. Menu Principal
9. Coffre fort
10. Aide

1.b.3 Liste des fonctionnalités et explication des algorithmes

• Analyse des fréquences :

- Parcourir le texte en entrée.
- Faire le calcul d'occurrences de chaque lettre .
- Faire le calcul de fréquence des lettres en pourcentage et le stocker dans une collection.

Pseudo-algo :

```
//on considère que le message est écrit en majuscule sans accent et espaces
Frequence(text):tab          // tab[0] corespond à la case de A ... tab[25] à celle de Z
Entrée   : message à l''entrée
Sortie   : tableau de fréquences
variable : tab: tableau de 26 cases
DEBUT_ALGORITHME
tab <-- {0};                 // on initialise les cases à 0
Pour i de 0 à taille(text)
DEBUT_POUR
        tab[text[i]-'A'] <-- tab[text[i]-'A'] + 1/taille(text);  // text[i]-'A' correspond à l'indice du tableau tab
Fin_pour
retourner tab;
FIN_ALGORITHME
```

Complexité O(n); // la taille du texte

• Chiffrement par substitution :

- Lecture de la clé qui devra être composée des 26 lettres différentes de l'alphabet, saisie par l'utilisateur sur l'interface graphique.
- Faire un tableau de deux dimensions. La première colonne contiendra les lettres dans l'ordre alphabétique. La deuxième colonne contiendra la clé (lettres de l'alphabet permutées par l'utilisateur).
- Lecture du fichier contenant le texte en clair écrit en entrée en remplaçant chaque lettre par sa correspondance dans le tableau selon la deuxième colonne. Ecriture du texte chiffré dans un autre fichier.

Pseudo-algo :

//on considère que le message est écrit en majuscule sans accent et espaces

Chiffrement_Substitution(texte, clef):texte
Entrée : texte: le message à chiffré, clef: la clef de chiffrement
Sortie : texte: le message chiffré
variable : tab: tableau de caractere
DEBUT_ALGORITHME
Pour i de 0 à 25
DEBUT_POUR
tab[i] <-- 'A' + i; // on initialise le tableau de A à Z
FIN_POUR
Pour i de 0 à taille(texte)
DEBUT_POUR
Pour j de 0 à 25
DEBUT_POUR
Si tab[j] = texte[i]
DEBUT_SI
texte[i] <-- clef[j]; // on change texte_clair en texte_chiffré
FIN_SI
FIN_POUR
FIN_POUR
retourner texte; //renvoie le message chiffré
FIN_ALGORITHME

Complexité O(n); // la taille du message

• Déchiffrement par substitution :

- Lecture de la clé qui devra être composée des 26 lettres différentes de l'alphabet, saisie par l'utilisateur sur l'interface graphique.
- Faire un tableau de deux dimensions. La première colonne contiendra les lettres dans l'ordre alphabétique. La deuxième colonne contiendra la clé (lettres de l'alphabet permutées par l'utilisateur).
- Lecture du fichier contenant le texte chiffré écrit en entrée en remplaçant chaque lettre par sa correspondance dans le tableau selon la première colonne. Ecriture du texte en clair dans un autre fichier.

Pseudo-algo :

```
//on considère que le message est écrit en majuscule sans accent et espaces
Dechiffrement_Substitution(texte, clef):texte
Entrée   : texte: le message à dechiffré, clef: la clef de chiffrement
Sortie   : texte: le message déchiffré
variable : tab: tableau de caractere (Alphabet)

DEBUT_ALGORITHME
Pour i de 0 à 25
DEBUT_POUR
        tab[i] <-- 'A' + i;   // on initialise le tableau de A à Z
FIN_POUR
Pour i de 0 à taille(texte)
DEBUT_POUR
  Pour j de 0 à 25
  DEBUT_POUR
   Si clef[j] = texte[i]
    DEBUT_SI
      texte[i] <-- tab[j];  // on change le texte chiffré en texte clair
    FIN_SI
    FIN_POUR
FIN_POUR
retourner texte; // retourne message en clair

FIN_ALGORITHME
```

Complexité O(n). // n est la taille du message

• Décryptage par substitution :

- Lecture du texte chiffré saisi par l'utilisateur sur l'interface graphique.
- Faire l'analyse des fréquences des lettres. Les comparer avec les fréquences des lettres de la langue choisie contenu dans le fichier de références.
- Lecture du fichier contenant le texte chiffré en remplaçant chaque lettre par sa correspondance. Ecriture du texte en clair dans un autre fichier.

Pseudo-algo :

Decryptage_Substitution(texte):texte
Entrée : texte: le message chiffre
Sortie : texte: le message clair
variable : tab_freq_francais[26]: tableau de frequence des lettres francaise, tab_freq_txt[26]: tableau de frequence du message chiffre
DEBUT_ALGORITHME
tab_freq_francais[26] <- 'E','A','S','I','N','T','R','L','U','O','D','C','P','M','V','G','F','B','Q','H','X','J','Y','Z','K','W';

Pour i de 0 à 25
DEBUT_POUR
tab_freq_txt[i] <-- max(Frequence(texte)); // la fonction max retourne la lettre la plus fréquente à partir de l'indice du tableau et mets la valeur correspondante à -1 pour pas stocker a chaque fois la même lettre .
FIN_POUR
Pour i de 0 à taille(texte)
DEBUT_POUR
Pour j de 0 à 25
DEBUT_POUR
Si tab_freq_txt[j] = texte[i]
DEBUT_SI
texte[i] <-- tab_freq_francais[j]; /// on remplace chaque lettre par sa correspondante (si X est le plus fréquente dans le texte chiffre il sera remplace par E qui est la lettre la plus fréquente dans la langue française ainsi de suite)

FIN_SI
FIN_POUR
FIN_POUR
retourner texte; //retourne texte en clair
FIN_ALGORITHME

Complexité O(n); // n est la taille du message

• Chiffrement de Vigenère :

- Ecriture de la clé à répétitions jusqu'à atteindre la taille du texte en clair à chiffrer.
- Assigner chaque lettre avec son indice.
- Additionner modulo 26 les indices du texte en clair et de la clé caractère par caractère de gauche à droite.
- Ecriture du texte chiffré dans un fichier en remplaçant les indices calculés par les lettres correspondantes.

Pseudo-algo

Chiffrement_Vigenere(texte_clair):texte_chiffre
VARIABLES
message EST_DU_TYPE CHAINE
i EST_DU_TYPE NOMBRE
code_lettre EST_DU_TYPE NOMBRE
decalage EST_DU_TYPE NOMBRE
clef EST_DU_TYPE CHAINE
longueur_message EST_DU_TYPE NOMBRE
longueur_clef EST_DU_TYPE NOMBRE
lettre EST_DU_TYPE CHAINE

DEBUT_ALGORITHME
LIRE message
LIRE clef
longueur_message PREND_LA_VALEUR message.length
longueur_clef PREND_LA_VALEUR clef.length
POUR i ALLANT_DE 1 A longueur_message
DEBUT_POUR
code_lettre PREND_LA_VALEUR message.charCodeAt(i-1)-65
decalage PREND_LA_VALEUR clef.charCodeAt((i-1)%longueur_clef)-65
SI (code_lettre-decalage>=0) ALORS
DEBUT_SI
lettre PREND_LA_VALEUR String.fromCharCode(65+(code_lettre-decalage)%26)
FIN_SI
SINON
DEBUT_SINON
lettre PREND_LA_VALEUR String.fromCharCode(65+(code_lettre-decalage)%26+26)
FIN_SINON
FIN_POUR
FIN_ALGORITHME

Complexité : O(n)

• Déchiffrement de Vigenère :

- Ecriture de la clé à répétitions jusqu'à atteindre la taille du texte chiffré à déchiffrer.
- Assigner chaque lettre avec son indice.
- Soustraction modulo 26 des indices du texte chiffré et de la clé caractère par caractère de gauche à droite.
- Ecriture du texte déchiffré dans un fichier en remplaçant les indices calculés par les lettres correspondantes.

Pseudo-algo

```
dechiffrement_Vigenere(texte_clair):texte_chiffre
VARIABLES
  message EST_DU_TYPE CHAINE
  i EST_DU_TYPE NOMBRE
  code_lettre EST_DU_TYPE NOMBRE
  decalage EST_DU_TYPE NOMBRE
  clef EST_DU_TYPE CHAINE
  longueur_message EST_DU_TYPE NOMBRE
  longueur_clef EST_DU_TYPE NOMBRE
  lettre EST_DU_TYPE CHAINE
DEBUT_ALGORITHME
  LIRE message
  LIRE clef
  longueur_message PREND_LA_VALEUR
message.length
  longueur_clef PREND_LA_VALEUR clef.length
  POUR i ALLANT_DE 1 A longueur_message
    DEBUT_POUR
    SI (message.charCodeAt(i-1)<=90) ALORS
      DEBUT_SI
      code_lettre PREND_LA_VALEUR
message.charCodeAt(i-1)-65
      decalage PREND_LA_VALEUR
clef.charCodeAt((i-1)%longueur_clef)-65
      SI (code_lettre-decalage>=0) ALORS
        DEBUT_SI
        lettre PREND_LA_VALEUR
String.fromCharCode(65+(code_lettre-decalage)%26)
        FIN_SI
        SINON
          DEBUT_SINON
          lettre PREND_LA_VALEUR
String.fromCharCode(65+(code_lettre-decalage)
%26+26)
          FIN_SINON
      FIN_SI
      SINON
        DEBUT_SINON
        code_lettre PREND_LA_VALEUR
message.charCodeAt(i-1)-97
        decalage PREND_LA_VALEUR
clef.charCodeAt((i-1)%longueur_clef)-97
        SI (code_lettre-decalage>=0) ALORS
          DEBUT_SI
          lettre PREND_LA_VALEUR
String.fromCharCode(97+(code_lettre-decalage)%26)
          FIN_SI
          SINON
            DEBUT_SINON
            lettre PREND_LA_VALEUR
String.fromCharCode(97+(code_lettre-decalage)
%26+26)
          FIN_SINON
        FIN_SINON
    FIN_POUR
FIN_ALGORITHME
```

Complexité : O(n)

• Décryptage de Vigenère :

- **<u>Trouver la taille de la clé :</u>**

1. **Test de Kasiski :**
   - Rechercher dans le texte chiffré les séquences qui se répètent au moins une fois.
   - Calculer la distance séparant les séquences qui se répètent.
   - Rechercher le diviseur commun de tous les écarts calculés. Ce diviseur représente la taille de la clé.

Pseudo-algo

```
Kasiski(texte_chiffre):longuer_cle
VARIABLES
   taille_sequence EST_DU_TYPE NOMBRE
   debut EST_DU_TYPE LISTE
   i EST_DU_TYPE NOMBRE
   taille_message  EST_DU_TYPE NOMBRE
   message EST_DU_TYPE CHAINE
   trouve EST_DU_TYPE CHIANE
   chaine EST_DU_TYPE CHAINE
   occurence EST_DU_TYPE NOMBRE
   ecart EST_DU_TYPE NOMBRE

DEBUT_ALGORITHME
 taille_sequence PREND_LA_VALEUR 3
  trouve PREND_LA_VALEUR true   //bool trouve =
true
   taille_message PREND_LA_VALEUR
length(message)
  TANT_QUE (trouve == true) FAIRE
  DEBUT_TANT_QUE
   trouve PREND_LA_VALEUR false
      POUR debut ALLANT_DE 0 A taille_message -
taille_sequence+1
     DEBUT_POUR
       chaine[taille_sequence + 1]
       POUR i ALLANT_DE 0 A taille_sequence
      DEBUT_POUR
     chaine[i] PREND_LA_VALEUR message[debut + i]
       FIN_POUR
       chaine[i] PREND_LA_VALEUR '\0'
       occurrence PREND_LA_VALEUR
chercherOccurrence(chaine, taille_sequence, debut +
taille_sequence)
       SI (occurrence != -1) ALORS
       DEBUT_SI
         ecart PREND_LA_VALEUR  occurrence -
           iDebut
         trouve PREND_LA_VALEUR true;
         ajouterEcart(ecart)
       FIN_SI
     FIN_POUR
   FIN_TANT_QUE
  RETURN PGCDEcart()
FIN_ALGORITHME
```

chercherOccurence : Renvoie la distance entre une séquence et sa 2ème occurrence si elle existe, sinon -1
ajouterEcart : ajoute l'écart trouvé dans un tableau
PGCDEcart : calculer le diviseur commun entre les écarts.

Complexité : O(n^2)

2. Indice de coincidence :

- Calculer l'indice de coïncidence du texte chiffré à l'aide de la formule suivante : $I_c = \frac{n_A(n_A-1)}{n(n-1)} + \cdots + \frac{n_Z(n_Z-1)}{n(n-1)}$
n : nombre de caractère du message
nA…..nZ : nombre de A….Z dans le texte chiffré.
- Découper le texte chiffré en ne prenant qu'une lettre sur 2 puis sur 3…, et appliquer la formule précédente à chaque fois et ce sur chaque découpage.
- On compare les indices de coïncidences calculés pour chaque découpage et on sélectionne le plus grand. La taille du découpage correspondant est la taille de la clé.

Pseudo-algo

```
longueur_clef_IC(decode) // je mets le nom pour
pouvoir faire appel à la fonction
DEBUT_ALGORITHME
   LIRE decode
   tab[0] PREND_LA_VALEUR -1
   POUR i ALLANT_DE 1 A 7
   DEBUT_POUR
      tampon PREND_LA_VALEUR ""
      Ic PREND_LA_VALEUR 0
      POUR l ALLANT_DE 1 A tampon.length
         DEBUT_POUR
         a[l] PREND_LA_VALEUR 0
         FIN_POUR
      k PREND_LA_VALEUR  0
      TANT_QUE (k<= decode.length) FAIRE
         DEBUT_TANT_QUE
         tampon PREND_LA_VALEUR tampon +
decode.substr(k,1)
         k PREND_LA_VALEUR  k+i
         FIN_TANT_QUE

      POUR j ALLANT_DE 1 A tampon.length
         DEBUT_POUR
         a[j] PREND_LA_VALEUR
tampon.charCodeAt(j-1)
         FIN_POUR
      POUR j ALLANT_DE 65 A 90
       DEBUT_POUR
         N PREND_LA_VALEUR 0
         POUR m ALLANT_DE 1 A tempon.length
            DEBUT_POUR
            SI (a[m] == j) ALORS
               DEBUT_SI
               N PREND_LA_VALEUR  N+1
               FIN_SI
            FIN_POUR
         Ic PREND_LA_VALEUR  ic+N(N-1)/
(tempon.length(tampon.length-1))
       FIN_POUR
      tab[i] PREND_LA_VALEUR ic
   FIN_POUR
   indice PREND_LA_VALEUR -1
   max PREND_LA_VALEUR -1
   POUR i ALLANT_DE 0 A 7
   DEBUT_POUR
      SI max < tab[i] ALORS
         DEBUT_SI
         max PREND_LA_VALEUR tab[i]
         indice PREND_LA_VALEUR i
      FIN_SI
FIN_POUR
RETOUNER indice
FIN_ALGORITHME
```

Complexité O(n)

- **Trouver les lettres de la clé :**

  - Faire un découpage du texte chiffré en k lignes (k la longueur de la clé trouvée précédemment).
  - Appliquer l'analyse des fréquences sur chaque ligne.
  - Pour chaque ligne la lettre la plus fréquente correspondra à 'E' en alphabet français normal. A partir de cela il est possible de trouver le décalage alphabétique effectué sur chaque lettre car chaque découpage a été chiffré avec la même lettre de la clé.
  - Ainsi chaque lettre correspondant a la lettre 'A' en position 0, sera la clé de décalage pour chaque découpage. Ces lettres-clés mises bout à bout en suivant l'ordre du découpage, constitueront le mot clé.
  - Une fois le mot clé trouvé on applique le déchiffrement de Vigenère.

Pseudo-algo

```
rechercher_clef(text,lclef):clef
Entrée   : text chiffré, lclef: la longueur de la clef
Sortie   : la clef en clair
variable : tab: tableau de chaine de caractere, clef: chaine
de caractere, freq: tableau contenant la frequence des
lettres, max: nombre, indice: nombre
DEBUT_ALGORITHME
clef <-- "";
Pour i de 0 à lclef-1
DEBUT_POUR
   tab[i] <-- "";                    // on initialise les chaine
à null
FIN_POUR
Pour i de 0 à taille(text)
DEBUT_POUR
   tab[i%lclef] += text[i];        //O(n)     on separe le text
en bloc.
FIN_POUR
Pour i de 0 à lclef-1
DEBUT_POUR
   freq <-- frequence[tab[i]]      //O(m) ici n2 est la taille
tab[i]
   max <-- 0;
   indice <-- 0;
FIN_POUR
   Pour j de 0 à 25
DEBUT_POUR
      Si max < freq[j]
      DEBUT_SI
         max <-- freq[j]
         indice <-- j;
      FIN_SI
   FIN_POUR
   clef += (indice + 22)%26;       // indice est la position
de la lettre la plus frequent et 22 est le decalage pour
trouver la clef
retourner clef;
FIN_ALGORITHME
```

Complexité O(n+m) m < n => O(n)

Complexité O(n^2)

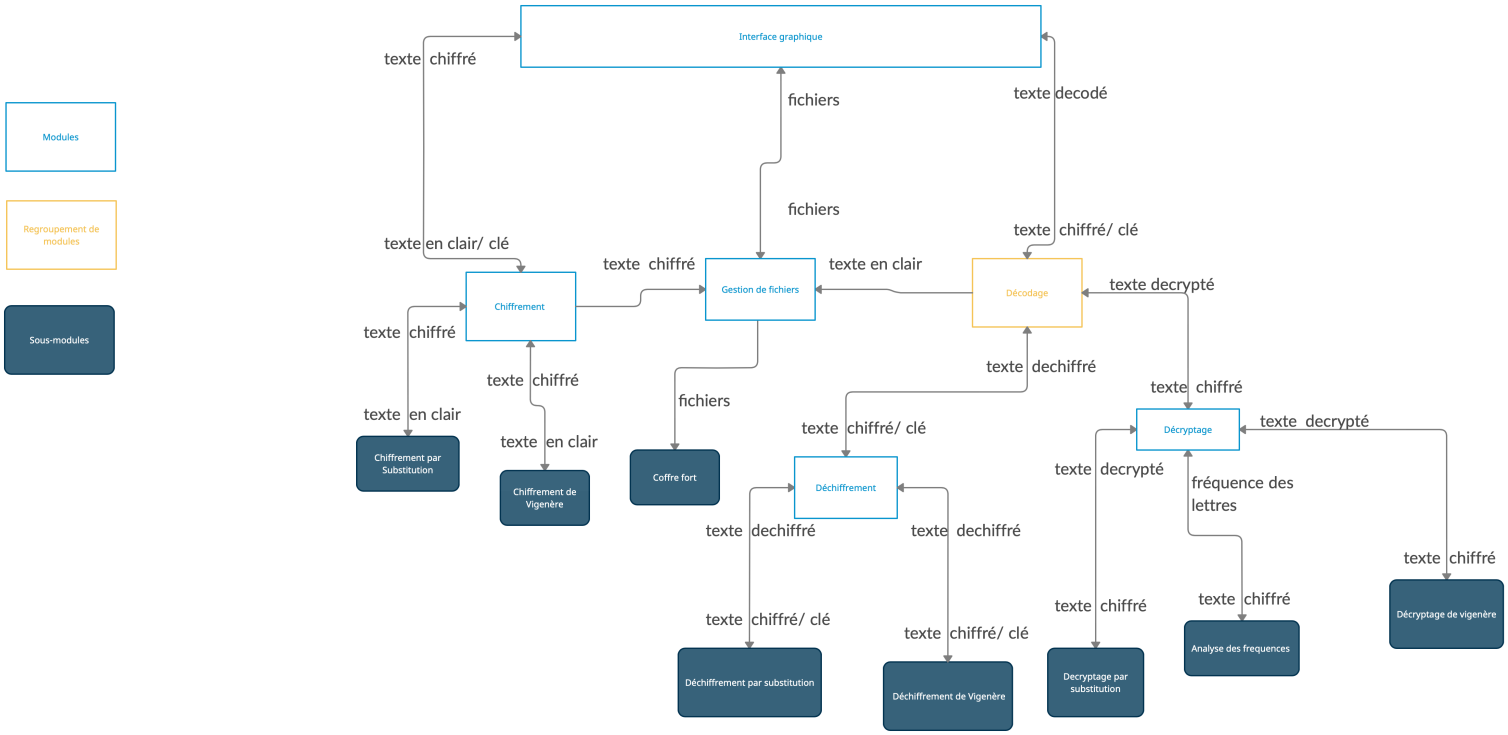
```
DEBUT_ALGORITHME
decryptage_vigenere(text):clair
lclef <-- longueur_clef_IC(text);          // O(n)
lclef <-- Kasiski(text);                   //O(n^2)
clef <-- rechercher_clef(text,lclef);      // O(n)
clair <-- dechiffrement_vigenere(clef);    //O(n)
RETOURNER clair;
FIN_ALGORITHME
```

Complexité O($n+n^2$+n+n) => O(n^2)

Exigences fonctionnelles

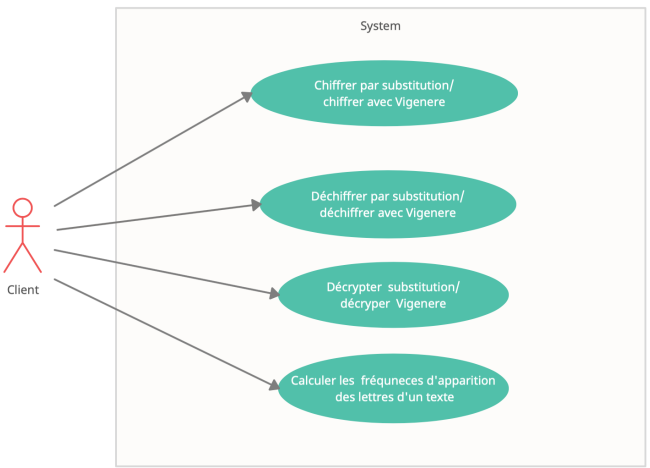
1. Portée du travail (contexte du projet)

L'organigramme ci-dessus a pour objet d'identifier l'ensemble d'informations qui circulent entre les différents modules et fonctionnalités de l'application. Le code couleur/forme est expliqué a gauche de l'image.



2. Portée du produit

2.a Limites du produit : diagramme de cas d'utilisation



2.b Description des cas d'utilisation

Le client aura la possibilité de chiffrer/déchiffrer/décrypter son texte soit par substitution ou avec Vigenère et notamment faire indépendamment l'analyse des fréquences de son message.

Exigences non fonctionnelles

1. Ergonomie et convivialité du produit

Les exigences de ce point concernent l'apparence de l'application (l'esprit de l'interface graphique).

1.a Les exigences de l'interface graphique:

La convivialité est une caractéristique importante pour rendre l'application attractive.

Pour cela l'application doit :

- Permettre à l'utilisateurs de naviguer intuitivement, et assurer une interaction fiable entre l'homme et la machine.
- Optimiser la lisibilité de l'information (couleurs, taille des textes adaptés…).
- Faciliter à l'utilisateur la saisie de texte, le choix de chiffrement, la sauvegarde de ses projets (en cliquant sur un bouton Enregistrer par exemple) et obtenir les résultats souhaités avec un minimum d'effort.

L'interface graphique donne une vision sur l'utilité, l'efficacité et la qualité de l'outil, elle représente la partie frontale de l'application qui regroupe ses recommandations dans le but de satisfaire les utilisateurs.

2. Facilité d'utilisation et facteurs humains

- L'application sera facile à utiliser pour des enfants de 13 ans.
- L'application guidera l'utilisateur afin de ne pas faire d'erreurs de saisie.
- L'application permettra à ses utilisateurs de tester les différentes options plusieurs fois.
- L'application sauvegardera l'avancement de ses utilisateurs dans un coffre fort pour faciliter l'accès aux anciens fichiers.

Autres aspects du projet

1. Estimation des coûts

Ce tableau résume l'estimation des lignes de codes des différents programmes ainsi que le temps nécessaire a consacrer pour chaque tâche en jours hommes.
Pour l'estimation du temps, nous nous sommes appuyés sur le temps entre la remise des spécifications et la soutenance (qui est de 1 mois) en utilisant la méthode PERT qui est une formule qui donne le temps moyen en fonction de l'estimation jugée plus probable pour la fin d'une tâche.
Estimation = (a + 4m + p) / 6
a : l'estimation la plus optimiste, m : l'estimation la plus probable, p : l'estimation pessimiste.
j/h : jours hommes.

Module :	Sous-module :	Responsables :	Temps jours hommes : (Toute les personnes)	Intervenants* :		Lignes de codes estimées :
CHIFFREMENT	Chiffrement par substitution	- Redha Dali - Mamadou Mouctar Barry	a=1/2, m= 2, p= 4 E = 2j/h a=1/2, m= 2, p= 4 E = 2j/h			60
	Chiffrement de Vigenère	- Sofiane Hamad - Rachid El Badaoui	a=1, m= 3, p= 5 E = 3j/h a=1, m= 3, p= 5 E = 3j/h			60
DÉCHIFFREMENT	Déchiffrement par substitution	- Abdelaziz Ameurlain - Redha Dali	a=1/2, m= 1, p= 3 E = 2j/h a=1/2, m= 1, p= 3 E = 2j/h			60
	Déchiffrement de Vigenère	- Linda Bedjaoui - Sofiane Hamad	a=1/2, m= 2, p= 4 E = 3j/h a=1/2, m= 2, p= 4 E = 3j/h			60
DÉCRYPTAGE	Analyse des fréquences	- Redha Dali - Abdelaziz Ameurlain	a=1/2, m= 1, p= 3 E = 2j/h a=1/2, m= 1, p= 3 E = 2j/h			50
	Décryptage par substitution	- Mamadou Mouctar Barry - Abdelaziz Ameurlain	a=4, m= 6, p= 8 E = 6j/h a=4, m= 6, p= 8 E = 6j/h			80
	Décryptage de Vigenère	- Rachid El Badaoui - Linda Bedjaoui	a=3, m= 5, p= 7 E = 7j/h a=6, m= 8, p= 10 E = 7j/h	- Sofiane Hamad - Mamadou Mouctar Barry	4j/h 2j/h	200
INTERFACE GRAPHIQUE		- Allaye Afo Diallo - Raphael Marouani	a=8, m= 10, p= 12 E = 10j/h a=8, m= 10, p= 12 E = 10j/h	- Redha Dali	4j/h	600

Intervenants* : Cette colonne désigne les personnes qui interviendront dans d'autres modules à la fin de leurs tâches principales. Devant chaque personne figure la charge de travail restante.

2. Tâches à faire pour livrer le système

Etape :	Description :
Définir les objectifs du projet	- Clarifier la demande du client et répondre à ses attentes.
Découper le projet et définir la liste des tâches	- Dresser la liste des tâches a réaliser en se basant sur les modules de l'application.
Définir l'enchaînement logique des tâches	- Etablir un tableau chronologique qui relie et ordonne les tâches FD(fin a début) - Définir le délai entre les tâches.
Définir et attribuer les ressources	- Le groupe contient 8 personnes. - Personnaliser les tâches aux groupe.
Planifier et assurer le suivi du projet	- Rédiger un planning par semaine qui doit avancer au rythme du projet.

3. Salle d'attente : idées pour les futurs versions

- Ajouter plus d'outils de décryptage et de langue.
- Communiquer et échanger des fichiers entre plusieurs utilisateurs (Alice envoie un message chiffré et une clé a Bob, Bob renvoie une réponse a Alice).

4. Explication du choix du langage

Les différentes caractéristiques de l'application :

- Assurer la bonne gestion des entrées/sorties (lecture/écriture sur des fichiers).
- Assurer la bonne gestion des données.
- Optimiser le temps d'exécution et la consommation maximale de la mémoire.
- Lecture et traitement de texte (longues chaines de caractères à manipuler).
- Vitesse de calcul.
- Developper une interface graphique très complète compatibles avec les principales plates-formes disponibles sur le marché.
- Possibilité d'exécuter l'application sous Windows/Linux/Mac.

Pour pallier à tous ces besoins, les fonctions et les bibliothèques disponibles en C et C++ permettant de manipuler facilement des fichiers textes présentent un grand avantage.
Cela dit dans l'optique de développer une interface graphique multiplateformes, QT est un outil interessant offrant beaucoup de possibilités facilitant le développement de ce type d'interface.
La framework étant particulièrement adapté au C++, cela constitue un argument de poids en faveur du C++ au détriment du C.
Pour ces raisons, notre choix se porte donc sur le langage C++ qui ce trouve être un langage hybride permettant de faire du procédurale ainsi que de l'orienté objet.

Conclusion

Ce projet a pour but de créer une application (ODD) qui répond aux attentes de notre client (Madame Leila Kloul).
Afin d'arriver à concevoir cette application qui constitue a chiffrer/déchiffrer/décrypter un texte en utilisant deux outils de décryptage (Chiffrement par substitution et Vigenère) et calculer la fréquence d'apparition d'une lettre dans un texte, nous avons suivi un modèle précis le modèle Volère pour la rédaction de ce cahier de charge. Il a fallu découper le projet en 4 modules repartis sur une équipe de 8 étudiants de dernière année licence informatique et rédiger un planning par semaine. Cette méthodologie ainsi que les conseils reçus lors des séances de TD nous a permis de mettre en oeuvre nos connaissances et de nous projeter dans une situation plus professionnelle.

Sources et références

- Douglas Stinson, *Cryptographie - Théorie et pratique*, 2ème édition. Traduit par Serge Vaudenay Avoine et Pascal Junod. Éditions Vuibert, 2003 (Chapitre 1 : Cryptographie classique).
- WIKIBOOKS, *Les fonctions de hachage cryptographique*, **[en ligne].** (Modifié le 16 avril 2020). Disponible sur : < https://fr.wikibooks.org/wiki/Les_fonctions_de_hachage_cryptographiques > (Consulté le 14 février 2021).
- WIKIPEDIA, *Fonction de hachage universelle à sens unique*, **[en ligne].** (Modifié le 16 avril 2020). Disponible sur : <https://fr.wikipedia.org/wiki/Fonction_de_hachage_universelle_%C3%A0_sens_unique> (Consulté le 16 février 2021).
- NDG, *comment vaincre le chiffre de Vigenère*, Algorithme de calcul d'indice de coïncidence. (Modifié par M.Philippe le 23 mars 2011. Disponible sur : <http://mphilippe.fr/cryptologie/seance4.pdf> (Consulté le 11 mars 2021).
- YOUTUBE, *L'art de la cryptographie Partie 1*, **[en ligne].** (Modifié le 30 juin 2019). Disponible sur : <https://www.youtube.com/embed/19obn2YSCA0> (Consulté le 15 février 2021)
- YOUTUBE, *L'art de la cryptographie Partie 2*, **[en ligne].** (Modifié le 14 juillet 2019). Disponible sur : <https://www.youtube.com/embed/MVs2NSWDbkU> (Consulté le15 février 2021)