esiea
ECOLE D'INGENIEURS
DU MONDE NUMERIQUE

TPC n°1 LAB1413

Enigma

Nicolas Bodin
Arnaud Bannier

La machine Enigma est une machine de chiffrement inventée par les allemands dès la fin de la première guerre mondiale. Réputée à l'époque comme incassable, Enigma résista pendant plus de 20 ans aux efforts des alliés. Nous vous proposons de découvrir dans ce TP maths / info son mode de fonctionnement.

1. Introduction

Un mécanisme de chiffrement permet de traduire un message compréhensible (*texte clair*) en un texte incompréhensible (*texte chiffré*). Par exemple, le chiffre de César consistant à remplacer une lettre par la troisième lettre la suivant dans l'alphabet (A devient D, B devient E etc.) peut être considéré comme un mécanisme de chiffrement. L'*alphabet* que nous utiliserons dans ce TP est notre alphabet traditionnel composé de 26 lettres.

FIGURE 1 – Photographie de la machine Enigma (source : `http://cryptomuseum.com`)

À la fin de la première guerre mondiale, les allemands développent une machine révolutionnaire permettant d'assurer la sécurité de leurs communications : Enigma (fig. 1).

Cette machine, simple d'utilisation, possède une caractéristique particulièrement intéressante pour les cryptographes : sa sécurité réside dans le nombre de clés possibles et dans le fait que la même lettre n'est pas toujours chiffrée de la même façon.

Lorsqu'un opérateur veut chiffrer un message, il saisit sur le clavier d'Enigma les lettres du clair une à une. En saisissant une touche, un courant électrique part de la lettre en clair, passe au travers de la machine et vient illuminer une lettre sur le tableau des chiffrés. Afin d'assurer la sécurité des communications dans le temps, l'opérateur doit regarder en début de journée les paramètres à utiliser pour la journée, paramètres que l'on peut traduire sous la forme de clé.

Le nombre de possibilités de ces paramètres est tellement grand qu'il est impossible pour un humain de tous les tester, même à plusieurs. Les alliés ont donc dû développer des algorithmes afin de venir à bout de ce chiffrement. Des premiers travaux réalisés par les français puis les polonais au début des années 30, jusqu'à la mise à mort d'Enigma par, entre autres, Alan Turing travaillant à Bletchley Park près de Londres, il s'est écoulé plus de 20 ans. Tous ces efforts et la création de machines nommées *bombes de Turing* – l'ancêtre de nos ordinateurs – auraient permis de raccourcir la guerre de 2 ans et d'éviter plusieurs millions de morts [1].

2. Quelques caractéristiques et observations

Comme indiqué sur la figure 2, un tableau de connexions sur le devant de la machine permet de permuter 12 lettres de l'alphabet 2 à 2. La disposition des 6 câbles est donc une partie de la clé. La machine de base dispose également de 3 rotors, permutant chacun les 26 lettres de l'alphabet. Leur position initiale (identifiable par un indice sur le coté du rotor, indice allant de A à Z) ainsi que leur ordre fait partie de la clé. Les connexions internes aux rotors sont toutes identiques pour l'ensemble des 100 000 machines Enigma. Au bout des trois rotors se trouve un réflecteur, permettant de renvoyer le courant en sens inverse, et ainsi le chiffrement se fait de la même façon que le déchiffrement. Notons que la présence de ce réflecteur empêche le chiffrement d'une lettre vers elle-même.

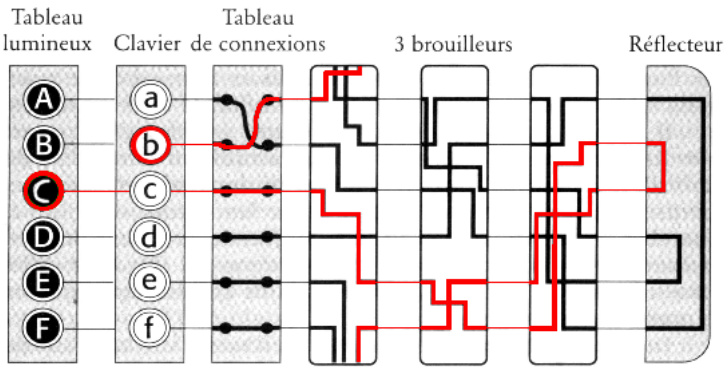

FIGURE 2 – Schéma du fonctionnement d'Enigma (source : `http://nomis80.org`)

Lors de la saisie d'une nouvelle lettre sur le clavier, la lettre chiffrée correspondante s'affiche sur la machine, puis le rotor 1 tourne d'un cran. Lorsque ce rotor passe de la

1. `https://fr.wikipedia.org/wiki/Enigma_(machine)`

position Z à la position A, le rotor 2 tourne lui aussi d'un cran. Le rotor 3 tourne de la même façon par rapport au rotor 2.

La figure 2 présentée plus haut montre un exemple de chiffrement de la lettre « `b` ». Le tableau de connexions permute cette lettre avec la lettre « `a` », le premier rotor la transforme en « `f` », le second en « `e` » et le troisième en « `c` ». Le réflecteur, suivi des rotors en sens inverse donne finalement le chiffré de « `b` » qui est « `C` ».

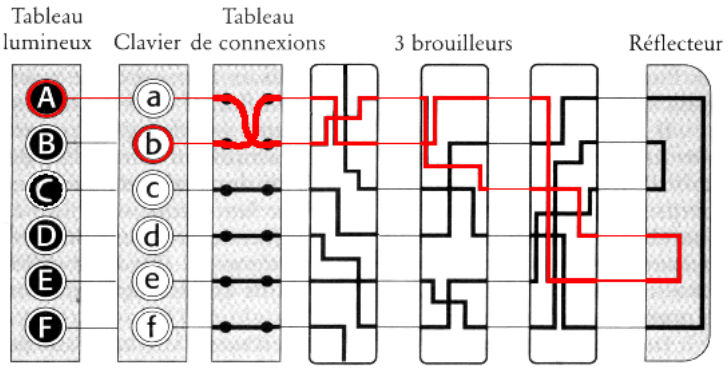

FIGURE 3 – Chiffrement de la lettre B après rotation d'un cran du premier rotor

Après la première lettre chiffrée, le rotor 1 tourne d'un cran ce qui change le chiffrement de la lettre « `b` » ainsi que celui des autres. Ainsi, comme le montre la figure 3, la lettre « `b` » est maintenant chiffrée en « `A` ».

3. Partie mathématiques

3.1. Dénombrement des clés

Si l'on dispose d'un message chiffré ainsi que d'une machine Enigma, il est (théoriquement) possible d'essayer toutes les clés (ou configurations initiales). Il y aura alors un unique texte clair intelligible si le chiffré n'est pas trop court. Cette attaque s'appelle la *force brute*. Bien que cette attaque soit toujours possible théoriquement, en pratique, elle n'est réalisable que si l'ensemble des clés n'est pas trop grand. Comme précisé en introduction, l'un des points forts d'Enigma réside dans son nombre de clés. A l'époque de sa création, il était impossible d'effectuer une force brute. Aujourd'hui encore, cette attaque reste très coûteuse. Comme mise en bouche, vous allez devoir calculer le coût d'une force brute.

Exercice 1. Dénombrement des clés possibles

Pour cet exercice, vous devez expliciter les formules que vous utilisez. Exceptionnellement, vous aurez le droit à la calculatrice pour calculer le résultat.

1.1) Donnez le nombre de possibilités pour l'ordre des rotors.

1.2) Donnez le nombre de possibilités pour la position initiale des rotors.

1.3) Donnez le nombre de façons de positionner les 6 câbles dans le tableau de connexions.

1.4) Déduisez-en le nombre total de clés possibles pour Enigma.

3.2. Tournez rotors !

Avant de commencer les choses sérieuses, un petit échauffement s'impose.

Exercice 2. Rappels

2.1) Rappelez la définition d'une application injective, surjective et bijective.

2.2) Soit $f : E \to F$ une application. Reliez les assertions équivalentes

f est injective •	• tout élément de F admet exactement un antécédent.
f est surjective •	• tout élément de F admet au moins un antécédent.
f est bijective •	• tout élément de F admet au plus un antécédent.

2.3) Expliquez pourquoi une fonction de chiffrement doit être injective.

Pour simplifier, nous supposons que notre alphabet n'est plus $\{\texttt{A}, \ldots, \texttt{Z}\}$ mais $[\![0, 25]\!]$. Ainsi, « `A` » est représenté par 0, « `B` » par 1, ... et « `Z` » par 25. Cette représentation est bien adaptée à l'arithmétique et à la programmation.

Définition 1 (Permutation). Soit E un ensemble. On appelle *permutation* de E toute bijection de E dans E.

Dans la suite, le symbole $\boxplus$ désigne une addition modulo 26 et le symbole $\boxminus$ une soustraction modulo 26. Par exemple, $22 \boxplus 5 = 1$ et $1 \boxminus 2 = 25$. Notons R_0 la permutation de $[\![0, 25]\!]$ associée à un rotor en position par défaut. On peut montrer que la permutation R_i associée au même rotor mais décalée de i positions est donnée pour tout x dans $[\![0, 25]\!]$ par

$$R_i(x) = R_0(x \boxminus i) \boxplus i \ .$$

Exercice 3. Réciproque d'un rotor

Soit $R_i : [\![0, 25]\!] \to [\![0, 25]\!]$ la permutation associée à un rotor en position i. Montrez que

$$R_i^{-1}(x) = R_0^{-1}(x \boxminus i) \boxplus i \ ,$$

pour tout élément x de $[\![0, 25]\!]$.

3.3. Propriétés d'Enigma

Dans cette section, nous allons étudier deux propriétés importantes d'Enigma. Premièrement, nous allons montrer que la procédure de chiffrement est involutive. C'est grâce à cette propriété que le déchiffrement est aussi élégant.

Définition 2 (Involution). Soit $f : E \to E$ une application. On dit que f est *involutive* si $f \circ f = \mathrm{Id}_E$.

Exercice 4. Une involution est bijective

Soit $f : E \to E$ une involution. Montrez que f est bijective. Attention, vous ne devez pas utiliser les résultats du cours, montrez directement qu'elle est injective et surjective.

Nous allons maintenant formaliser les différents composants d'Enigma. Le tableau de connexions peut être vu comme une permutation P de l'ensemble $[\![0, 25]\!]$. De même, le réflecteur est une permutation involutive R de $[\![0, 25]\!]$. Les trois rotors sont appelés A, B, C et leurs positions courantes a, b et c. La permutation de $[\![0, 25]\!]$ associée au rotor A en position a est notée A_a. On définit de même les permutations B_b et C_c.

Exercice 5. Enigma est involutive

5.1) Avec les notations précédentes, explicitez la fonction de chiffrement

$$E_{a,b,c,P} : [\![0, 25]\!] \to [\![0, 25]\!]$$

associée aux rotors A, B, C en positions respectives a, b, c et à la permutation initiale P.

5.2) Montrez que $E_{a,b,c,P}$ est une involution.

5.3) Pour que $E_{a,b,c,P}$ soit involutive, est-il nécessaire que le réflecteur R soit involutif ? Même question pour le tableau de connexions P.

5.4) Déduisez-en la procédure de déchiffrement d'Enigma.

Nous allons maintenant découvrir une seconde propriété d'Enigma. Contrairement à la première, cette propriété n'était pas souhaitée par les créateurs de la machine puisqu'elle a permis aux alliés de casser le chiffrement.

Définition 3 (Point fixe). Soit $f : E \to E$ une application. On dit que x est un *point fixe* de f si $f(x) = x$.

Exercice 6. Pas de point fixe pour Enigma

Rappelons que le réflecteur R n'a aucun point fixe. Par l'absurde, supposons que la fonction de chiffrement E_k admette un point fixe $x \in \mathcal{A}$.

6.1) Posons $y = C_c \circ B_b \circ A_a \circ P(x)$. Montrer que $R(y) = y$.

6.2) Terminer la démonstration.

4. Partie programmation

Il est maintenant temps de réaliser la simulation d'une machine Enigma. Vous venez de voir dans la partie mathématiques la fonction vous permettant de chiffrer une lettre. L'objectif de cette partie est de coder le chiffrement (et donc le déchiffrement), sans oublier le paramétrage possible grâce à la clé. Enigma fonctionne sur l'alphabet traditionnel de 26 lettres. Pour plus de simplicité, nous utiliserons l'alphabet numérique $[\![0, 25]\!]$.

Le chiffrement complet sera présenté sous la forme d'un texte récupéré de la part de l'utilisateur en tant qu'argument de la ligne de commande. Le chiffrement de chaque lettre sera réalisé au travers de la fonction `Encrypt()` prenant en paramètre un entier inférieur à 26 (lettre en clair) et retournant la valeur du chiffré correspondant.

Exercice 7.

7.1) Codez le prototype de la fonction `Encrypt()`. Pour le moment, cette fonction doit simplement renvoyer la lettre passée en argument.

7.2) Dans votre `main()`, réalisez une boucle sur le texte clair donné par l'utilisateur, et appelez la fonction `Encrypt()` pour chaque lettre de ce clair.

4.1. Le tableau de connexions

Le tableau de connexions permet à l'opérateur de permuter 12 lettres de l'alphabet deux à deux. Une permutation est réalisée en C sous la forme d'un tableau de 26 cases dont la valeur de la case i représente l'image de la lettre i. Dans l'exemple du tableau de

connexions, l'application de permutation se présentera sous la forme d'un tableau dans lequel la case i contiendra la valeur i, sauf pour les 12 lettres permutées.

Exercice 8.

8.1) Définissez le tableau `connexions` en variable globale, représentant la permutation identité des 26 lettres de l'alphabet.

8.2) Réalisez la fonction `ConnectCables()` qui demande à l'utilisateur six fois deux lettres, et qui modifie le tableau `connexions` pour simuler le branchement des câbles sur le tableau de connexions. Attention, l'utilisateur ne peut pas permuter deux fois la même lettre !

8.3) Complétez la fonction `Encrypt()` pour calculer la valeur de la lettre après passage dans le tableau de connexions.

4.2. Premier passage dans les rotors

Chaque rotor est équivalent à une permutation des lettres de l'alphabet. La machine Enigma possédant trois rotors, nous avons besoin de trois tableaux simulant ces rotors. Afin d'optimiser la structure du code, ces trois tableaux seront regroupés sous la forme d'une matrice, définie en tant que variable globale, nommée `rotor` et composée de 3 lignes, 26 colonnes. Pour simplifier, l'ordre de ces rotors sera fixe.

Exercice 9. Déclaration des rotors

9.1) Définissez et initialisez une matrice `textRotor`, réalisant la permutation des lettres indiquées dans le tableau suivant (source : `https://en.wikipedia.org/wiki/Enigma_rotor_details`).

Rotor I	`EKMFLGDQVZNTOWYHXUSPAIBRCJ`
Rotor II	`AJDKSIRUXBLHWTMCQGZNPYFVOE`
Rotor III	`BDFHJLCPRTXVZNYEIWGAKMUSQO`

9.2) Puisque nous manipulons des entiers et non des caractères, rédigez la fonction `Text2Value()` permettant de convertir les caractères de la matrice `textRotor` en entiers dans la matrice `rotor`.

9.3) Complétez la fonction `Encrypt()` pour simuler le passage de la lettre au travers de ces trois rotors.

4.3. Le réflecteur

Après que le signal soit passé au travers des trois rotors, il arrive sur un réflecteur permettant de renvoyer le signal sur les rotors, et ainsi de rendre le chiffrement involutif. Ce réflecteur peut être vu lui aussi comme une permutation involutive des lettres, sans point fixe.

Exercice 10.

10.1) Définissez et initialisez un tableau `textReflector`, réalisant la permutation des lettres indiquée dans le tableau suivant. Convertissez ce texte en un tableau d'entiers nommé `reflector` réalisant la permutation donnée par le réflecteur en complétant la fonction `Text2Value()`.

`EJMZALYXVBWFCRQUONTSPIKHGD`

10.2) Complétez la fonction `Encrypt()` pour simuler le passage de la lettre au travers de ce réflecteur.

4.4. Second passage dans les rotors

Le réflecteur renvoie donc le signal dans les trois rotors, en commençant par le dernier. L'application associée est donc une permutation inverse des rotors. Là où « `B` » était transformé en « `K` » dans le rotor 1, « `K` » doit être transformé en « `B` » lors du second passage dans ce même rotor, et non en « `N` ». Il vous faut donc calculer dans un premier temps la permutation inverse de chaque rotor avant de l'appliquer à la lettre courante. Pour cela, sachant que $R(i)$ représente l'image de la lettre i par l'application R, vous devez calculer l'image inverse $R^{-1}(i)$ pour tout entier $i \leq 26$. En sachant que $R^{-1}(R(i)) = i$, le codage de cette fonction en C ne devrait vous poser aucun problème.

Exercice 11.

11.1) Définissez la matrice `invRotor` formée de trois lignes, 26 colonnes en variable globale.

11.2) Réalisez la fonction `GenerateInvPerm()` qui crée la permutation inverse de chaque rotor en remplissant le tableau `invRotor`. La ligne i de cette matrice contiendra la permutation inverse de la ligne i de la matrice `rotor`.

11.3) Complétez la fonction `Encrypt()` pour simuler le passage de la lettre au travers de ces trois rotors, en sens inverse.

4.5. Tableau de connexion et rotation des rotors

Il ne reste plus au signal qu'à traverser à nouveau le tableau de connexions, ce qui sera simulé une nouvelle fois à l'aide du tableau `connexions`. La dernière partie du chiffrement consiste en la rotation des rotors. Vous avez vu dans la partie mathématiques que cette rotation est simulée sous la forme d'une action $+k/-k$ lors de l'accès à une case du tableau de rotor pour la k-ieme lettre à chiffrer.

Remarque 4. Attention, le modulo en C ne se comporte pas exactement comme en mathématiques. Que vous renvoie l'instruction `-27%26` ?

Rappelons qu'après chaque lettre chiffrée, le premier rotor tourne d'un cran. Dès que ce rotor revient sur la position A, le second rotor tourne d'un cran, de même pour le dernier rotor par rapport au second. Il est donc nécessaire pour le codage du chiffrement de mémoriser le décalage actuel de chaque rotor. Cela est réalisé à l'aide d'un tableau de trois entiers, dont la case i indique le décalage du rotor i. Ce tableau nommé `shift` sera déclaré en variable globale.

Exercice 12.

12.1) Modifiez votre `main()` pour récupérer de la part de l'utilisateur les positions initiales des rotors et modifiez le tableau `shift` en conséquences.

12.2) Complétez la fonction `Encrypt()` pour simuler le passage de la lettre au travers du tableau de permutation.

12.3) Complétez la fonction `Encrypt()` pour simuler la rotation des rotors.

12.4) Il ne vous reste plus qu'à retourner la valeur de la lettre chiffrée pour terminer votre programme.

5. Challenge

Parce que tout bon TP de cryptographie se doit de comporter un challenge, voici quelques questions pour conclure cette séquence mathématiques informatique. On suppose pour tout l'exercice suivant, les rotors et réflecteurs présentés ci-dessus ont été utilisés pour obtenir les messages suivants.

Exercice 13.

13.1) Nos agents ont intercepté le texte suivant.

```
DZEPWXVLKNJPDFMJSFOCZWXVWKSFMJMIEYVW
YTXVXJOVPSNFEYIPJRRSFMNDTJXBYOLTHARF
```

Ils savent également que le tableau de permutation échangeait par paire les lettres `E - J`, `H - R` et `I - X`. Quel est donc ce message ?

13.2) Nos agents ont également récupéré le message suivant, mais visiblement, les émetteurs ont du modifier un des éléments de la clé. Pouvez-vous identifier lequel ?

```
ACVQGFYMKJWZEUKTDYVGKWRXQDDZGZKGRLQIRBNZEJUZSRWEMQKYEKFCZA
```