Sécurité

634-2.1-Sécurité technique

haute école neuchâtel berne jura — arc gestion neuchâtel delémont

AUTEUR	Maël Voyame & Bastien Mittempergher
CLASSE	3IG-TPart
ANNÉE	3ème année 2017/2018
SEMESTRE	S6
MODULE	634-2-Sécurité des SI
COURS	634-2.1-Sécurité technique
DOMAINE	Sécurité

Suivi des modifications

DATE	VERSION	MODIFICATION	AUTEUR
13.05.2018	1.0	Rédaction du document	Maël Voyame

Table des matières

1 INTRODUCTION

1.1 Technologie utilisées

Pour ce projet, nous avons utilisé le langage JAVA. L'application se présente sous forme de ligne de commande.

2 STRUCTURE DU PROJET

2.1 Classe – ADFGVX

Cette classe définit le tableau de substitution nécessaire au cryptage du message. Le tableau est défini manuellement. Il ne peut pas être choisi. Pour le projet, nous avons utilisé le tableau du support de cours.

```
public class ADFGVX {
    private HashMap<String, HashMap<String, String>> tableau_substitution;


    public ADFGVX(){
        this.init_tableau();
    }

    public HashMap<String, HashMap<String, String>> getTableau_substitution() {
        return tableau_substitution;
    }

    public void init_tableau(){
      tableau_substitution = new HashMap<String, HashMap<String, String>>();

      HashMap<String, String> temp_A = new HashMap<String, String>();
      HashMap<String, String> temp_D = new HashMap<String, String>();
      HashMap<String, String> temp_F = new HashMap<String, String>();
      HashMap<String, String> temp_G = new HashMap<String, String>();
      HashMap<String, String> temp_V = new HashMap<String, String>();
      HashMap<String, String> temp_X = new HashMap<String, String>();

      temp_A.put("A", "c");
      temp_A.put("D", "1");
      temp_A.put("F", "o");
      temp_A.put("G", "f");
      temp_A.put("V", "w");
      temp_A.put("X", "j");

      tableau_substitution.put("A", temp_A);

      temp_D.put("A", "y");
      temp_D.put("D", "m");
      temp_D.put("F", "t");
      temp_D.put("G", "5");
      temp_D.put("V", "b");
      temp_D.put("X", "4");

      tableau_substitution.put("D", temp_D);

      temp_F.put("A", "i");
      temp_F.put("D", "7");
      temp_F.put("F", "a");
      temp_F.put("G", "2");
      temp_F.put("V", "8");
      temp_F.put("X", "s");

      tableau_substitution.put("F", temp_F);

      temp_G.put("A", "p");
      temp_G.put("D", "3");
      temp_G.put("F", "0");
      temp_G.put("G", "q");
      temp_G.put("V", "h");
      temp_G.put("X", "x");

      tableau_substitution.put("G", temp_G);

      temp_V.put("A", "k");
      temp_V.put("D", "e");
      temp_V.put("F", "u");
      temp_V.put("G", "l");
      temp_V.put("V", "6");
      temp_V.put("X", "d");

      tableau_substitution.put("V", temp_V);

      temp_X.put("A", "v");
      temp_X.put("D", "r");
      temp_X.put("F", "g");
      temp_X.put("G", "z");
      temp_X.put("V", "n");
      temp_X.put("X", "9");

      tableau_substitution.put("X", temp_X);
    }
}
```

	A	D	F	G	V	X
A	c	1	o	f	w	j
D	y	m	t	5	b	4
F	i	7	a	2	8	s
G	p	3	0	q	h	x
V	k	e	u	l	6	d
X	v	r	g	z	n	9

2.2 Classe - Cryptage

Cette classe offre toutes les méthodes nécessaires au chiffrement et déchiffrement du message. Chaque méthode est détaillée dans ce document.

2.2.1 getTexteIntermediaireCrypte

Cette méthode permet d'obtenir le texte intermédiaire sur la base du tableau de substitution. Elle renvoi une liste de String qui contient les couples de valeurs (A-D-F-G-V-X) pour chaque caractère du message.

```
/**
 * Cette méthode va scinder le texte à chiffrer dans un tableau (une case un caractère),
 * elle va ensuite parcourir le tableau de substitution et tester la correspondance des valeurs.
 * Quand le test passe, ajoute le couple de clef au tableau de retour.
 * @param text Texte à chiffrer
 * @return ArrayList de string correspondant au texte chiffré intermédiaire
 */
public static ArrayList<String> getTexteIntermediaireCrypte (String text){
    // Récupération du tableau de substitution
    ArrayList<String> tabTI = new ArrayList<>();
    HashMap<String, HashMap<String, String>> tabSub = new HashMap<>(tab.getTableau_substitution());

    // Sépaaration du texte en tableau de String[]
    String[] tabSplit = text.split("");

    // Parcours du tableau de String[]
    for (int i = 0; i < tabSplit.length; i++) {
        String current = tabSplit[i];

        // Parcours du tableau de substitution
        for (Map.Entry<String, HashMap<String, String>> en : tabSub.entrySet()) {
        Object key = en.getKey();
        Object value = en.getValue();

        // Récupération de la ligne du tableau de substitution
        HashMap<String, String> tabSubLine = new HashMap<>(tabSub.get(key));

            // Parcours de la ligne du tableau de substitution
            for (Map.Entry<String, String> entry : tabSubLine.entrySet()) {
                String keyl = entry.getKey();
                String valuel = entry.getValue();

                // Test de la valeur du String courrent et de la valeur de la case
                if(valuel.equals(current)){

                    // Ajout du couple de clefs correspondant dans le tableau de retour
                    tabTI.add((String) key);
                    tabTI.add(keyl);
                }
            }
        }
    }
    return tabTI;
}
```

Cette méthode permet de faire la manipulation suivante :

Texte clair	o	b	j	e	c	t	i	f	a	r	r	a	s	1	5	h	2	8
Texte chiffré intermédiaire	AF	DV	AX	VD	AA	DF	FA	AG	FF	XD	XD	FF	FX	AD	DG	GV	FG	FV

2.2.2 *getTableOrderedCrypte*

Cette méthode permet de transposer le texte intermédiaire dans un tableau. Elle utilise un code secret pour remplir ce tableau. Ce tableau comporte en entête le code secret.

```java
/**
 * Reçoit le TCI et le mot de passe, et remplis le tableau
 * @param TCI Texte chiffré intermédiaire à passer dans le tableau
 * @param pwd Mot de passe pour le cryptage
 * @return
 */
public ArrayList<ArrayList<String>> getTableOrderedCrypte(ArrayList<String> TCI, String pwd){
    // Déclaration des lignes du tableau
    ArrayList<ArrayList<String>> tableOrdered = new ArrayList<>();
    // ArrayList<String> tableLine0 = new ArrayList<>();
    ArrayList<String> tableLine1 = new ArrayList<>();
    ArrayList<String> tableLine2 = new ArrayList<>();
    ArrayList<String> tableLine3 = new ArrayList<>();
    ArrayList<String> tableLine4 = new ArrayList<>();
    ArrayList<String> tableLine5 = new ArrayList<>();
    ArrayList<String> tableLine6 = new ArrayList<>();

    // Séparation du mot de passe
    String[] tabSplit = pwd.split("");

    for (int i = 0; TCI.size()<37; i++) {
        TCI.add("F");
    }
    tableLine1.add(tabSplit[0]);
    tableLine2.add(tabSplit[1]);
    tableLine3.add(tabSplit[2]);
    tableLine4.add(tabSplit[3]);
    tableLine5.add(tabSplit[4]);
    tableLine6.add(tabSplit[5]);
    for (int i = 0; i < 36; i = i+6) {
        if(!TCI.get(i).isEmpty()){
            tableLine1.add(TCI.get(i));
            tableLine2.add(TCI.get(i+1));
            tableLine3.add(TCI.get(i+2));
            tableLine4.add(TCI.get(i+3));
            tableLine5.add(TCI.get(i+4));
            tableLine6.add(TCI.get(i+5));
        }
    }


    // Ajout des ligne dans le tableau pour le retour
    // tableOrdered.add(tableLine0);
    tableOrdered.add(tableLine1);
    tableOrdered.add(tableLine2);
    tableOrdered.add(tableLine3);
    tableOrdered.add(tableLine4);
    tableOrdered.add(tableLine5);
    tableOrdered.add(tableLine6);

    return tableOrdered;
}
```

Sur la base tu texte intermédiaire, on obtiendra un tableau de cette forme :

Grille 1

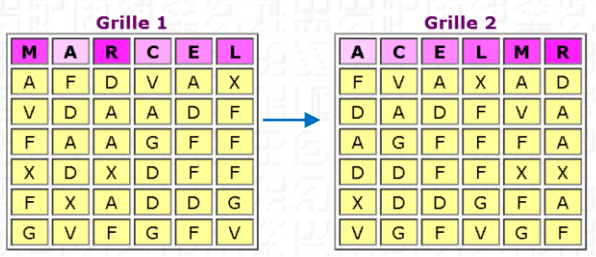
M	A	R	C	E	L
A	F	D	V	A	X
V	D	A	A	D	F
F	A	A	G	F	F
X	D	X	D	F	F
F	X	A	D	D	G
G	V	F	G	F	V

2.2.3 *getTableCrypted*

Cette méthode de réorganiser le tableau de l'étape précédente. Le code secret sera trié par ordre alphabétique. Donc, toutes les valeurs seront également réorganisées.

```java
/**
 * Prend le tableau ordonné et le mot de passe afin de les parcourir,
 * puis tester la valeur de la lettre d'en-tête (mot de passe), puis ajoute la colonne à son nouvel emplacement.
 * @param TO Tableau ordonné
 * @param pwd Mot de passe
 * @return Tableau crypté
 */
public ArrayList<ArrayList<String>> getTableCrypted(ArrayList<ArrayList<String>> TO, String pwd){
    // Déclaration des variables
    ArrayList<ArrayList<String>> tableCrypted = new ArrayList<>();
    ArrayList<String> lineTemp = new ArrayList<>();
    String lp1, lp2, lp3, lp4, lp5, lp6;

    // Ajout de ligne vide
    for (int i = 0; i < 6; i++) {
        tableCrypted.add(i, null);
    }

    // Séparation du mot de passe et tri alphabethique
    ArrayList tabSplit = new ArrayList<>();
    for (int i = 0; i < 6; i++) {
        tabSplit.add(pwd.split("")[i]);
    }
    Collections.sort(tabSplit);

    // Ajout des lettres du mot de passe dans des variables
    lp1 = (String) tabSplit.get(0);
    lp2 = (String) tabSplit.get(1) ;
    lp3 = (String) tabSplit.get(2) ;
    lp4 = (String) tabSplit.get(3) ;
    lp5 = (String) tabSplit.get(4) ;
    lp6 = (String) tabSplit.get(5) ;

    // Parcours de la table ordonnée (TO)
    for (int i = 0; i < TO.size(); i++) {
        lineTemp = (TO.get(i));
        String currentL = lineTemp.get(0);

        // Test des valeurs de la clef
        if(currentL.equals(lp1)){
            // Remove la ligne null
            tableCrypted.remove(0);
            // Ajoute un clone de la colonne en cours
            tableCrypted.add(0, (ArrayList<String>) lineTemp.clone());
        }
        if(currentL.equals(lp2)){
            tableCrypted.remove(1);
            tableCrypted.add(1, (ArrayList<String>) lineTemp.clone());
        }
        if(currentL.equals(lp3)){
            tableCrypted.remove(2);
            tableCrypted.add(2, (ArrayList<String>) lineTemp.clone());
        }
        if(currentL.equals(lp4)){
            tableCrypted.remove(3);
            tableCrypted.add(3, (ArrayList<String>) lineTemp.clone());
        }
        if(currentL.equals(lp5)){
            tableCrypted.remove(4);
            tableCrypted.add(4, (ArrayList<String>) lineTemp.clone());
        }
        if(currentL.equals(lp6)){
            tableCrypted.remove(5);
            tableCrypted.add(5, (ArrayList<String>) lineTemp.clone());
        }
    }

    return tableCrypted;
}
```

Cette méthode permet de faire la manipulation suivante (Grille 1 à Grille 2) :



2.2.4 *getTexteCrypted*

Cette méthode permet d'obtenir le texte chiffré sur la base du tableau précédent. Le tableau de l'étape précédente sera lu de haut en bas et de gauche à droite pour constituer le message chiffré.

```
/**
 * Reçoit le tableau crypté (TC), le parcours
 * puis retourne le texte final
 * @param TC Tableau crypté
 * @return le texte final
 */
public String getTexteCrypted(ArrayList<ArrayList<String>> TC){
    StringBuilder CT = new StringBuilder();

    // parcours des colonnes
    for (int i = 0; i < 6; i++) {

        // parcours des lignes
        for (int j = 1; j < 7; j++) {

            // Ajoute l'élément j qui se trouve dans la colonne i
            CT.append(TC.get(i).get(j));
        }
    }
    return CT.toString();
}
```

A la sortie de cette méthode, nous obtiendrons un message sous cette forme :

```
FDADX VVAGD DGADF FDFXF FFGVA VFXFG DAAXA F
```

2.2.5 *getTableOrderedDecrypt*

Cette méthode permet d'obtenir le tableau avec le mot clé trié alphabétiquement sur la base du texte chiffré.

```java
/**
 * Retourne le tableau crypté avec le mot clé trié alphabétiquement
 * @param CT Texte chiffré que l'on veut décrypter
 * @param pwd Mot clé utilisé pour crypter
 * @return Le tableau trié avec le mot clé
 */
public ArrayList<ArrayList<String>> getTableOrderedDecrypt(String CT, String pwd){
    //Initialisation
    ArrayList<ArrayList<String>> tableOrdered = new ArrayList<>();
    ArrayList<String> motCle = new ArrayList<>();
    ArrayList<String> textChiffre = new ArrayList<>();
    ArrayList<String> lineTemp = new ArrayList<>();

    //Ajoute le mot clé dans une liste et le trie par ordre alphabétique
    for (int i = 0; i < 6; i++) {
        motCle.add(pwd.split("")[i]);
    }
    Collections.sort(motCle);

    //Récupère le texte chiffré dans une liste pour la parcourir
    for (int i = 0; i < CT.length(); i++) {
        textChiffre.add(CT.split("")[i]);
    }

    //Préparation du tableau crypté ordonné
    int indexMin = 0;
    int indexMax = 6;
    for (int i = 0; i < pwd.length(); i++){
        lineTemp.add(motCle.get(i));
        for(int j = indexMin; j < indexMax; j++){
            lineTemp.add(textChiffre.get(j));
        }
        indexMin = indexMin+6;
        indexMax = indexMax+6;
        tableOrdered.add(i , (ArrayList<String>) lineTemp.clone());
        lineTemp.clear();
    }

    return tableOrdered;
}
```

Cette méthode permet de passer du texte chiffré à un tableau de cette forme :

Grille 2

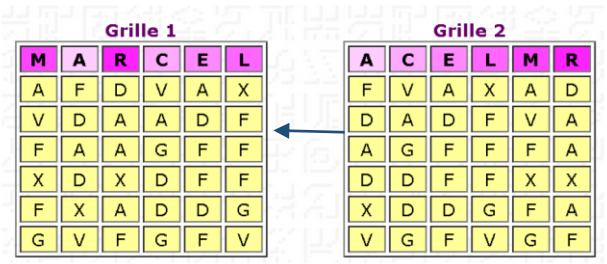
A	C	E	L	M	R
F	V	A	X	A	D
D	A	D	F	V	A
A	G	F	F	F	A
D	D	F	F	X	X
X	D	D	G	F	A
V	G	F	V	G	F

2.2.6 *getTableDecrypted*

Cette méthode permet de transposer le tableau avec le code secret trié alphabétiquement dans un tableau avec le code secret dans le bon ordre.

```java
/**
 * Retourne le tableau décrypté avec le mot clé dans le bon ordre
 * @param tableauOrdonne Tableau trié alphabétiquement sur le mot clé
 * @param pwd Mot clé utilisé pour le cryptage
 * @return Retourne le tableau décrypté avec le mot clé dans le bon ordre
 */
public ArrayList<ArrayList<String>> getTableDecrypted(ArrayList<ArrayList<String>> tableauOrdonne, String pwd){
    //Initialisation
    ArrayList<ArrayList<String>> tableDecrypted = new ArrayList<>();
    ArrayList<String> motCle = new ArrayList<>();

    //Ajoute le mot clé dans une ligne temporaire et le trie par ordre alphabétique
    for (int i = 0; i < pwd.length(); i++) {
        motCle.add(pwd.split("")[i]);
    }

    int compteur = 0;
    for(int i = 0; i < pwd.length(); i++){
        for(int j = 0; j < pwd.length(); j++){
            //Si dans le tableau ordonné on trouve la première lettre ("h" - motCle.get(compteur)) etc...
            // on ajoute cette liste dans le tableau décrypté
            if(tableauOrdonne.get(j).get(0).equals(motCle.get(compteur))){
                tableDecrypted.add(tableauOrdonne.get(j));
            }
        }
        compteur++;
    }

    return tableDecrypted;
}
```

Cette méthode permet de réaliser la manipulation suivante :

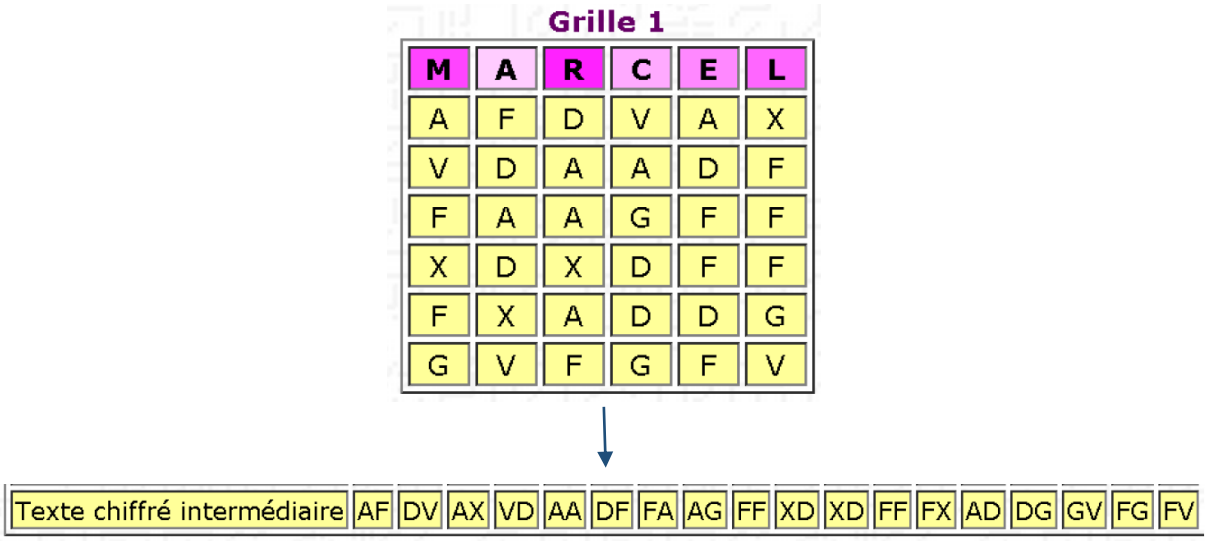


2.2.7 *getTexteIntermediaireDecrypte*

Cette méthode permet de passer du tableau avec le code secret dans le bon ordre dans un tableau qui contient le couple de clés pour retrouver les bons caractères dans le tableau de substitution.

```java
/**
 * Retourne le texte intermediaire sous forme d'ArrayList d'ArrayList
 * @param tableDecrypted Tableau décrypté (avec le mot clé dans le bon ordre)
 * @return Le texte intermediaire sous forme de liste
 */
public ArrayList<ArrayList<String>> getTexteIntermediaireDecrypte(ArrayList<ArrayList<String>> tableDecrypted){
    //Initialisation
    ArrayList<ArrayList<String>> tableTextIntermediaire = new ArrayList<>();
    ArrayList<String> lineTemp = new ArrayList<>();

    int indexTabTextInter = 0;
    int indexTabDecrypted2 = 1;
    int indexTabDecrypted1 = 0;
    int checkClear = 0;
    //Parcours du tableau décrypté pour remplir le tableau avec le texte intermediaire
    //pour les 6 emplacements des lettres (chaque ArrayList compte 6 caractère ADFGVX)
    for(int i = 0; i < 6; i++){
        // j < 9 car 3*2 lineTemp + 2*Clear = 8
        //pour chaque ArrayList contenu dans l'ArrayList tableDecrypted
        //on récupère un caractère pour former des couples
        //ces couples seront utilisé pour le tableau de substitution
        for(int j = 0; j < 9; j++){
            if(checkClear == 2){
                tableTextIntermediaire.add(indexTabTextInter, (ArrayList<String>) lineTemp.clone());
                indexTabTextInter++;
                lineTemp.clear();
                checkClear = 0;
            }else{
                lineTemp.add(tableDecrypted.get(indexTabDecrypted1).get(indexTabDecrypted2));
                indexTabDecrypted1++;
                checkClear++;
            }
        }
        indexTabDecrypted1 = 0;
        indexTabDecrypted2++;
    }

    return tableTextIntermediaire;
}
```

Cette méthode permet la manipulation suivante :



2.2.8 *getMessageDecrypte*

Cette méthode permet d'obtenir le message décrypté sur la base du tableau contenant le texte intermédiaire. Le texte intermediaire contient le couple de clés permettant de retrouver le bon caractère dans le tableau de substitution.

```java
/**
 * Retourne le message déchiffré
 * @param tableTextIntermediaire tableau contenant le texte intermediaire
 * @return Le message déchiffré
 */
public StringBuilder getMessageDecrypte(ArrayList<ArrayList<String>> tableTextIntermediaire){
    //Initialisation
    StringBuilder message = new StringBuilder();
    String fragment_message;
    String keyOne;
    String keyTwo;

    // Récupération du tableau de substitution
    HashMap<String, HashMap<String, String>> tabSub = new HashMap<>(tab.getTableau_substitution());

    //le tableau de texte intermediaire contient 18 position
    for(int i = 0; i < 18; i++){
        //Récupération des couples de clé pour trouver la correspondance
        //dans le tableau de substitution
        keyOne = tableTextIntermediaire.get(i).get(0);
        keyTwo = tableTextIntermediaire.get(i).get(1);

        //Recherche de la valeur dans le tableau de substitution
        fragment_message = tabSub.get(keyOne).get(keyTwo);

        //Ajoute le fragment au message
        message.append(fragment_message);
    }

    return message;
}
```

Cette méthode permet la manipulation suivante :

Texte clair	o	b	j	e	c	t	i	f	a	r	r	a	s	1	5	h	2	8
Texte chiffré intermédiaire	AF	DV	AX	VD	AA	DF	FA	AG	FF	XD	XD	FF	FX	AD	DG	GV	FG	FV

2.2.9 *toStringTable*

Cette méthode permet d'afficher les tableaux qui contiennent le code secret.

```java
/**
 * Affiche un pseudo tableau
 * @param TO Un tableau de liste
 */
public void toStringTable(ArrayList<ArrayList<String>> TO){

    // parcours des lignes
    for (int i = 0; i < 7; i++) {
        System.out.print("|");

        // parcours des colonnes
        for (int j = 0; j < 6; j++) {

            // Affiche l'élément i qui se trouve dans la colonne j
            System.out.print(TO.get(j).get(i) + "|");
        }
        System.out.print("\n------------\n");
    }
}
```

Cette méthode produit le résultat suivant :

```
|h|u|m|i|d|e|
------------
|V|D|G|A|F|F|
------------
|G|A|F|A|V|G|
------------
|V|G|A|F|X|V|
------------
|V|D|F|X|D|F|
------------
|D|V|V|D|F|F|
------------
|V|F|F|F|F|F|
------------
```

2.3 Classe – Application

La classe Application produit 2 résultats. Le premier résultat affiche le chiffrement/déchiffrement d'un message avec toutes les étapes pas à pas.

```
/**
 * PARTIE 1 - Affichage pas à pas
 */
//Partie cryptage
//Message à crypter --> Texte Intermediaire
ArrayList<String> TCI = cr.getTexteIntermediaireCrypte("Ce papillon est beau");
System.out.println("Message à chiffrer :\n");
System.out.println("\t" + "Ce papillon est beau\n\n");
System.out.println("Texte intermediaire du message :\n");
System.out.println("\t" + TCI);
System.out.println("\n\n");

//Texte Intermediaire --> Tableau Ordonné
ArrayList<ArrayList<String>> TO = cr.getTableOrderedCrypte(TCI, "humide");
System.out.println("Tableau Ordonné avec le mot clé 'humide' :\n");
cr.toStringTable(TO);
System.out.println("\n\n");

//Tableau Ordonné --> Tableau Chiffré
ArrayList<ArrayList<String>> TC = cr.getTableCrypted(TO, "humide");
System.out.println("Tableau Chiffré (transposition du tableau ordonné):\n");
cr.toStringTable(TC);
System.out.println("\n\n");

//Tableau Chiffré --> Message Crypté
String CT = cr.getTexteCrypted(TC);
System.out.println("Message crypté :\n");
System.out.println(CT);
System.out.println("\n\n----------------------------------------\n\n");
```

```
Message à chiffrer :

        Ce papillon est beau


Texte intermediaire du message :

        [V, D, G, A, F, F, G, A, F, A, V, G, V, G, A, F, X, V, V, D, F, X, D, F, D, V, V, D, F, F, V, F]


Tableau Ordonné avec le mot clé 'humide' :

|h|u|m|i|d|e|
------------
|V|D|G|A|F|F|
------------
|G|A|F|A|V|G|
------------
|V|G|A|F|X|V|
------------
|V|D|F|X|D|F|
------------
|D|V|V|D|F|F|
------------
|V|F|F|F|F|F|
------------
```

```
Tableau Chiffré (transposition du tableau ordonné):

|d|e|h|i|m|u|
------------
|F|F|V|A|G|D|
------------
|V|G|G|A|F|A|
------------
|X|V|V|F|A|G|
------------
|D|F|V|X|F|D|
------------
|F|F|D|D|V|V|
------------
|F|F|V|F|F|F|
------------
```

```
Message crypté :

FVXDFFFGVFFFVGVVDVAAFXDFGFAFVFDAGDVF
```

```java
//Partie décryptage
//Message Crypté --> Tableau Chiffré
ArrayList<ArrayList<String>> tableCrypteOrdonne = cr.getTableOrderedDecrypt(CT, "humide");
System.out.println("Tableau Chiffré sur la base du message crypté :\n");
cr.toStringTable(tableCrypteOrdonne);
System.out.println("\n\n");

//Tableau Chiffré --> Tableau Ordonné
ArrayList<ArrayList<String>> tableDecrypte = cr.getTableDecrypted(tableCrypteOrdonne, "humide");
System.out.println("Tableau Ordonné sur la base du tableau chiffré :\n");
cr.toStringTable(tableDecrypte);
System.out.println("\n\n");

//Tableau Ordonné --> Tableau Texte Intermediaire
ArrayList<ArrayList<String>> tableTextInter = cr.getTexteIntermediaireDecrypte(tableDecrypte);
System.out.println("Tableau du texte intermediaire :\n");
System.out.println(tableTextInter);
System.out.println("\n\n");

//Tableau Text Intermediaire --> Message Décrypté
StringBuilder message = cr.getMessageDecrypte(tableTextInter);
System.out.println("Message décrypté :\n");
System.out.println("\t" + message.toString() + "\n\n");
```

Tableau Chiffré sur la base du message crypté :

```
|d|e|h|i|m|u|
------------
|F|F|V|A|G|D|
------------
|V|G|G|A|F|A|
------------
|X|V|V|F|A|G|
------------
|D|F|V|X|F|D|
------------
|F|F|D|D|V|V|
------------
|F|F|V|F|F|F|
------------
```

Tableau Ordonné sur la base du tableau chiffré :

```
|h|u|m|i|d|e|
------------
|V|D|G|A|F|F|
------------
|G|A|F|A|V|G|
------------
|V|G|A|F|X|V|
------------
|V|D|F|X|D|F|
------------
|D|V|V|D|F|F|
------------
|V|F|F|F|F|F|
------------
```

Tableau du texte intermediaire :

```
[[V, D], [G, A], [F, F], [G, A], [F, A], [V, G], [V, G], [A, F], [X, V], [V, D], [F, X], [D, F], [D, V], [V, D], [F, F], [V, F], [F, F], [F, F]]
```

Message décrypté :

epapillonestbeauaa

La seconde partie permet à l'utilisateur de choisir le message et le code secret pour chiffrer.

```java
/**
 * PARTIE 2 - Saisie utilisateur
 */
Scanner sc = new Scanner(System.in);
System.out.println("Veuillez saisir le message que vous voulez crypter : ");
String messageUser = sc.nextLine();
System.out.println("\nVeuillez saisir le code secret pour chiffrer le message : ");
String pwdUser = sc.nextLine();

ArrayList<String> TCIUser = cr.getTexteIntermediaireCrypte(messageUser);
ArrayList<ArrayList<String>> TOUser = cr.getTableOrderedCrypte(TCIUser, pwdUser);
ArrayList<ArrayList<String>> TCUser = cr.getTableCrypted(TOUser, pwdUser);
String CTUser = cr.getTexteCrypted(TCUser);
System.out.println("\nVotre message est chiffré : " + CTUser);

System.out.println("\n\nVeuillez saisir le message crypté pour le déchiffré : ");
String messageChiffreUser = sc.nextLine();
System.out.println("\nVeuillez saisir le code secret pour déchiffrer le message : ");
String pwd2User = sc.nextLine();

ArrayList<ArrayList<String>> tableCrypteOrdonneUser = cr.getTableOrderedDecrypt(messageChiffreUser, pwd2User);
ArrayList<ArrayList<String>> tableDecrypteUser = cr.getTableDecrypted(tableCrypteOrdonneUser, pwd2User);
ArrayList<ArrayList<String>> tableTextInterUser = cr.getTexteIntermediaireDecrypte(tableDecrypteUser);
StringBuilder messageDecrypteUser = cr.getMessageDecrypte(tableTextInterUser);
System.out.println("\n\nMessage décrypté :\n");
System.out.println("\t" + messageDecrypteUser.toString());
```

```
Veuillez saisir le message que vous voulez crypter :
rdvdemain12h

Veuillez saisir le code secret pour chiffrer le message :
maelvo

Votre message est chiffré : DXFDFFVVFFFFXDAGFFXVFAFFADVVFFXDXGFF

Veuillez saisir le message crypté pour le déchiffré :
DXFDFFVVFFFFXDAGFFXVFAFFADVVFFXDXGFF

Veuillez saisir le code secret pour déchiffrer le message :
maelvo


Message décrypté :

        rdvdemain12haaaaaa
```