P_Secu 114

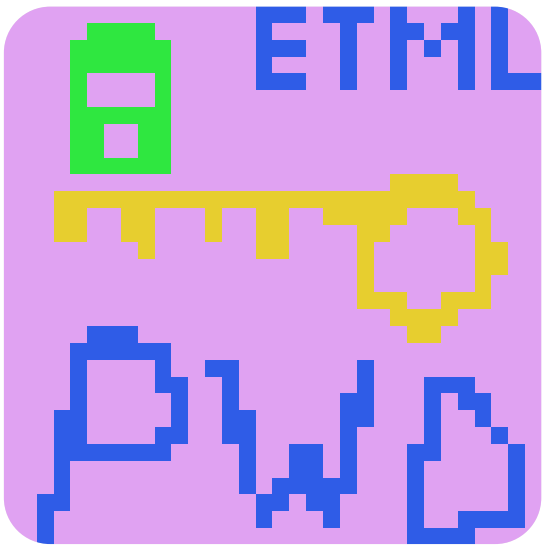

Figure 1 - Logo de l'application

Segalen Alban – mid2b
Sébeillon
24p
Helder Costa Lopes

Table des matières

1 INTRODUCTION

1.1 Titre

P_Secu 114 : Gestionnaire de mot de passe

1.2 Description

Réaliser un gestionnaire de mot de passeCahier des charges

1.2.1 Objectifs et portée du projet (objectifs SMART)

Réaliser une application qui permet de gérer ses mots de passes.

1.2.2 Fonctionnalités requises (du point de vue de l'utilisateur)

L'utilisateur utilise un Master Password pour accéder aux mots de passes. Le Master Password est utilisé comme clé de Vigenère.

On peut ajouter, modifier, supprimer un mot de passe.

Un mot de passe contient : un titre, l'URL du site, le nom d'utilisateur et le mot de passe.

1.3 Les points suivants seront évalués

- Le rapport
- Les planifications (initiale et détaillée)
- Le journal de travail
- Le code et les commentaires
- Les documentations de mise en œuvre et d'utilisation

1.4 Validation et conditions de réussite

- Compréhension du travail
- Possibilité de transmettre le travail à une personne extérieure pour le terminer, le corriger ou le compléter
- Etat de fonctionnement du produit livré

2 DESCRIPTION GÉNÉRALE

2.1 Présentation de l'application

L'application est un gestionnaire de mots de passes sécurisés, réalisés en windows forms. Le chiffrement des mots de passes se fait en utilisant le chiffrement de Vigenère.

2.2 Fonctionnalités principales

Ajouter, supprimer, modifier et sauvegarder des mots de passes de manières sécurisé.

3 ALGORITHME ET CONCEPTION

3.1 Chiffrement de Vigenère

Récupérer la valeur du premier caractère du message. Y ajouter la valeur du premier caractère de la clé. Y soustraire le nombre de possibilité (fait automatiquement avec c#). Répéter l'opération avec la deuxième lettre du message et de la clé. Si on arrive au bout de la clé, on prendre la première lettre de la clé.

https://eduvaud.sharepoint.com/sites/msteams_3cc8eb/Supports%20de%20cours/Forms/AllItems.aspx?id=%2Fsites%2Fmsteams%5F3cc8eb%2FSupports%20de%20c

ours%2FICT%2F114
2DCodificationChiffrement%2Fb%2DUnitesEnseignement%2FS%2D114%2DCompe
ndio%2Epdf&parent=%2Fsites%2Fmsteams%5F3cc8eb%2FSupports%20de%20cours
%2FICT%2F114%2DCodificationChiffrement%2Fb%2DUnitesEnseignement

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

Figure 2 - Table de Vigenère - ictteachersug.net

3.2 Schémas des algorithmes

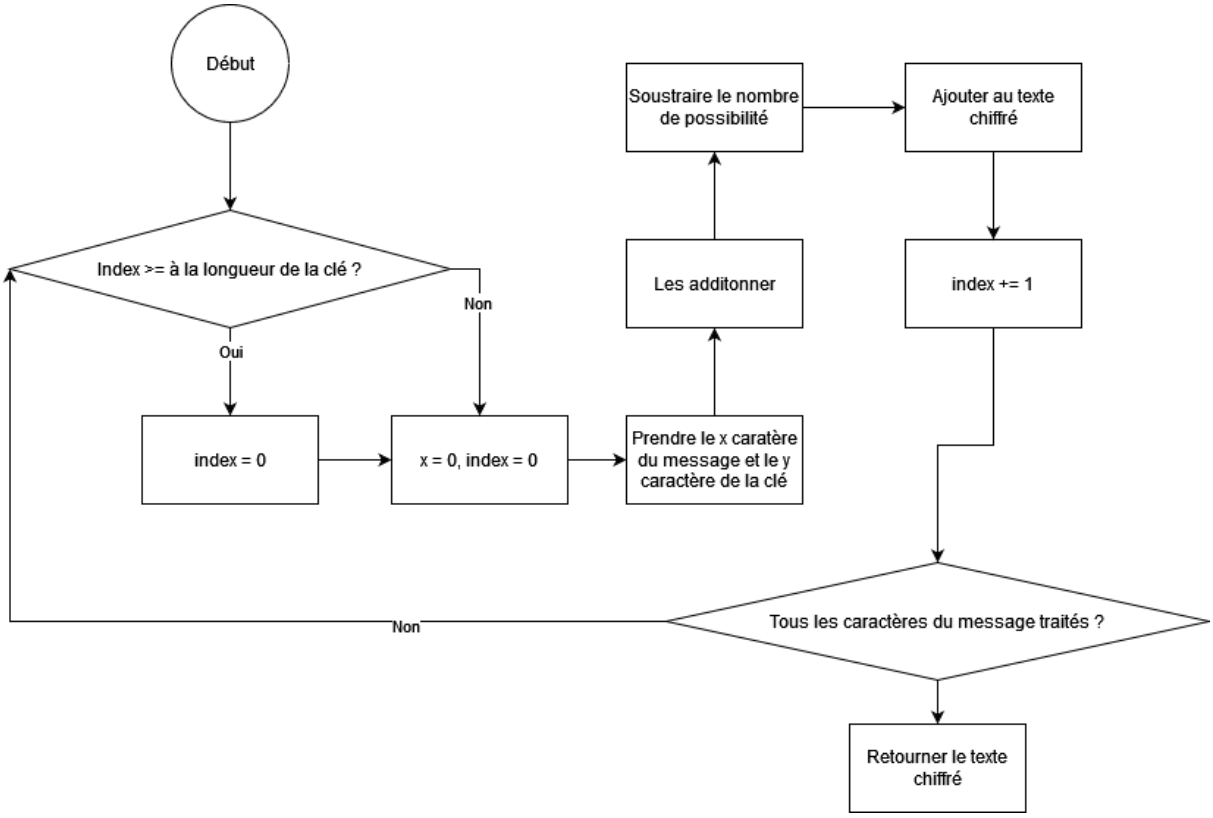

Figure 3 - Schéma du chiffrement avec Vigenère

4 IMPLÉMENTATION

4.1 Structure du code

Le code est structuré sous la forme d'une application WindowsForms, ce qui implique l'utilisation de la programmation orienté objet. WindowsForms à été choisi car je voulais améliorer ma connaissance de cet outil.

4.2 Modules et fonctions principales

4.2.1 Fonction de chiffrement

```
        /// <summary>
        /// Chiffre un texte
        /// </summary>
        /// <param name="text">Le texte à chiffrer</param>
        /// <returns>Un texte chiffré</returns>
        public static string Encrypt(string text)
        {
            string encryptedText = ""; //Le texte chiffré
            int keyIndex = 0; //L'index de la clé

            //Pour chaque charactère du texte
            foreach (char c in text)
            {
                //Si l'index de la clé est plus grand que la
longueur de la clé, on revient au début de la clé
                if (keyIndex >= MasterPassword.Key.Length)
                {
                    keyIndex = 0;
                }

                int characterValue = (int)c; //Le code du
caractère à chiffrer
                characterValue +=
(int)MasterPassword.Key[keyIndex]; //On ajoute le code du
caractère de la clé au code du caratère à chiffrer
                encryptedText += (char)characterValue; //On
ajoute le caractère chiffré au texte chiffré

                keyIndex++; //On incremente l'index de la clé
            }
            return encryptedText; //On retourne le texte chiffré
        }
```

Figure 4 - Fonction de chiffrement

Réflexion pour l'implémentation (dans l'ordre auquel j'y ai pensé):

- Une boucle foreach pour parcourir le message
- Ensuite un index pour parcourir la clé
- Il faut remettre l'index à 0 quand on à atteint le bout de la clé

4.2.2 Fonction de déchiffrement

```
        /// <summary>
        /// Déchiffre un text
        /// </summary>
        /// <param name="text">Le texte à déchiffrer</param>
        /// <returns>Un texte déchiffré</returns>
        public static string Decrypt(string text)
        {
            string decryptedText = ""; //Le texte déchiffré
            int keyIndex = 0; //L'index de la clé

            //Pour chaque charactère du texte
            foreach (char c in text)
            {
                //Si l'index de la clé est plus grand que la
longueur de la clé, on revient au début de la clé
                if (keyIndex >= MasterPassword.Key.Length)
                {
                    keyIndex = 0;
                }

                int characterValue = (int)c; //Le code du
caractère à déchiffrer
                characterValue -=
(int)MasterPassword.Key[keyIndex]; //On soustrait le code du
caractère de la clé au code du caratère à déchiffrer

                decryptedText += (char)characterValue; //On
ajoute le caractère déchiffré au texte déchiffré

                keyIndex++; //On incremente l'index de la clé

            }
            return decryptedText; //On retourne le texte
déchiffré
        }
```

Figure 5 - Fonction de déchiffrement

Pareil que le chiffrement, sauf qu'on soustrait.

4.2.3 Fonction de sauvegarde

```
        /// <summary>
        /// Sauvegarde les entrées
        /// </summary>
        public static void SaveEntries()
        {
            //Crée le dossier password si nécessaire
            if (!Directory.Exists(savePath))
            {
                Directory.CreateDirectory(savePath);
            }

            //Crée le fichier de configuration si nécessaire et y
écrit le Master password chiffré
            if (!File.Exists(configFile))
            {
                File.WriteAllText(configFile, EncryptionMan-
ager.Encrypt(MasterPassword.Key));
            }

            //Rempli la liste de mots de passes
            foreach (Entry entry in PasswordManager.PasswordList)
            {
                File.WriteAllText(Path.Combine(savePath,
entry.Title + fileExtension), JsonSerializer.Serialize(entry));
            }
        }
```

Figure 6 - Fonction de déchiffrement

Réflexion pour l'implémentation (dans l'ordre auquel j'y ai pensé):

- D'abord : écrire la liste de mots de passes avec un foreach
- Ensuite : Créer le dossier et le fichier de config si ils n'existent pas

4.2.4 Fonction de récupération

```
/// <summary>
        /// Récupère les mots de passes du fichier de sauvegardes
        /// </summary>
        /// <param name="inputMasterPassword">Le mot de passe que
l'utilisateur a rentrer</param>
        /// <exception cref="WrongPasswordException">Lance une
exception si le mot de passe n'est pas correct</exception>
        public static void ReadEntries(string inputMasterPassword)
        {
            try
            {
                MasterPassword.Key = inputMasterPassword;

                //Si le fichier de configuration n'est pas
présent, on le crée
                if (!File.Exists(configFile))
                {
                    SaveEntries();
                }
                //Si le mot de passe est correct
                if (EncryptionManager.Encrypt(inputMasterPassword)
== File.ReadAllText(configFile)) ;
                {
                    string[] files = Directory.GetFiles(savePath);
//Récupère les noms des fichiers

                    //Rempli la liste de mots de passes
                    foreach (string file in files)
                    {

PasswordManager.PasswordList.Add(JsonSerializer.Deserialize<Entry>
(File.ReadAllText(file)));
                    }
                    return;
                }
                throw new WrongPasswordException("Mauvais mot de
passe");
            }
            catch (Exception ex)
            {
                //Si le mot de passe n'est pas correct
                if (ex.GetType() == typeof(WrongPasswordExcep-
tion))
                {
                    //On affiche un pop-up
                    string message = "Vous vous êtes trompés de
mot de passe : l'application va s'éteindre";
                    string caption = "Erreur dans le mot de
passe";
                    MessageBoxButtons buttons =
MessageBoxButtons.OK;
                    MessageBox.Show(message, caption, buttons);

                    //On quitte l'application
                    Application.Exit();
                }
            }
        }
```

Figure 7 - Fonction de récupération / Lecture des entrées

Cette méthode est beaucoup plus grande que la sauvegarde, car elle arrête l'application en cas de mot de passe erroné.

Réflexion pour l'implémentation (dans l'ordre auquel j'y ai pensé):

- On récupère la liste des fichiers
- On lit et déchiffre leur contenu et on rempli la liste de mots de passe
- Si le fichier de config n'est pas présent, on le crée
- Si le mot de passe n'est pas correct, on affiche un pop-up et on quitte l'application

5 SÉCURITÉ

Les entrées sont toujours chiffrées sur le disque. Les mots de passes sont cachés à la saisie et à la consultation, avec un bouton pour les copier.

6 TESTS ET VALIDATION

Tests	Résultats	Validé
Entrer un master password erroné	L'application s'arrête	Oui
Ajouter une entrée	L'entrée est ajoutées	Oui
Quittez l'application et se connecter	Les mots de passes sont toujours disponibles	Oui
Supprimer une entrée	L'entrée est supprimée	Oui
Modifier le champ titre d'une entrée	Le titre est mis à jour	Oui
Quittez l'application et se connecter après avoir mis à jour le titre	Le titre est bien à jour	Oui
Modifier une le champ url d'une entrée et quitter l'application avec alt + f4 et se connecter	L'url de l'entrée à bien été modifiée.	Oui

7 JOURNAL DE TRAVAIL

Voir *AlbanSegalen-JDT.xlsm*

8 CONCLUSION

8.1 Bilan des fonctionnalités demandées

Toutes les fonctionnalités du cahier des charges sont présentes dans l'application. Le code pourrait encore être amélioré et optimisé.

8.2 Bilan personnel

Comme la sécurité des données est primordiale de nos jours, c'est un projet nécessaire.
J'ai bien aimé ce projet car il m'a permis de réaliser une application que je pourrais utiliser pour le restant de ma formation.
Ce projet m'a également permis de perfectionner mes compétences en C#.

9 DIVERS

10 ANNEXES

Le code : *P_Secu-114-WinForms*

Le journal de travail : *AlbanSegalen-JDT.xlsm*

L'auto-évaluation : *EVAL_PRATIQUE.xlsx*

Le cahier des charges : *P_Sec-Gestionnaire de mots de passe.pdf*

Le repo GitHub : https://github.com/ASETML/P_Secu-114-Gestionnaire-de-mot-de-passe