Cryptogen User Manual

Zander Labuschagne 23585137
Elnette Möller 235
B.Sc.(Hons) Computer Science & Information Systems 2017

May 2017

Contents

1 Introduction

1.1 Intended Readership

This documented is intended for the users of the Cryptogen application used for the encryption and decryption of messages and files which is developed by two computer science students at the Nort-West University, Zander Labuschagne & Elnette Möller.

1.2 Applicability

This user manual only applies to version 1.x of the Cryptogen application developed by Zander Labuschagne & Elnette Möller.

1.3 Purpose

The purpose of this document is to aid the users of the Cryptogen application so they can use it properly and as intended by the developers. This document hopes to clarify any misunderstandings its users might have.

1.4 Motivation & Background

Cryptogen is developed as part of the Computer Security module for B.Sc. Computer Science & Information Systems Honnors students at the North-West University, Potchefstroom Campus. The goal of this project was to provide students with experience in elementary cryptosystems by developing a small system which implements different encryption algorithms. It was expected of one to two student to work together and develop an application which satisfies the following requirements:

- Handle encryption and decryption of messages and files.
- Graphical interface to simplify the functioning.
- Any programming language(s) may be used.
- The following algorithms had to be implemented:
  - Vigenère Cipher
  - Vernam Cipher

- Columnar Transposition Cipher
- Own implementation

2 User Guide

2.1 Support

2.1.1 Linux

This project was partly developed on a GNU/Linux system, more specifically Manjaro ArchLinux 17.01 KDE x86_64 running Manjaro kernel version 4.11. Linux systems are fully supported and Cryptogen is expected to run flawless on these systems, but is not guaranteed. Any bugs and inconsistencies may be reported to the developers, at least one of the developers aims to maintain this project for UNIX-like systems. A shell script named install.sh is provided which one can run to install the application on a Linux system. The installation adds an application entry to the applications menu or better known as a desktop entry for the application. Since a Java .jar file is provided one can run the .jar executable on Linux systems if Java Runtime Environment 1.8 or higher is installed, **very important: OpenJDK will not suffice!** but it is more convenient to do the installation and to use the application as intended. One can run the install with the terminal command "sudo sh install.sh", if a permission error is encountered first enter the following command before installation: "sudo chmod +x install.sh", this will add executable permission to the install script. To be able to read this document a PDF reader is required such as Okular.

2.1.2 Windows

This project was partly developed on a Windows platform, thus some Windows support is provided such as a GUI specifically adapted for Windows. Many problems or inconsistencies were experienced by the developers when the application was used on a Windows platform such as malfunctioning drag ń drop, so there is no guarantee that the application will function propoly on Windows systems. It is very complicated to create an installer or portable .exe for Windows from a .jar executable and the little third party software that exists to make this process simpler is very expensive hence no Windows executable or intsaller. There is no guaratee that the developers will address any bugs or problems experienced on Windows systems but feedback is welcome just to keep record of what functions well. Since a Java .jar file is provided one can run the .jar executable on Windows systems if Java Runtime Environment 1.8 or higher is installed. To be able to read this document a PDF reader is required such as Adobe Reader.

2.1.3 macOS

This application was not yet tested on a macOS system and since the lack of a macOS system from the developers no dedicated installer or macOS executable(.app or .dmg) was made. However at least one of the two developers aims add support for macOS systems in the future. Since a Java .jar file is provided one can run the .jar executable on a macOS system if Java Runtime Environment 1.8 or higher is installed. To be able to read this document a PDF reader is required such as Skim.

2.2 Features Overview

2.2.1 Cryptosystems

This application currently provides Vigenère, Vernam, columnar transposition ciphers and our own implementation called the Elephant Cipher. Support for more cryptosystems in the future is planned. One can select a cryptosystem to use by choosing one of the options on the top left area, the dot represents the selected cryptosystem.

The Vigenère cipher only encrypts and decrypts english alphabetical characters, and leaves all special characters in their original form since the Vigenère cipher was designed for text or message encryption only. The key used for Vigenère enryption and decryption may only consist of alphabetical characters as seen in *figures 1* and *2*, violating this would produce an error.

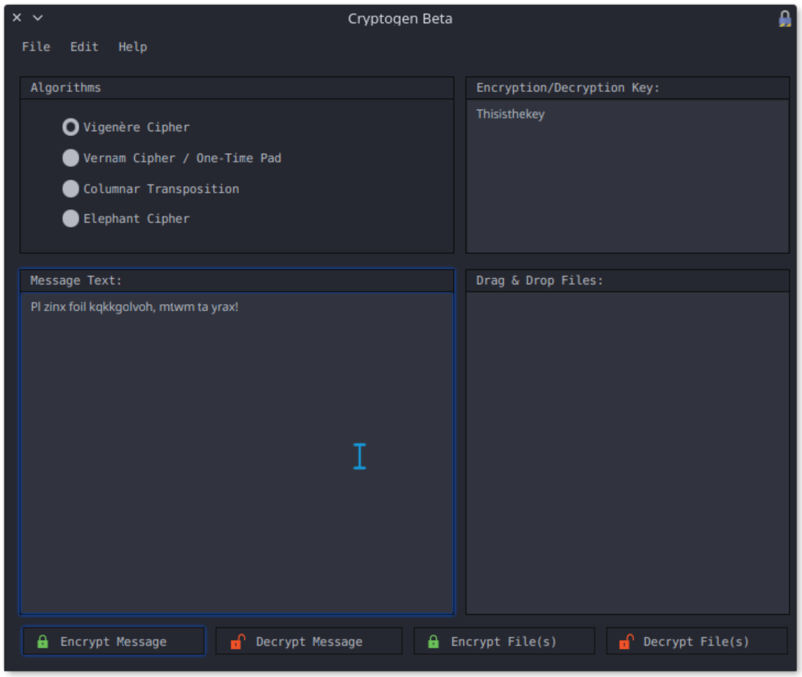
The Vernam cipher accepts a key containing any characters unlike the Vigenère cipher, this means that the cipher text will also conatin any other characters as well. The Vernam cipher encrypts and decrypts all characters.

The columnar transposition cipher can take a key containg any characters as well and encrypts and decrypts all charcters just like the Vernam cipher does.
The Elephant cipher is a self implemented cipher designed by the developers of this application, Zander Labuschagne & Elnette Möller. This cipher can not be considered as a sound cryptosystem because it is based on simple cryptosystems of which one is considerd to be unsound in modern

days.

2.2.2 Message Encryption & Decryption

Encryption and decryption of messages can be done by using the text area. The user may choose a cryptosystem in the box on the top left corner followed by entering a plain text message into the text area and encrypt it with any of the provided encryption algorithms, see *figure 1* for an example of encryption of a message by the Vigenère cipher, this results in an encrypted message provided in the same text area after the "Encrypt Message" button has been clicked. The user may also enter an encrypted message in the text area and click on the "Decrypt Message" button to decrypt the encrypted message, the result which is the decrypted message will be displayed in the same text area. See *figure 2* for an example of message decryption by using the Vigenȓe cipher.

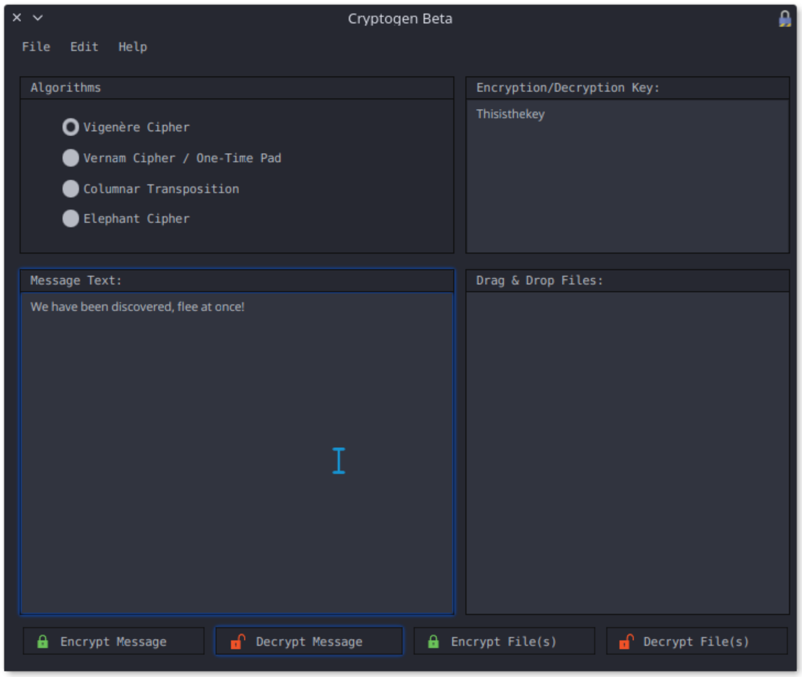


Algorithms
[✓ Vigenère Cipher]
Vernam Cipher / One-Time Pad
Columnar Transposition
Elephant Cipher

Encryption/Decryption Key:
Thisisthekey

Message Text:
Pl zinx foil kqkkgolvoh, mtwm ta yrax!

Figure 1: Vigenère message encryption



Thisisthekey

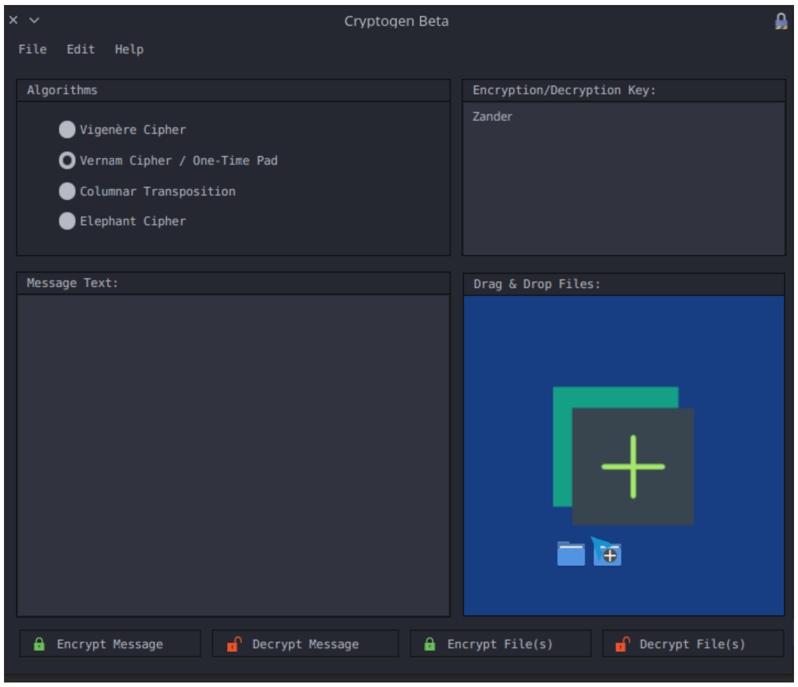
We have been discovered, flee at once!

Figure 2: Vigenère message decryption

2.2.3 File(s) Encryption & Decryption

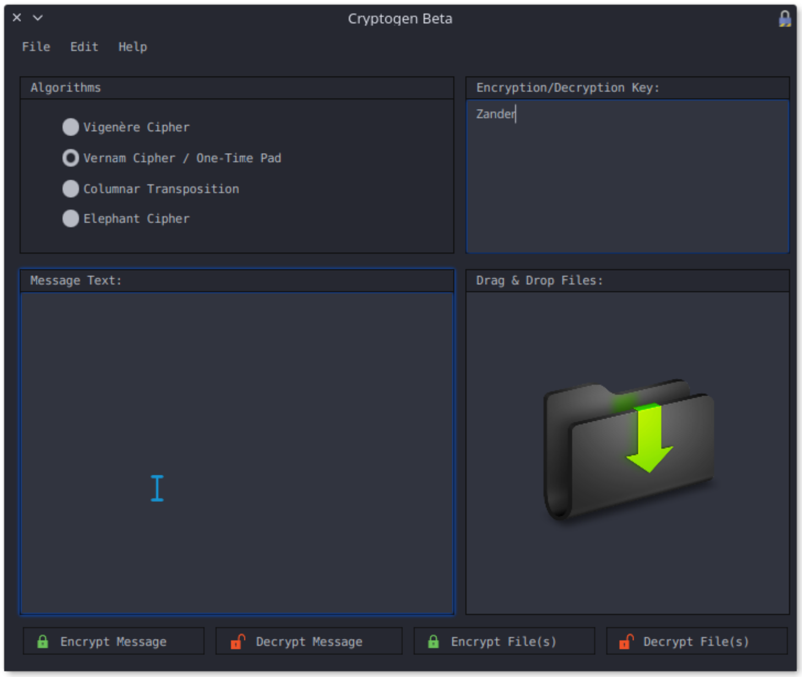
All sorts of video, image, audio and document files may also be encrypted and decrypted(.jpeg, .pdf, .png, .mp4, .flv, .mp4, .docx, .xlsx etc), there are two ways of doing such, the first is to drag and drop the file(s) into the area on the lower right, as seen in *figure 3*, once they are dropped they are ready to be encrypted or decrypted as seen in *figure 4*. The other way is to select some files or file in your favorite file explorer, on Windows it's Windows Explorer, macOS it's Finder and Linux with KDE it can be Dolphin, and to right click and click on copy or just CTRL+C followed by clicking the File button on the menu inside Cryptogen and choosing "Paste Files". After some files are attached, a cryptosystem of choice is chosen and a key is entered in the Key text area on the upper right as in *figure 4*, they may be encrypted or decrypted by clicking on the "Encrypt File(s)" button or the "Decrypt File(s)" button respectively. While files are being encrypted or decrypted a progress bar for each file will be displayed, one can move them to different places in the screen to see all progress simulataneously. The size limit of the files depends on the amount of RAM available.

One can attach one file, multiple files or even multiple folders containing files and more folders, this will encrypt/decrypt all the files inside all the directories, so be careful. When files are encrypted they get the extension ".cg" to differentiate them from the normal or decrypted files, the original files are also deleted in the process. When files are decrypted only the encrypted files with the extension ".cg" will be selected for decryption, all encrypted files are deleted while the original files are recreated from their encrypted counterparts. Becareful with the total file size, Cryptogen currently puts all files in RAM, so large files may fill up your RAM and may not get encrypted or decrypted, an alternative solution is planned for a future release. To remove attached files one can click on "File" followed by clicking on "Clear Files".



Zander

Figure 3: Files being dragged into drop area



Encryption/Decryption Key: Zander

Figure 4: Files attached for encryption or decryption

2.2.4 Additional Features

The "Help" menu provides a button to open this document for help as well as an "About" form which displays information about the application as well as the developers. The "Look and Feel" menu provides additional look and feels or themes as some might call them, some people have different tastes than others which is why a light look and feel called "Midna" is included and two dark look and feels called "Midna Dark" and "Breath".

2.2.5 Technical Features

The main programming langauage used is Java with Oracle Java Development Kit version 1.8. A JavaFX framework is used to display the graphical user interface, it includes .fxml files containing the graphical user interface layout, .css files containing information on how the graphical user interface looks and .java files containing code to execute instructions in order to provide functionality into the application. A feature not seen directly is multithreading, when encrypting or decrypting files, each file is encryptes or decrypted on its own thread meaning multiple files can be encrypted or decrypted simultaneously, one can see a progress bar for each file or thread, Appendix B is an example of multithreading. Some might not call this a feature but open source gives many advantages over closed source, this project is open source and available on GitHub, any feedback, criticism or pull requests are welcome.

3 Cryptosystems Overview

This section explains how each cipher is implemented in the application.

3.1 Vigenr̀e Cipher

The Vigenère encryption for text and messages works as follows [2]:
First the key is converted to uppercase letters, then the following encryption function is applied to the key and only alphabetical characters in the message.
$c = E(m, k) = m - 65 + k \; mod \; 26 + 65 \; \forall$ upper case letters
$c = E(m, k) = (m - 97 + k \; mod \; 26 + 32 - 97) \; mod \; 26 + 97 \; \forall$ lower case letters

The Vigenère decryption for text and messages works as follows:
First the key is converted to uppercase letters just as in encryption, then the following decryption function is applied to the key and only alphabetical characters in the message.
$m = D(c, k) = (c - 65 - k + 65) \; mod \; 26 + 65 \; \forall$ upper case letters.
$m = D(c, k) = (c - 65 - k + 65) \; mod \; 26 + 65 + 26 \; \forall$ upper case letters and if a negative value would have been obtained(less than A).
$m = D(c, k) = (c - 97 - (k + 32 - 97)) \; mod \; 26 + 97) \; \forall$ lower case letters.
$m = D(c, k) = (c - 97 - (k + 32 - 97)) \; mod \; 26 + 97 + 26) \; \forall$ lower case letters and if a negative value would have been obtained(less than a).

The Vigenère encryption for files works as follows:
The key is never converted to uppercase letters as in the encryption for text, then the following encryption function is applied to the key and all bytes contained in the set of files.
$c = E(m, k) = m + k \; \forall$ bytes contained in files.

The Vigenère decryption for files works as follows:
Just as in the encryption for files the key is never converted to uppercase, then the following decryption function is applied to the key and all bytes contained in the set of files.

$m = D(c, k) = c - k \; \forall$ bytes contained in files.

3.2 Vernam Cipher

Gilbert Vernam invented the onte time pad or Vernam cipher which xor a stream of bits with another to produce an ireversable stream of bits when the key is not known [1]. The Vernam encryption and decryption for files works as follows[4]:

$c = E(m, k) = m \oplus k$

$m = D(c, k) = c \oplus k$

Thus if a file is encrypted twice, the original file will be reproduced. However the Vernam encryption for text works very differently, it is still an xor but modified so that it never enters the non readable character zones.

3.3 Columnar Transposition Cipher

This cipher takes a stream of either characters or bytes of a file, depending what type of data is being encrypted, and put it in a matrix of characters or bytes with a width equal to the length of the key provided. The matrix is then transposed to get the cipher text or files. The decryption of messages and files works in the same fashion[3].

3.4 Elephant Cipher

This cipher takes plain text or data and perform an xor operation with the key followed by rotating the result as many times as the length of the key. The rotation operation takes the last character or byte and place it in front. The exact inverse is performed for the decryption part, The cipher text or bytes are rotated in the inverse direction followe by a xor operation.

4 Acknowledgements

This project was developed and is being maintained with JetBrains IntelliJ IDEA Ultimate edition with a JetBrains student license. This documentation was compiled with LaTeX. The icons on the "Encrypt" and "Decrypt" buttons are made by Maxim Basinski from http://www.flaticon.com at http://www.flaticon.com/authors/maxim-basinski and is licensed by Creative Commons BY 3.0 at http://creativecommons.org/licenses/by/3.0/. All other icons are made by Madebyoliver

from http://www.flaticon.com at http://www.flaticon.com/authors/madebyoliver and is licensed by Creative Commons BY 3.0. at http://creativecommons.org/licenses/by/3.0/. The default look and feel of Cryptogen is designed after the MidnaDark look and feel for KDE's Plasma theme which is the dark theme used and designed by KaOS, a very beautiful KDE Linux distribution. E-Mail of author, Anke Boersma: demm@kaosx.us KaOS Website: https://kaosx.us. The Midna look and feel is also designed by KaOS and is the default look and feel for KaOS. The Breath look and feel is also a look and feel for KDE's Plasma theme designed by Manjaro, a Linux distribution based on Arch Linux which is very lightweight and embraces simplicity. GitHub is used to do version control over this project since it is easily integratable with JetBrains IntelliJ.

5 License

6 Reflection

This section contains personal experiences and views from the developer's point of view while the project commenced.

6.1 Developer #1: Zander Labuschagne

I have found this project somewhat daunting, I though that the file encryption and decryption would be the hardest to implement but in fact it was the messages because you have to place limits on the message in order to stay within readable characters. The Vigenère and Vernam ciphers was extremely easy to implement on files, and yet everyone else struggled with those ciphers, it might be because of my previous experience with cryptography in my second and third year at university of which the others lacked of. This is my third JavaFX application an everytime I use it I make sure I learn something new, during this project I had my second experience with multithreaded development, although it was not planned to be this soon. I have gained more experience with CSS design and especially file handling.

I would like to improve this application by not keeping all files in the RAM so it would be safer and more reliable to encrypt and decrypt multiple large files. I would also like to add an overall progress status indicator which displays the progress of all files and not just the progress of one independantly.

6.2 Developer #2: Elnette Möller

References

[1] Steven M Bellovin. Frank miller: Inventor of the one-time pad. *Cryptologia*, 35(3):203–222, 2011.

[2] Aiden A Bruen and Mario A Forcinito. *Cryptography, information theory, and error-correction: a handbook for the 21st century*, volume 68. John Wiley & Sons, 2011.

[3] Charles P Pfleeger and Shari Lawrence Pfleeger. *Security in computing.* Prentice Hall Professional Technical Reference, 2002.

[4] Gilbert S Vernam. Secret signaling system, July 22 1919. US Patent 1,310,719.

Appendix A: Vigenère Message Decryption Algorithm

```
/**
 * Decryption method to decrypt Vigenère encrypted files
 * @param cipherFile encrypted file to decrypt
 * @param key key used to decrypt the file
 * @return the decrypted original file
 */
public static void decrypt(File cipherFile, char[] key)
{
try
{
Path path = Paths.get(cipherFile.getAbsolutePath());
final byte[] cipherData = Files.readAllBytes(path);
byte[] plainData = new byte[cipherData.length];

Progress progress = new Progress("Decryption");
// In real life this task would do something useful and return
// some meaningful result:
Task<byte[]> task = new Task<byte[]>()
{
@Override
public byte[] call() throws InterruptedException
{
for(int iv = 0; iv < cipherData.length; iv++)
{
updateProgress(iv, cipherData.length);
plainData[iv] = (byte) ((int) cipherData[iv] - (int) key[iv % key.length]);
}

return plainData;
}
```

```
};
// binds progress of progress bars to progress of task:
progress.activateProgressBar(task);

// in real life this method would get the result of the task
// and update the UI based on its value:
task.setOnSucceeded(event ->
{
try
{
progress.getDialogStage().close();
FileOutputStream fos = new FileOutputStream(cipherFile.getAbsolutePath().substring(0, cipherFi
fos.write(task.getValue());
fos.close();
cipherFile.delete();
}
catch (FileNotFoundException ex)
{
ex.printStackTrace();
Cryptography.handleException(ex);
}
catch(IOException ex)
{
ex.printStackTrace();
Cryptography.handleException(ex);
}
catch (Exception ex)
{
ex.printStackTrace();
Cryptography.handleException(ex);
}
});
progress.getDialogStage().show();

Thread thread = new Thread(task);
thread.setDaemon(true);
thread.start();
}
catch (IOException ex)
{
ex.printStackTrace();
Cryptography.handleException(ex);
}
catch (Exception ex)
{
ex.printStackTrace();
Cryptography.handleException(ex);
}
}
```

Appendix B: Vernam File Encryption Algorithm

```
/**
 * Encryption method used to encrypt files with the Vernam cipher
 * @param plainFile file about to be encrypted
 * @param key used to encrypt the file
 * @return the encrypted file
 */
public static void encrypt(File plainFile, char[] key)
{
try
{
```

```
Path path = Paths.get(plainFile.getAbsolutePath());
final byte[] plainData = Files.readAllBytes(path);
byte[] cipherData = new byte[plainData.length];

Progress progress = new Progress("Encryption");
// In real life this task would do something useful and return
// some meaningful result:
Task<byte[]> task = new Task<byte[]>()
{
@Override
public byte[] call() throws InterruptedException
{
for (int vi = 0; vi < plainData.length; vi++)
{
updateProgress(vi, plainData.length);
cipherData[vi] = (byte) ((int) plainData[vi] ^ (int) key[vi % key.length]);
}

return cipherData;
}
};
// binds progress of progress bars to progress of task:
progress.activateProgressBar(task);

// in real life this method would get the result of the task
// and update the UI based on its value:
task.setOnSucceeded(event ->
{
try
{
progress.getDialogStage().close();
FileOutputStream fos = new FileOutputStream(plainFile.getAbsoluteFile() + ".cg");
fos.write(task.getValue());
fos.close();
plainFile.delete();
}
catch (FileNotFoundException ex)
{
ex.printStackTrace();
Cryptography.handleException(ex);
}
catch (IOException ex)
{
ex.printStackTrace();
Cryptography.handleException(ex);
}
catch (Exception ex)
{
ex.printStackTrace();
Cryptography.handleException(ex);
}
});

progress.getDialogStage().show();

Thread thread = new Thread(task);
thread.setDaemon(true);
thread.start();
}
catch (IOException ex)
{
ex.printStackTrace();
Cryptography.handleException(ex);
}
```

```
catch (Exception ex)
{
ex.printStackTrace();
Cryptography.handleException(ex);
}
}
```

Appendix C: Encryption of Multiple Directories

```
/**
     * Handle encryption of multiple files and directories
     * @param files List of files to be encrypted
     */
    private void encryptFiles(List<File> files)
    {
        try
        {
            char[] key = txtKey.getText().toCharArray();
            for (int ii = 0; ii < files.size(); ii++)//Encrypt Each File
            {
                if (files.get(ii).isDirectory())
                    encryptFiles(Arrays.asList(files.get(ii).listFiles()));
                else if (files.get(ii).isFile())
                {
                    if (radVigenere.isSelected())
                    {
                        Cryptography.VigenereCipher.encrypt(files.get(ii), key);
                        method = "Vigenère cipher.";
                    }
                    else if (radVernam.isSelected())
                    {
                        Cryptography.VernamCipher.encrypt(files.get(ii), key);
                        method = "Vernam cipher.";
                    }
                    else if (radColumnarTrans.isSelected())
                    {
                        Cryptography.ColumnarTranspositionCipher.encrypt(files.get(ii), key);
                        method = "columnar transposition.";
                    }
                    else if (radElephant.isSelected())
                    {
                        Cryptography.ElephantCipher.encrypt(files.get(ii), key);
                        method = "Elephant Encryption.";
                    }
                }
            }
        }
        catch (Exception ex)
        {
            ex.printStackTrace();
            handleException(ex);
        }
    }
```

Appendix D: Drag and Drop Feature

```
/**
     * Event necessary to execute before DragDropped may be executed
     * Allows DragDropped to receive files by Copy or Move
     * @param event
     */
    @FXML
    protected void onDragOver(DragEvent event)
    {
        //data is dragged over the target
        if(event.getGestureSource() != stackPane && event.getDragboard().hasString())
            event.acceptTransferModes(TransferMode.COPY_OR_MOVE);
        event.consume();
    }

    /**
     * Method to execute when file is dragged over area
     * @param event
     * @throws IOException
     */
    @FXML
    protected void onDragEntered(DragEvent event)
    {
        pneFilePane.getStyleClass().remove("pneDefault");
        pneFilePane.getStyleClass().remove("pneFilePaneError");
        pneFilePane.getStyleClass().add("pneFilePaneDrag");
    }

    /**
     * Method to execute when file is no longer above area
     * @param event
     */
    @FXML
    protected void onDragExited(DragEvent event)
    {
        pneFilePane.getStyleClass().remove("pneFilePaneDrag");
        pneFilePane.getStyleClass().add("pneDefault");
    }

    /**
     * Method to execute when file(s) is/are dropped in area
```

```
 * @param event
 * @throws IOException
 */
@FXML
protected void onDragDropped(final DragEvent event) throws IOException
{
    final Dragboard db = event.getDragboard();
    boolean success = false;
    if (db.hasFiles())
    {
        success = true;
        // Only get the first file from the list
        files = db.getFiles();
        Platform.runLater(new Runnable()
        {
            @Override
            public void run()
            {
                try
                {
                    /*for(int i = 0; i < files.size(); i++)
                        System.out.println(files.get(i).getAbsolutePath());*/

                    if(!stackPane.getChildren().isEmpty())
                    {
                        stackPane.getChildren().remove(0);
                    }
                    pneFilePane.getStyleClass().remove("pneFilePaneDrag");
                    pneFilePane.getStyleClass().remove("pneDefault");
                    pneFilePane.getStyleClass().add("pneFilePaneDropped");
                }
                catch (Exception ex)
```

```
                    {
                        System.out.println(ex.toString());
                    }
                }
            });
        }
        event.setDropCompleted(success);
        event.consume();
    }
```