Viscrypt - An Image Encryption Software

A Project report submitted in fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Yash Mehrotra (171340)

Under the supervision of

Dr. Kapil Sharma

To

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234,

Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Viscrypt"** in fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2021 to May 2021 under the supervision of Dr. Kapil Sharma(Assistant Professor , Computer science department).
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

[illegible]

(Student's Signature)
Yash Mehrotra
171340

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

[illegible]

(Supervisor's Signature)
Dr. Kapil Sharma
Assistant Professor
Computer Science Department
Dated:

Acknowledgement

I have put a lot of effort into this project, but this was not possible without the kind support and help of many people. I must thank all of them sincerely. I am very grateful to our undertaking chief Dr. Kapil Sharma for providing leadership and continuous supervision as well as the necessary information about my project and for supporting the completion of this undertaking. I would like to give special thanks and appreciation to our friends and colleagues who have given us time and attention.

Thank you.

Date: May 18th, 2021

Yash Mehrotra (171340)

Contents

List of Figures

Abstract

Cryptography is a practice that is thousands of years old used to protect secrets. Until recent decades this practice could be called classic cryptography, as it dealt with pro tecting important documents and government secrets. But in the 1960s, we moved to a new form of cryptography called Modern/Digital cryptography, which deals with pr otecting secret information on a computer. Such an encryption is achieved by using a lgorithms that have a key to encrypt and decrypt data.

DES(Data Encryption Standard) was an early US government approved algorithm de veloped by IBM, which was used for encryption. This algorithm used asymmetric ke y ciphers(also called public ciphers). However, the quality of encryption must also in crease with the improvement of technology. Around 1990s, the use of Internet for co mmercial transactions called for a widespread standard of encryption. The DES(Data Encryption Standard) was deemed insecure in the 1990s due to its relatively small 56 -bit key size.

DES was eventually replaced by AES(Advanced Encryption Standard) in 2001. AES was developed Vincent Rijmen and Joan Daemen, so it was also known as Rijndael a lgorithm. This project implements AES algorithm to develop an encryption software that can encrypt images into a desired number of key images that can be decrypted to gether to get the original image. The software developed through the course of this pr oject is named Viscrypt which is short for Visual Cryptography.

CHAPTER 1) INTRODUCTION

1.1 Problem Statement and the presented Solution

1.1.1 Understanding DES

Data Encryption Standard (DES) was created in the early 1970s by IBM and adopted by the National Institute of Standards and Technology (NIST). It is a symmetric blo ck key cipher. This algorithm takes plain text in 64bit blocks and converts it into ci pher text using 56-bit keys.

Because it’s a symmetric key algorithm, same key is employed in both encryption an d decryption of the data. Asymmetrical algorithms use two different keys for encryp tion and decryption of data.

Cryptographer Horst Feistel developed the Feistal cipher, also known as LUCIFER, in 1971. DES is based on the LUCIFER cipher itself. DES employed 16 rounds of t he Feistel structure, and used a different key for each of the rounds.

In November 1976, DES became the federal approved encryption standard. The alg orithm was reaffirmed as the standard throughout the 70s, 80s, and the 90s. DES was the encryption standard in information security throughout this time.

In 2002, a public competition was held to find a replacement for DES. Subsequently , Advanced Encryption Standard (AES) replaced the DES encryption algorithm as th e accepted standard. DES’s 1999 reaffirmation was officially withdrew by NIST in May 2005. However, Triple DES also known as 3DES remains approved through 2 030 for sensitive government information.

3DES is also a symmetric block key cipher that applied the DES cipher, but using t hree keys. Triple DES does encryption with first key, decryption with second key, a nd again encryption with third key.

DES algorithm had to be replaced because of its 56 bit key lengths being too small, in consideration with the increased processing power of newer hardware. Since key length influences encryption strength, DES’s 56 bit key was no longer good enough to handle the advances in technology and computing.

It should be noted however, that Triple DES is still used today, although it is consid ered a legacy encryption algorithm. From 2024 onward, NIST plans to disallow all f orms of 3DES as well. AES is now the standard of encryption.

1.1.2 Why DES was Replaced

A series of public challenges took place to show the DES was no more adequate and should be replaced. Electronic Frontier Foundation(EFF) and distributed.net were tw o organizations who were important in proving that DES is outdated.

The first contest, was held in 1997, and took 84 days to break the encrypted message using a brute force attack.

There were two challenges held in 1998. It took a little over a month to decrypt the first challenge's text which was: "The unknown message is: Many hands make light work". And it took lesser than 3 days to break the second challenge's text which was :"It's time for those 128-, 192-, and 256-bit keys".

The third and final challenge, took place in early 1999. It only lasted 22 hours and 15 minutes until it was broken by EFF's Deep Crack computer and distributed.net's co mputing network. The Deep Crack computer costed around $250K. The decrypted te xt was: "See you in Rome (Second AES Candidate Conference, March 22-23, 1999)".

Even 3DES was proven inefficient against brute force attacks. On July 19th, 2018, NI ST published a draft guidance informing that 3DES is going to be officially retired in 2024.

1.1.3 What is AES algorithm?

Currently, the most popular and widely used symmetric encryption algorithm is the
Advanced Encryption Standard (AES). It is supposed to be almost six times faster t
han triple DES.

The AES algorithm was developed by Vincent Rijmen and Joan Daemen. For this r
eason it is sometimes also called the Rijndael algorithm. It is a symmetrical block ci
pher algorithm that takes plain text in blocks of 128 bits and converts them to ciphe
rtext using keys of 128, 192, and 256 bits. AES algorithm is the current worldwide s
tandard because it is considered secure.

Some features of AES algorithm are −

1. Symmetric Block Cipher and Symmetric Key

2. 128/192/256-bit keys

3. Six times faster than Triple-DES

4. It provides complete specification and design detail

5. Software is implementable in languages such as C and Java

The table given below shows the comparison between DES and AES;

	DES	AES
Developed in	1977	2000
Type of Cipher	Symmetric Block Cipher	Symmetric Block Cipher
Size of Block	64-bits	128-bits
Length of Key	56-bits	128/192/256-bits

1.1.4 AES Algorithm Steps

SP network, also known as substitution permutation network is employed by AES al gorithm to produce a ciphertext over multiple rounds. The number of these rounds d epend on the size of the key that is being used. For example, a 128bit key will use 1 0 rounds, a 192bit key size will use 12 rounds, and 256bit key size will use 14 round s. One key is inputted into the algorithm. This key is used to get round keys for every round, which are required for the next round.

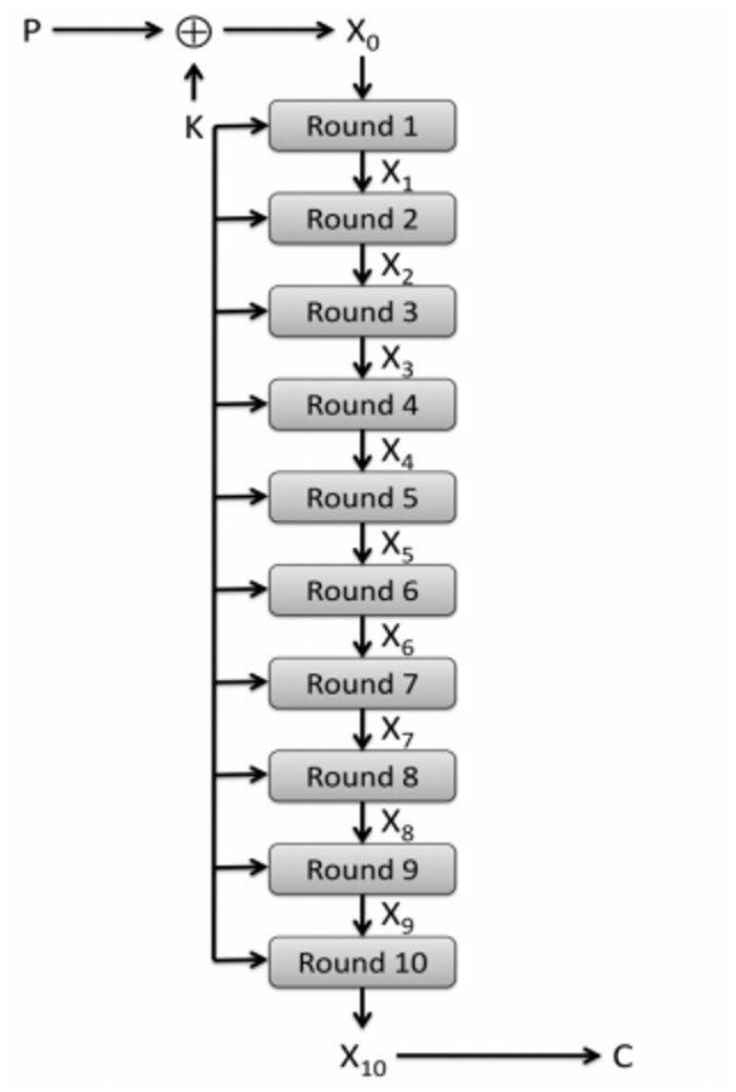


Fig 1.1 AES Rounds for 128-bit

Every round in the algorithm is divided into four steps.

1.1.4.1 Byte Substitution

In the very first step, the block text bytes have to be substituted based on rules dictat ed by predefined S-boxes. S-boxes are also called substitution boxes.

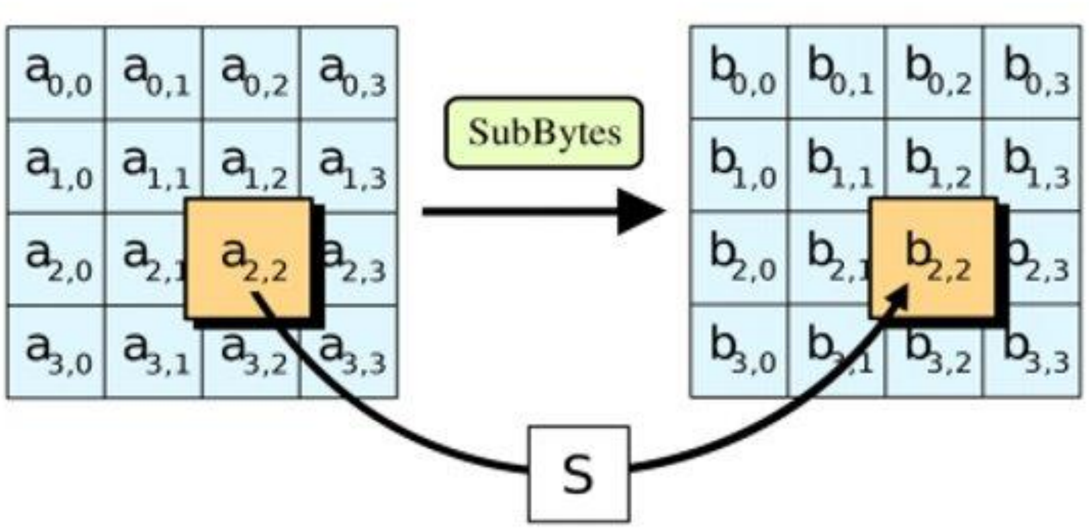


Fig 1.2 AES Algorithm Step 1

1.1.4.2 Row Shifting

Permutation is the next step. In this step, all rows except the first are shifted by one.

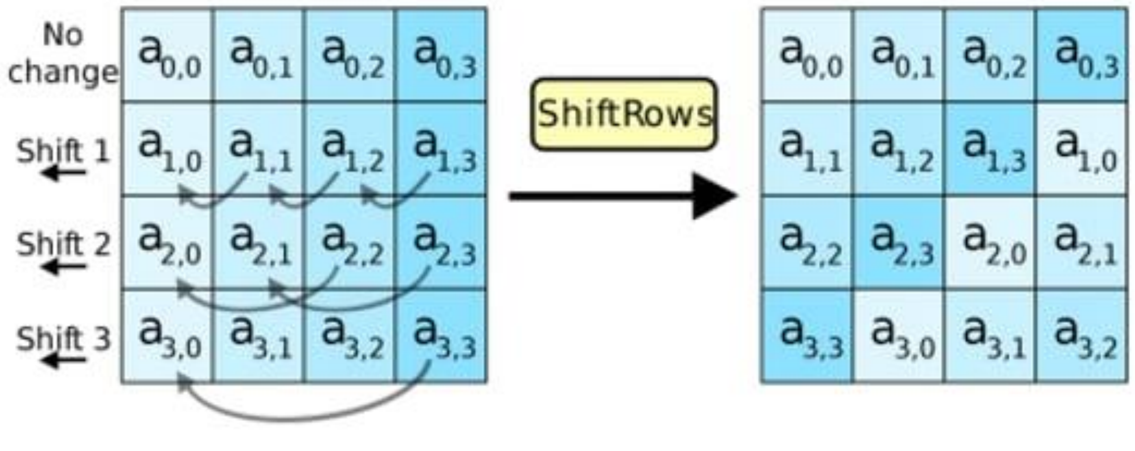
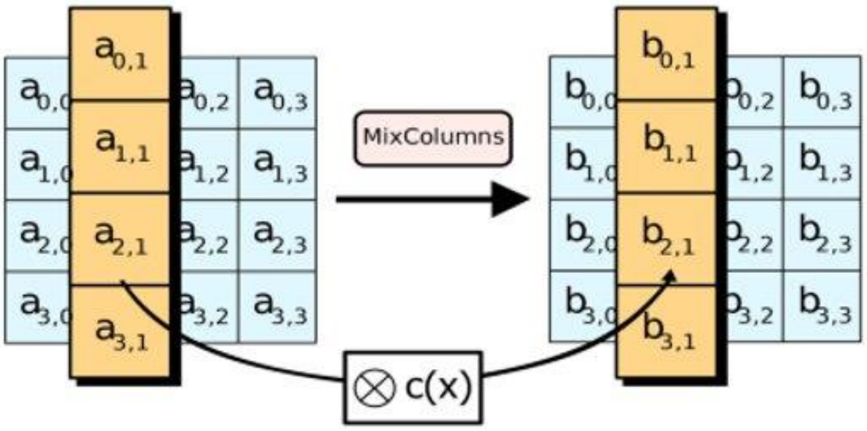


Fig 1.3 AES Algorithm Step 2

1.1.4.3 Column Mixing

In the third step, a Hill cipher is used to jumble up the message furthermore by mixi ng the block’s columns.

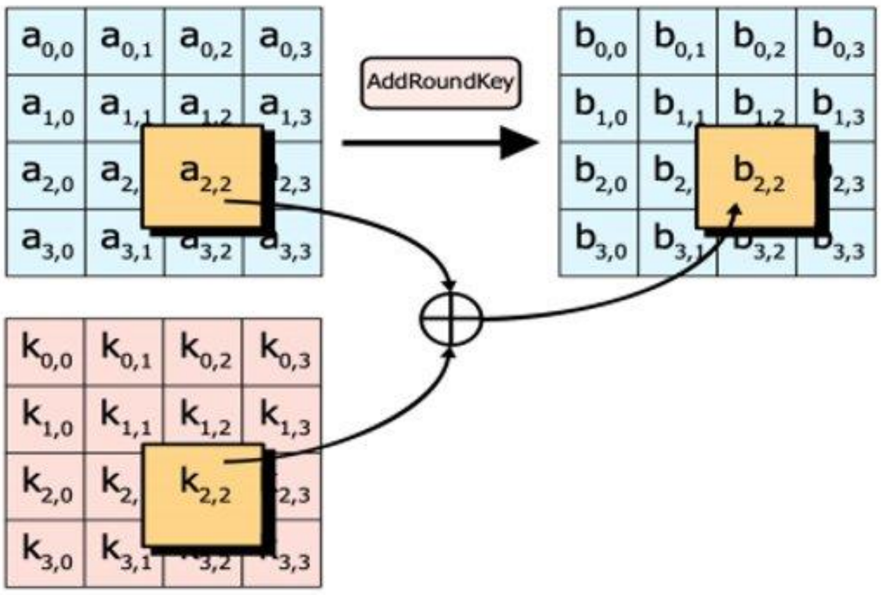


MixColumns
$\otimes c(x)$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

Fig 1.3 AES Algorithm Step 3

1.1.4.4 Round-key Adding

In the final step, the message is XORed with the respective round key.



AddRoundKey
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

Fig 1.4 AES Algorithm Step 4

These steps are done repeatedly to ensure that the final cipher text is secure.

1.1.5 Hill Cipher

The Hill cipher is a polygraphic substitution cipher based on Linear Algebra. It is a v ery mathematical cipher compared to other cipher because it uses modulo arithmetic, matrix multiplication, and matrix inverses. The Hill cipher can work on arbitrary size d blocks because it is a block cipher.

To understand how Hill cipher works, let us encrypt and later decrypt an example tex t: “CODE”. Hill cipher gets more complex with size of key matrix, so, for simplicity, we shall use a straightforward substitution scheme. In this scheme the letter A is mar ked 0, B is mapped 1, and so on.

1.1.5.1 Encryption

Following operation will be used to encrypt the Hill cipher;

E(K, P) = (K*P) mod 26

In the above formula,

K is key matrix.

P is plaintext in vector form.

Matrix multiplication of K and P will give us the encrypted ciphertext.

Let's do so step by step:

1. Pick any random keyword to encrypt the plaintext message. For this example, let’s take the keyword; “DCDF”. Use the substitution scheme to convert this keyword int o a numerical 2x2 key matrix.

2. For the next step we have to convert the plaintext message into vector form. The v ector needs to be 2x1, since the key matrix is 2x2. The message; “CODE” is four let ters long so we it needs to be split it into blocks of two and then substituted to get the 2x1 plaintext vectors.

3. Finally we must perform matrix multiplication between the key matrix and each 2 x1 plaintext vector. Take the moduli of the resulting 2x1 vectors by 26, and concate nate the results to get the final ciphertext. In this example; “WWVA”.

1.1.5.2 Decryption

Following operation is used in order to decrypt a Hill cipher;

D(K, C) = (K-1 *C) mod 26

In the given formula,

K is the key matrix.

C is the ciphertext in vector form.

Matrix multiplication of the inverse of the key matrix with the ciphertext will give u s the decrypted key.

Let's do this step by step with our ciphertext, "WWVA":

1. For the first step, we must calculate the inverse of the key matrix. We must use m odulo 26 to keep the result between 25. We can use the Extended Euclidean algorith m to find the modular multiplicative inverse of the key matrix determinant.

2. For the second and last step, we must perform multiplication between 2x1 blocks o f the ciphertext and the inverse of the key matrix to get our original plaintext messag e. For this example the original plaintext was: “CODE”.

1.2 What is Viscrypt?

Viscrypt is an application that was developed to showcase the learning done throughout this project. Viscrypt is short for Visual Cryptography. As such, the application takes an image, and encrypts is using AES algorithm to produce multiple cipher images, which can be used to decrypt the original image. Viscrypt was designed to make visual cryptography simpler. It takes an image in the format of jpg, bmp, or png. It uses the input image to create byte grey image which consists of two colors- black and white. Afterwards it generates cipher images with dots on it like noise. These images can be decrypted by using Photoshop with exclusion option to get the original grey image back.

1.3 Objectives

While the broader objective was to develop a simple application to display the learning of cryptography, doing so required a series smaller objectives to deliver a finished application.

For this I prepared a Gant Chart to help myself through the complete process:

Task	January	February	March	April	May
Planning	■				
Research		■	■		
Design			■	■	
Implementation			■	■	■
Documentation					■

Fig 1.5 Gant Chart

This chart displays time spent and managed throughout the semester to achieve the completion of the project. But in terms of development of Viscrypt, I have detailed below what the objectives were to finish the application itself.

1.3.1 Viscrypt UI/UX Objectives Checklist

Following were the User Experience objectives that had to be met to develop Viscryp t:

1. Implementing image import option. The importer must use file explorer to select a n image that is of jpg, png, or bmp format only. The name and path of the image sho uld be displayed for clarity.

2. A threshold meter that defines the amount of noise in each cipher image. Has the s ame feature as having 128/192/256-bit keys in AES.

3. Dropdown menu to select the number of cipher images to be created.

4. Exporter that lets the user select the destination of cipher images using file explore r.

5. A console that displays the operations taking place to encrypt the selected image.

6. A button to start encryption process

1.3.1 Viscrypt Technical Objectives Checklist

1. The importer shouldn’t import any other file than jpg, png or bmp extension.

2. The threshold slider should make the encryption denser or sparser.

3. Multiple images should be able to be generated but need to work in tandem to be d ecrypted into a single image.

4. Saving the cipher images on user selected location on the hard drive.

5. The console should also mention the time at which each operation took place to be easier to debug.

6. New encryption algorithm needs to be created that encrypts the image and also tak es into account every other feature mentioned above.

1.4 Tools and Techniques Used:

1.4.1 Java

Java is a high level language. It is class based. It is object-oriented. And it has very low implementation dependencies. Java is used to develop desktop applications, games, web apps, mobile apps, and more. Java is a pretty solid language for this project because it is great for creating both GUI and implement algorithms like AES.

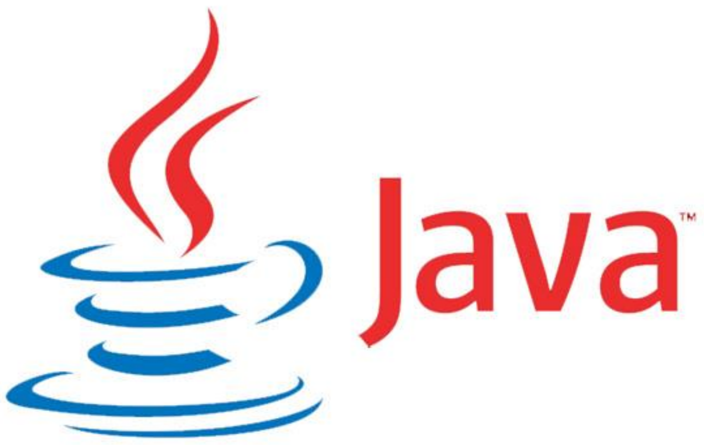


Fig 1.6 Java Logo

1.4.2 Java Cryptography Architecture(JCA)

Cryptography, public key infrastructure, authentication, secure communication, and access control are strongly emphasized with Java.

Java Cryptography Architecture contains a "provider" architecture and a set of APIs for digital signatures, message digests (hashes), certificates and certificate validation, encryption (symmetric/asymmetric block/stream ciphers), key generation and management, and secure random number generation, and more. The aforementioned APIs allow for easy integration of security into code. JCA was designed around following principles.

Implementation independence: Instead of implementing security algorithm themselves, applications can request security services from Java platform. Providers are plugged into Java platform to implement security services through a standard interface.

Applications can rely on more than one independent providers for security functional ity.

Implementation interoperability: Providers have interoperability functionality acro ss different applications. A provider is not bound to a specific application and an app lication is also not bound to a specific provider.

Algorithm extensibility: A number of widely used security services are included in Java platform. Providers are one of them. Providers perform a basic set of security se rvice. However, some applications may rely on emerging standards not yet impleme nted, or on proprietary services. The Java platform supports installation of such cust om providers that implement these services as well.

1.4.3 Java AWT

The Java Abstract Window Toolkit is an API that is used to develop Graphical User Interfaces or window-based applications in java.

AWT is platform-
dependent, which means its components are displayed according to the view of oper ating system. AWT is also heavyweight, which means its components use the operat ing system's resources.

The name of the package is java.awt. This package provides various classes for GUI creation.

For example; TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc, are all classes for the AWT API.

1.4.4 Swing

Swing is a part of Java Foundation Classes (JFC) that is also used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT however, Swing provides platform-
independent and lightweight components.

The name of the package is javax.swing. This package also provides various classes f or window-based application creation.

For example; JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser, etc, are all classes for the swing API.

1.4.5 Intellij IDEA

IntelliJ IDEA is an IDE(integrated development environment) written entirely in Java for the purpose of developing computer software. It was developed by JetBrains. Majority of this project has been developed within this IDE.

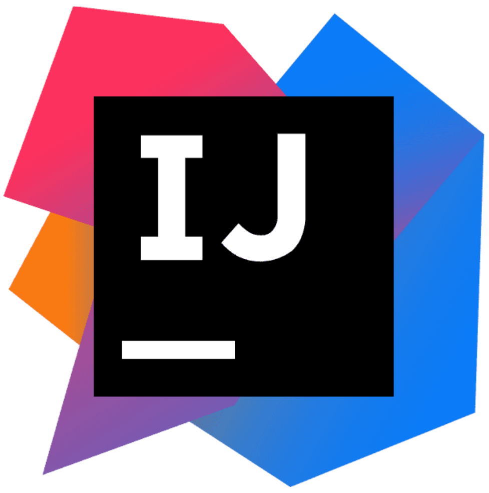


Fig 1.6 Intellij IDEA Logo

CHAPTER 2) LITERATURE SURVEY

2.1 Symmetric and Asymmetric Encryption. [Online] Available at: https://www.professormesser.com/security-plus/sy0-501/symmetric-and-asymmetric-encryption

-By P. Messer

This blog post was the paramount for this project. As a student with no prior knowle dge of cryptography and encryption, this was a starting point for me to learn about th e subject and also influenced me in picking the subject of this project.

The blog talks in explicit detail about Symmetric and Asymmetric Encryption, and th e differences between them.

It also talks about the shared key algorithm-

"If you've implemented symmetric encryption using a single shared key for both the encryption process and the decryption process, if some third party gains access to th is key, you'll need to throw that key away, use a different key, and distribute that key to both the sender and the recipient. This uses a shared key algorithm."

The blog post also discusses how symmetric encryption is relatively faster than asym metric encryption and has less overhead cost. But they are often combined to create a n even stronger encryption.

Influences-

The blog post has significance as the starting point of my deep dive into encryption t echnologies.

2.2 Sneaky "Honey Encryption" Stops Hackers By Drowning Them in Phony Data [Online] Available at: https://gizmodo.com/sneaky-honey-encryption-stops-hackers-by-drowning-the-1511718913

-By A. Feinberg

Another very influential and interesting blog post that talks about an encryption techn ique that turns brute force attacks into an attack against the perpetrators themselves.

Ari Juels, previous chief scientist at computer security company RSA, and Thomas R istenpart of the University of Wisconsin worked together to develop a different type of encryption device that delivers fake data to the intruder every time and incorrect p assword or encryption is guessed.

This encryption algorithm was named Honey Encryption, and makes intrusions infini tely harder for hackers.

Influences-

The blog post helped show how and what king of innovations can be made in the fie ld of cryptography.

2.3 What Is AES and Why You Already Love It [Online] Available at: https://techjury.net/blog/what-is-aes/

-By B. Chernev

This paper talks in detail about AES and AES alone. Chernev starts with an introduct ion to Advanced Encryption Standard. He then talks about its history and why it was needed. Then he gives a deep dive on how the algorithm works using an example. He also discusses its various implementations and safety.

The paper also talks about the various WiFi security protocols. The original WPA-PSK (Wi-Fi Protected Acess- Personal Shared Key) as well as WPA2-PSK were developed using AES algorithm.

Influences-

This paper solidified AES as a must learn and implement algorithm for this project.

2.4 5 Common Encryption Algorithms and the Unbreakable of the Future [Online] Available at: https://blog.storagecraft.com/5-common-encryption-algorithms/

-By C. Bradford

This paper talks about five popular encryption algorithms, namely-
Triple DES, RSA, Blowfish, Twofish, AES.
Triple DES was developed to replace DES algorithm. Triple DES triplicates DES's 56-bit keys to create three keys of total length 168 bit, but equated to 112-bit key. 3DES is deemed to be unsafe after 2024.
RSA algorithm is an acronym that comes from the surnames of its creators-
Ron Rivest, Adi Shamir, and Leonard Adleman. It is public-key encryption system. RSA is an asymmetric encryption algorithm because it uses a pair of keys- public and private keys.

Blowfish is another algorithm made to replace DES. It is a symmetric cipher that splits messages into 64-bits and encrypts them individually. Blowfish is known for its incredible speed.

Twofish is the successor of Blowfish. It is also symmetric. Keys in Twofish can be upto 256-bits in length. It is faster than Blowfish.

AES is the standard algorithm used by US government.

The paper also talks about the future of encryption.

Influences-
This paper helped me contrast and decide which algorithm would be the best for the project and also explore other algorithms for future prospects.

CHAPTER 3) SOFTWARE REQUIREMENT SPECIFICA TIONS

3.1 REQUIREMENT ANALYSIS

Under mentioned is the list of requirements as per the Requirement Specification She et:

- Easy to use and interactive interface.
- User should be able to import/export images and ciphers wherever on the file explorer.
- User should not be able to import a file that is not an image format.
- Separate window for decryption.
- Modified searching capability to facilitate location.
- Apply various filters in image category.
- Able to select the amount of cipher images to generate.
- Maintain encryption algorithm.
- Console of operations for debugging purposes.

3.2 DATA FLOW DIAGRAM

Data Flow Diagrams are used to show the flow of data from external entities into the system, and from one process to another within the system.

There are four important symbols used for drawing a DFD:

- Rectangles representing external entities, which are sources or destinations of data.
- Ellipses representing processes, which take data as input, validate and process it and output it.
- Arrows representing the data flows, which can either, be electronic data or p hysical items.
- Open-
  ended rectangles or a Disk symbol representing data stores, including electron

ic stores such as databases or XML files and physical stores such as filing cab inets or stacks of paper.

Figure below is the Data Flow Diagram for AES Algorithm;

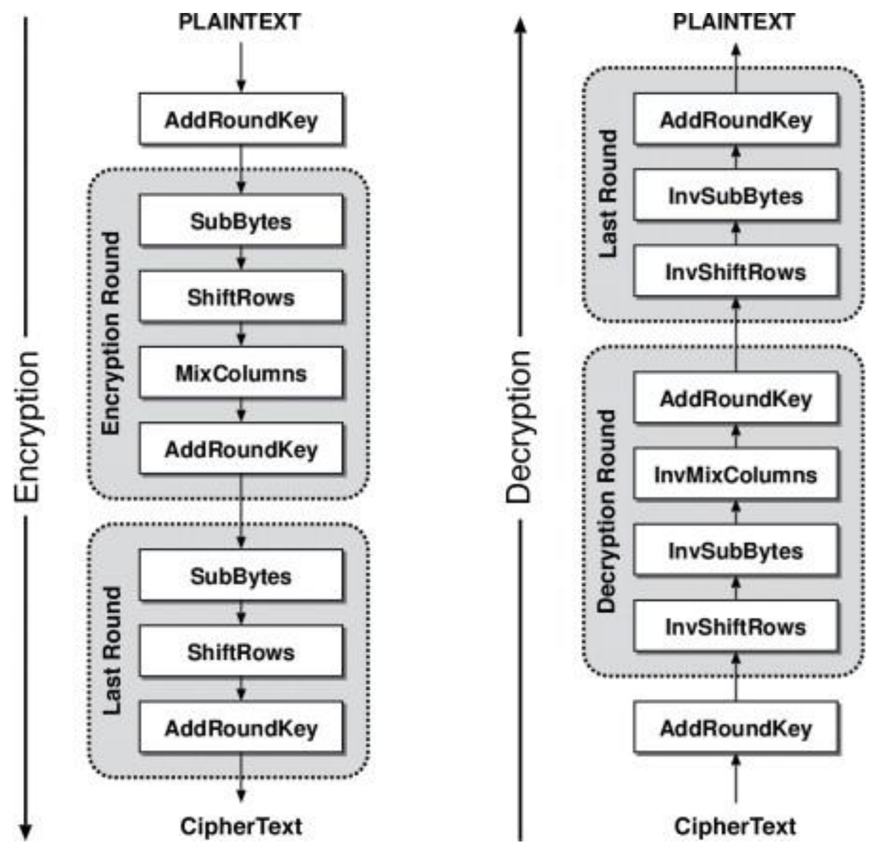


Fig 3.1 AES Algorithm DFD

CHAPTER 4) SYSTEM DEVELOPMENT

4.1 GUI Components

The components that were used for creating Viscrypt GUI are:

Frames:

A Frame is used to define the window. It has a border and a title.

The size of the frame’s border is included within the size of the frame. Because of thi s, the border obscures a part of the frame. A method the obtain the dimensions of the border is getInsets. The dimensions of the border are platform dependent.

The default layout for a frame is called BorderLayout.

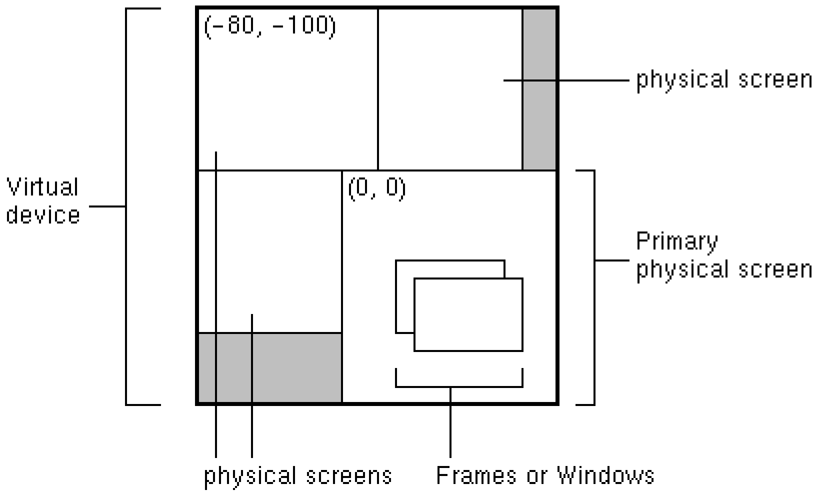


Fig 4.1 Frame Example

GridBagConstraints:

GridBagLayout is a complex and flexible layout manager provided by Java platform. It uses a grid of rows and columns to place components. It allows specified compon ents to span for multiple rows and columns. Rows don’t have to have the same heigh t, and columns don’t have to have the same width.

The figure below shows the grid for an applet. The grid contains three rows and three columns. However, GridBagConstraints allows the button in the second row to span all the columns; and the button in the third row can span for the two right columns.

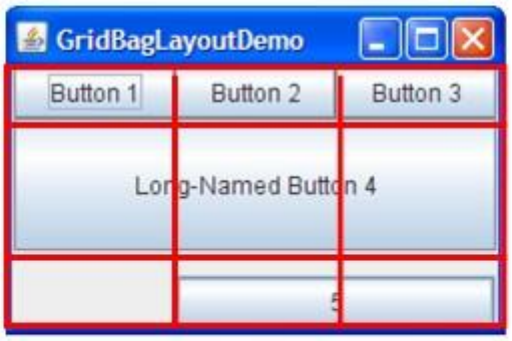


Fig 4.2 GridBagConstraints Example

JLabel:

JLabel class is a component that is used for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

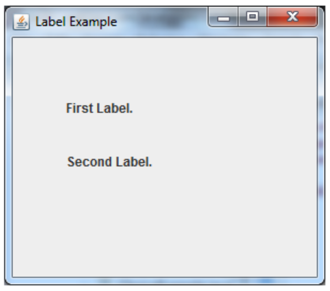


Fig 4.3 JLabel Example

JButton:

Used to create a button with a label. The button has platform independent implement ation. Pushing the button signals the application to perform some operation. It inherit s AbstractButton class.

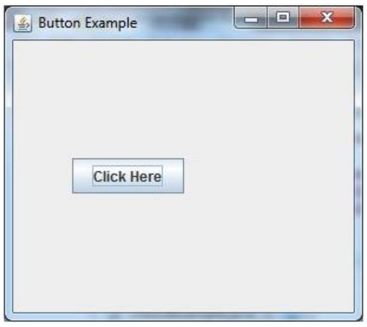
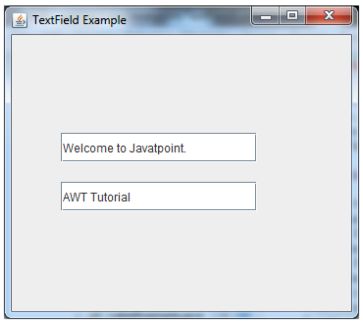


Fig 4.4 JButton Example

JTextField:

This class is used to create a text component that allows the editing of a single line te xt. It inherits JTextComponent class.



Welcome to Javatpoint.

AWT Tutorial

Fig 4.5 JTextField Example

JPanel:

It is the simplest container class. JPanel’s job is to provide a space in which an applic ation can attach any other component. It inherits the JComponents class.

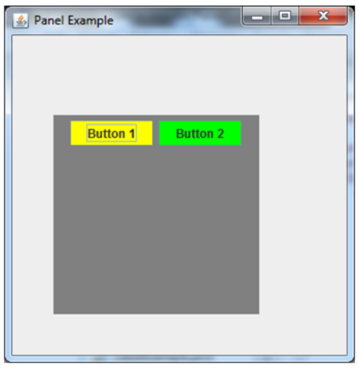


Fig 4.6 JPanel Example

JComboBox:

JComboBox class is used to show popup menu of choices. The choice selected by th e user is shown on the top of a menu. It inherits JComponent class.

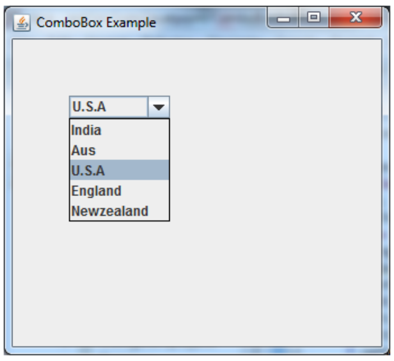


Fig 4.7 JComboBox Example

JSlider:

This class is used to create a slider object. It allows the user to select a value between a specific range.

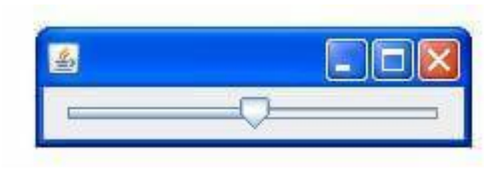
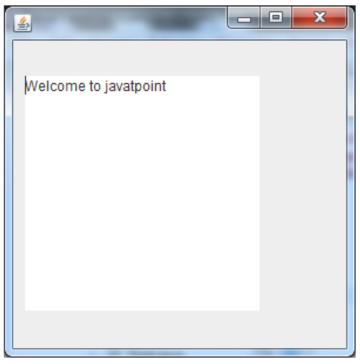


Fig 4.8 JSlider Example

JTextArea:

This class creates a multi line region that displays text. It allows for the editing of the whole region. It inherits JTextComponent class.



Welcome to javatpoint

Fig 4.9 JTextArea Example

JScrollPane:

This class is used to make scroll-
able view of a component. It can be used to display more content when screen size c
an’t accommodate all of the components.

BufferedImage:

BufferedImage is a subclass of the Image class. BufferedImage is used to decide how
 image data is to be handled.

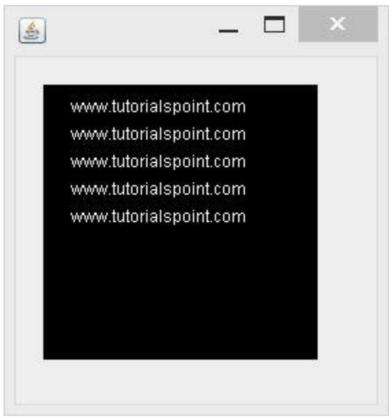

www.tutorialspoint.com
www.tutorialspoint.com
www.tutorialspoint.com
www.tutorialspoint.com
www.tutorialspoint.com

Fig 4.10 BufferedImage Example

4.2 Technical Components

The poject uses three scripts:

4.2.1 Main.java

```
import java.awt.*;
import javax.swing.*;

public class Main {
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                try {
                    for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
```

```
                if ("Windows".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        JFrame frame = new JFrame("VisCry");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(1000, 800);
        frame.setMinimumSize(new Dimension(1000, 700));
        frame.setLayout(new GridBagLayout());
        frame.getContentPane().setBackground(new Color(255, 255, 255));

        new MainFrame(frame);
      }
    });
  }
}
```

4.2.2 MainFrame.java

```
import javax.imageio.ImageIO;
import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.text.SimpleDateFormat;
import java.util.Date;

public class MainFrame {

  private Frame frame;

  private GridBagConstraints gbc;

  private JLabel selectImageLabel;
```

```
    private JButton selectImageButton;

    private JTextField selectImageField;

    private JLabel saveImageLabel;

    private JTextField saveImageField;

    private JButton saveImageButton;

    private JButton generateCipher;

    private JPanel imagePanel;

    private JLabel imageLabel;

    private JComboBox numberBox;

    private JLabel numberOfSecretImages;

    private JSlider thresholdSlider;

    private JLabel thresholdLabel;

    private JTextArea logMessages;

    private JScrollPane logScrollPane;


    private BufferedImage bufferedImage;

    private BufferedImage lastInstance;

    public MainFrame(Frame frame) {
        this.frame = frame;
        initializeObjects();
        createLocation();
    }


    private void initializeObjects() {
        selectImageLabel = new JLabel("Open image: ");
        selectImageField = new JTextField(80);
        selectImageButton = new JButton("Browse");

        String[] numbers = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16",
"17", "18", "19", "20",
                "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31", "32", "33", "34", "35", "36",
"37", "38", "39", "40",
                "41", "42", "43", "44", "45", "46", "47", "48", "49", "50", "51", "52", "53", "54", "55", "56",
"57", "58", "59", "60",
                "61", "62", "63", "64", "65", "66", "67", "68", "69", "70", "71", "72", "73", "74", "75", "76",
"77", "78", "79", "80",
                "81", "82", "83", "84", "85", "86", "87", "88", "89", "90", "91", "92", "93", "94", "95", "96",
"97", "98", "99", "100"};

        numberBox = new JComboBox(numbers);
```

```java
        generateCipher = new JButton("Generate encrypted images");

        saveImageLabel = new JLabel("Save encrypted images: ");
        saveImageField = new JTextField(80);
        saveImageButton = new JButton("Browse");

        imageLabel = new JLabel();
        imagePanel = new JPanel();

        thresholdSlider = new JSlider(JSlider.VERTICAL);
        thresholdSlider.setMinimum(0);
        thresholdSlider.setMaximum(255);
        thresholdSlider.setValue(128);

        numberOfSecretImages = new JLabel("Number of secret images");

        thresholdLabel = new JLabel("Threshold");

        logMessages = new JTextArea(100, 100);
        logMessages.setFont(new Font(logMessages.getFont().getFontName(),
logMessages.getFont().getStyle(), logMessages.getFont().getSize() + 2));
        logMessages.append(new SimpleDateFormat("HH:mm:ss").format(new Date()) + ": VisCry
opened\n");
        logScrollPane = new JScrollPane(logMessages);

        gbc = new GridBagConstraints();
        gbc.insets = new Insets(3, 3, 3, 3);

    }

    void createLocation() {

        gbc.anchor = GridBagConstraints.EAST;
        gbc.weightx = 0.5;
        gbc.weighty = 2;
        gbc.gridx = 0;
        gbc.gridy = 0;
        frame.add(selectImageLabel, gbc);


        gbc.anchor = GridBagConstraints.CENTER;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.weightx = 2;
        gbc.weighty = 2;
        gbc.gridx = 1;
        gbc.gridy = 0;
        selectImageField.addKeyListener(new SelectFileListener());
        frame.add(selectImageField, gbc);


        gbc.anchor = GridBagConstraints.WEST;
        gbc.fill = GridBagConstraints.NONE;
        gbc.weightx = 0.5;
        gbc.weighty = 0.5;
        gbc.gridx = 2;
        gbc.gridy = 0;
    selectImageButton.addActionListener(new SelectButtonListener());
    frame.add(selectImageButton, gbc);


    gbc.anchor = GridBagConstraints.SOUTH;
    gbc.fill = GridBagConstraints.NONE;
    gbc.weightx = 0;
    gbc.weighty = 1;
    gbc.gridx = 1;
    gbc.gridy = 2;
    frame.add(numberOfSecretImages, gbc);


    gbc.anchor = GridBagConstraints.NORTH;
    gbc.weightx = 0;
    gbc.weighty = 0.2;
    gbc.gridx = 1;
    gbc.gridy = 3;
    frame.add(numberBox, gbc);


    gbc.anchor = GridBagConstraints.EAST;
    gbc.weightx = 0;
    gbc.weighty = 1;
    gbc.gridx = 0;
    gbc.gridy = 4;
    frame.add(saveImageLabel, gbc);


    gbc.anchor = GridBagConstraints.CENTER;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.weightx = 5;
    gbc.weighty = 1;
    gbc.gridx = 1;
    gbc.gridy = 4;
    saveImageField.addKeyListener(new SelectFileListener());
    frame.add(saveImageField, gbc);


    gbc.anchor = GridBagConstraints.WEST;
    gbc.fill = GridBagConstraints.NONE;
    gbc.weightx = 0.5;
    gbc.weighty = 0.5;
    gbc.gridx = 2;
    gbc.gridy = 4;
    saveImageButton.addActionListener(new SelectButtonListener());
    frame.add(saveImageButton, gbc);


    gbc.anchor = GridBagConstraints.NORTH;
    gbc.weightx = 0.5;
    gbc.weighty = 1;
    gbc.gridx = 1;
    gbc.gridy = 6;
    generateCipher.addActionListener(new SelectButtonListener());
    frame.add(generateCipher, gbc);
```

```java
        gbc.fill = GridBagConstraints.VERTICAL;
        gbc.weightx = 0.5;
        gbc.weighty = 2;
        gbc.gridx = 0;
        gbc.gridy = 1;
        thresholdSlider.addChangeListener(new ThresholdListener());
        frame.add(thresholdSlider, gbc);


        gbc.anchor = GridBagConstraints.NORTH;
        gbc.fill = GridBagConstraints.NONE;
        gbc.weightx = 0.5;
        gbc.weighty = 1;
        gbc.gridx = 0;
        gbc.gridy = 2;
        frame.add(thresholdLabel, gbc);

        gbc.anchor = GridBagConstraints.CENTER;
        gbc.fill = GridBagConstraints.BOTH;
        gbc.weightx = 0.5;
        gbc.weighty = 5;
        gbc.gridx = 1;
        gbc.gridy = 5;
        frame.add(logScrollPane, gbc);

        frame.setVisible(true);
    }

    public static Dimension getScaledDimension(Dimension imgSize, Dimension boundary) {

        int new_width = imgSize.width;
        int new_height = imgSize.height;

        if (imgSize.width > boundary.width) {
            new_width = boundary.width;
            new_height = (new_width * imgSize.height) / imgSize.width;
        }

        if (new_height > boundary.height) {
            new_height = boundary.height;
            new_width = (new_height * imgSize.width) / imgSize.height;
        }
        return new Dimension(new_width, new_height);
    }


///////////////////////////////////////////////////////////////////////////////////////////////
///             Select File Listener
    public class SelectFileListener implements KeyListener {

        boolean isPressed = false;

        @Override
        public void keyTyped(KeyEvent e) {

        }
```

```
      @Override
      public void keyPressed(KeyEvent e) {
         if (e.getKeyCode() == KeyEvent.VK_ENTER && !isPressed &&
e.getSource().equals(selectImageField)) {
            selectImage(selectImageField.getText());
            isPressed = true;
         }
      }

      @Override
      public void keyReleased(KeyEvent e) {
         isPressed = false;
      }
   }


//////////////////////////////////////////////////////////////////////////////////////////////////
Threshold Listener
   public class ThresholdListener implements ChangeListener {

      @Override
      public void stateChanged(ChangeEvent e) {
         if(lastInstance != null) {
            BufferedImage grayImage = GenerateImage.thresholdImage(bufferedImage,
thresholdSlider.getValue());
            Dimension boundary = new Dimension(700, 700);
            Dimension d = new Dimension(bufferedImage.getWidth(), bufferedImage.getHeight());
            Dimension scaled = getScaledDimension(d, boundary);
            Image img = grayImage.getScaledInstance(scaled.width, scaled.height,
Image.SCALE_SMOOTH);
            imageLabel.setIcon(new ImageIcon(img));
            imageLabel.setPreferredSize(scaled);
            imagePanel.add(imageLabel);
            gbc.anchor = GridBagConstraints.CENTER;
            gbc.fill = GridBagConstraints.NONE;
            gbc.weightx = 0;
            gbc.weighty = 2;
            gbc.gridx = 1;
            gbc.gridy = 1;
            frame.add(imagePanel, gbc);
            frame.validate();
            lastInstance = grayImage;
         }
      }
   }


//////////////////////////////////////////////////////////////////////////////////////////////////
Select Button Listener
   public class SelectButtonListener implements ActionListener{

      @Override
      public void actionPerformed(ActionEvent e) {
         if(e.getSource().equals(selectImageButton)){
            FileDialog fileDialog = new FileDialog(frame, "Select image");
            fileDialog.setVisible(true);
            if(fileDialog.getFile() != null && fileDialog.getDirectory() != null){
                selectImage(fileDialog.getDirectory() + fileDialog.getFile());
            }
        }


        if(e.getSource().equals(saveImageButton)){
            JFileChooser fileChooser = new JFileChooser();
            fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);

            if (fileChooser.showOpenDialog(frame) == JFileChooser.APPROVE_OPTION){
                saveImageField.setText(fileChooser.getSelectedFile().toString());
            }
        }


        if(e.getSource().equals(generateCipher)){
            try {
                File file = new File(saveImageField.getText());
                if(file.isDirectory()){
                    int width = lastInstance.getWidth();
                    int height = lastInstance.getHeight();

                    BufferedImage lastKey = new BufferedImage(width, height,
BufferedImage.TYPE_BYTE_GRAY);

                    int number = Integer.valueOf(numberBox.getSelectedItem().toString());
                    boolean isEven = (number % 2 == 0);

                    for (int i = 0; i < number - 1; i++) {
                        BufferedImage temp = GenerateImage.getRandomizedImage(width, height);
                        ImageIO.write(temp, "png", new File(saveImageField.getText() + "\\Viscry " + (i + 1) +
".png"));

                        lastKey = GenerateImage.getExcludedImage(lastKey, temp, isEven);

                        logMessages.append(new SimpleDateFormat("HH:mm:ss").format(new Date()) + ":
Image \"Viscry " + (i + 1) + ".png\" is saved\n");
                    }
                    lastKey = GenerateImage.getExcludedImage(lastKey, lastInstance, isEven);
                    ImageIO.write(lastKey, "png", new File(saveImageField.getText() + "\\VisCry " +
(number) + ".png"));

                    logMessages.append(new SimpleDateFormat("HH:mm:ss").format(new Date()) + ":
Image \"VisCry" + number + ".png\" is saved\n");
                    logMessages.append("\n========================ENCRYPTING IS DONE
SUCCESSFULLY!===========================\n");
                }
                else {
                    logMessages.append(new SimpleDateFormat("HH:mm:ss").format(new Date()) + ":
Directory \"" + saveImageField.getText() + "\" doesn't exist\n");
                }
            }
            catch (Exception ex){
                logMessages.append(new SimpleDateFormat("HH:mm:ss").format(new Date()) + ":
Couldn't save images\n");
            }
        }
    }
}


private void selectImage(String filePathAndName){
    try {
        selectImageField.setText(filePathAndName);

        bufferedImage = ImageIO.read(new File(filePathAndName));

        BufferedImage grayImage = GenerateImage.thresholdImage(bufferedImage,
thresholdSlider.getValue());
        Dimension boundary = new Dimension(700, 700);
        Dimension d = new Dimension(bufferedImage.getWidth(), bufferedImage.getHeight());
        Dimension scaled = getScaledDimension(d, boundary);
        Image img = grayImage.getScaledInstance(scaled.width, scaled.height,
Image.SCALE_SMOOTH);
        imageLabel.setIcon(new ImageIcon(img));
        imageLabel.setPreferredSize(scaled);
        imagePanel.add(imageLabel);
        gbc.anchor = GridBagConstraints.CENTER;
        gbc.fill = GridBagConstraints.NONE;
        gbc.weightx = 0;
        gbc.weighty = 0;
        gbc.gridx = 1;
        gbc.gridy = 1;
        frame.add(imagePanel, gbc);
        frame.validate();
        lastInstance = grayImage;
        logMessages.append(new SimpleDateFormat("HH:mm:ss").format(new Date()) + ": Image
loaded\n");

    }catch (Exception ex){
        logMessages.append(new SimpleDateFormat("HH:mm:ss").format(new Date()) + ": File is not
supported. Please, use images with extension: jpg, bmp, png\n");
    }
}
}
```

4.2.3 GenerateImage.java

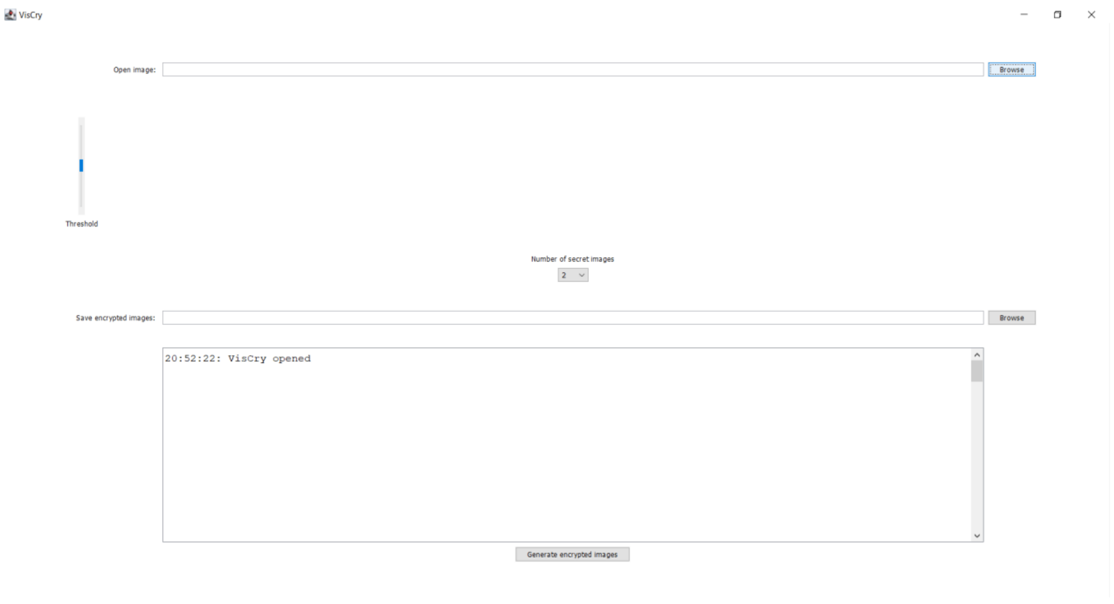
```
import java.awt.image.BufferedImage;
import java.awt.image.WritableRaster;
import java.util.Random;

public class GenerateImage {
    public static BufferedImage getRandomizedImage(int width, int height){
        BufferedImage result = new BufferedImage(width, height, BufferedImage.TYPE_BYTE_GRAY);
        WritableRaster raster = result.getRaster();
        int[] pixels = new int[width];
        for(int y = 0; y < height; y++){
            for(int i = 0; i < width; i++){
                if(new Random().nextBoolean()){
                    pixels[i] = 255;
                }
                else{
                    pixels[i] = 0;
                }
            }
            raster.setPixels(0, y, width, 1, pixels);
        }
        return result;
    }

    public static BufferedImage getExcludedImage(BufferedImage first, BufferedImage second, boolean
isEven){
        int width = first.getWidth();
        int height = first.getHeight();
        BufferedImage result = new BufferedImage(first.getWidth(), first.getHeight(),
BufferedImage.TYPE_BYTE_GRAY);
        WritableRaster raster = result.getRaster();
        WritableRaster firstRaster = first.getRaster();
        WritableRaster secondRaster = second.getRaster();
        int[] pixels1 = new int[width];
        int[] pixels2 = new int[width];
        int[] pixels = new int[width];

        if(isEven){

            for(int y = 0; y < height; y++){
                firstRaster.getPixels(0, y, width, 1, pixels1);
                secondRaster.getPixels(0, y, width, 1, pixels2);
                for(int i = 0; i < width; i++){
                    if(pixels1[i] == pixels2[i]){
                        pixels[i] = 255;
                    }
                    else {
                        pixels[i] = 0;
                    }
                }
                raster.setPixels(0, y, width, 1, pixels);
            }
        }
        else{
            for(int y = 0; y < height; y++){
                firstRaster.getPixels(0, y, width, 1, pixels1);
                secondRaster.getPixels(0, y, width, 1, pixels2);
                for(int i = 0; i < width; i++){
                    if(pixels1[i] == pixels2[i]){
                        pixels[i] = 0;
                    }
                    else {
                        pixels[i] = 255;
                    }
                }
                raster.setPixels(0, y, width, 1, pixels);
            }
        }
        return result;
    }

    public static BufferedImage thresholdImage(BufferedImage image, int threshold) {
        BufferedImage result = new BufferedImage(image.getWidth(), image.getHeight(),
BufferedImage.TYPE_BYTE_GRAY);
        result.getGraphics().drawImage(image, 0, 0, null);
        WritableRaster raster = result.getRaster();
        int[] pixels = new int[image.getWidth()];
        for (int y = 0; y < image.getHeight(); y++) {
            raster.getPixels(0, y, image.getWidth(), 1, pixels);
            for (int i = 0; i < pixels.length; i++) {
                if (pixels[i] < threshold) pixels[i] = 0;
                else pixels[i] = 255;
            }
            raster.setPixels(0, y, image.getWidth(), 1, pixels);
        }
        return result;
    }
}
```

CHAPTER 5) RESULTS AND PERFORMANCE ANALY SIS

5.1 Imported Image

Fig 5.1 Imported Image

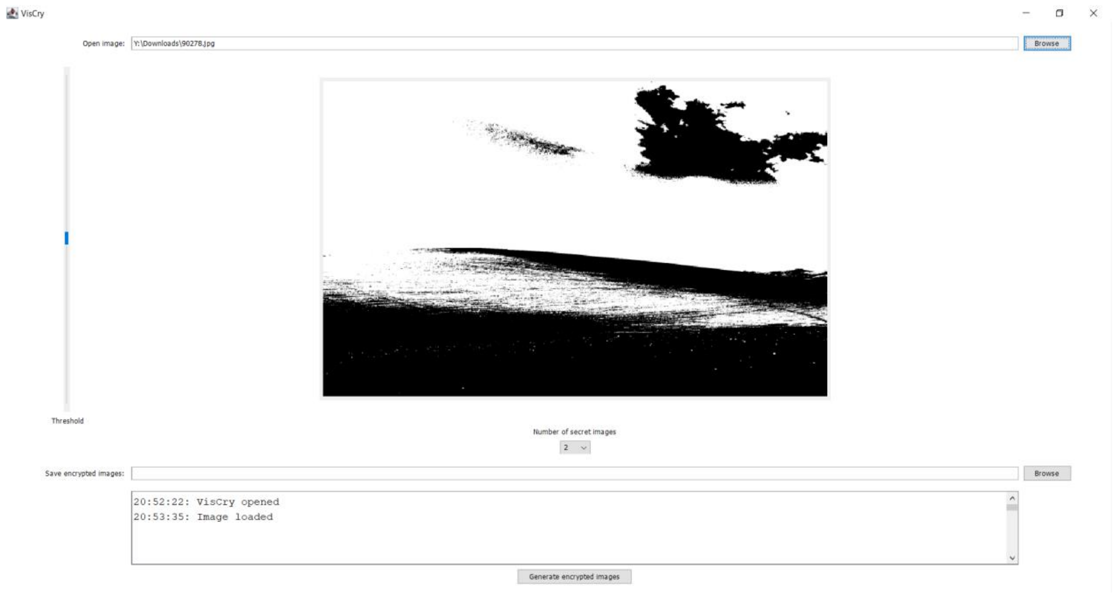
5.2 Viscrypt Interface



```
20:52:22: VisCry opened
```

Fig 5.2 Viscrypt Interface

5.3 Loaded Image



Open image: Y:\Downloads\90278.jpg

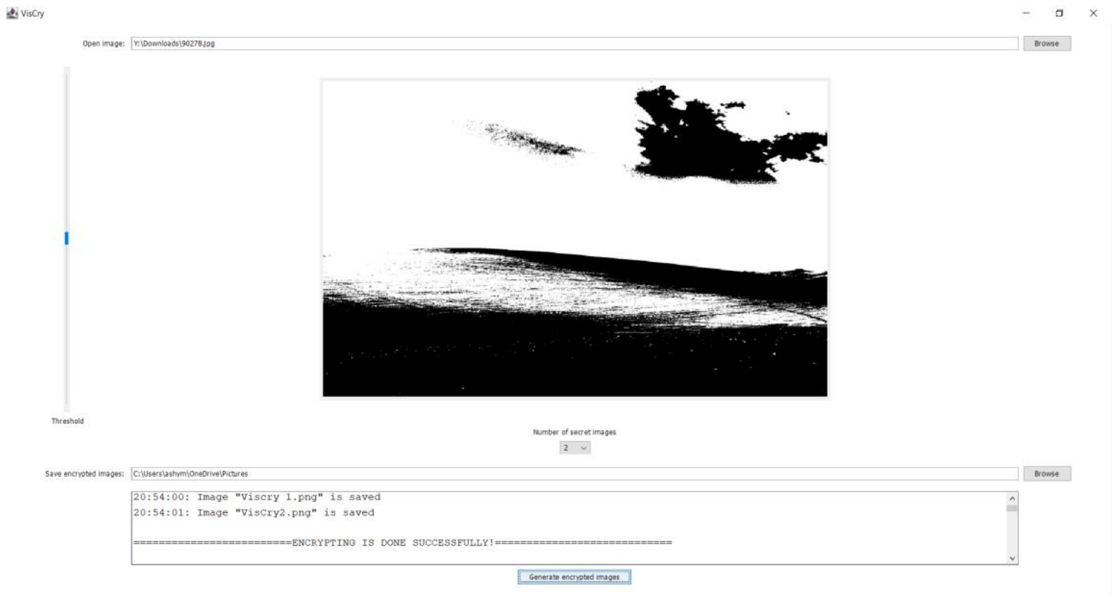
Number of secret images: 2

Save encrypted images:

```
20:52:22: VisCry opened
20:53:35: Image loaded
```

Fig 5.3 Loaded Image

5.4 Encryption Complete



Open image: Y:\Downloads\90278.jpg

Number of secret images: 2

Save encrypted images: C:\Users\ashym\OneDrive\Pictures

```
20:54:00: Image "Viscry 1.png" is saved
20:54:01: Image "VisCry2.png" is saved

========================ENCRYPTING IS DONE SUCCESSFULLY!==========================
```

Fig 5.4 Encryption Complete

5.5 Cipher Images

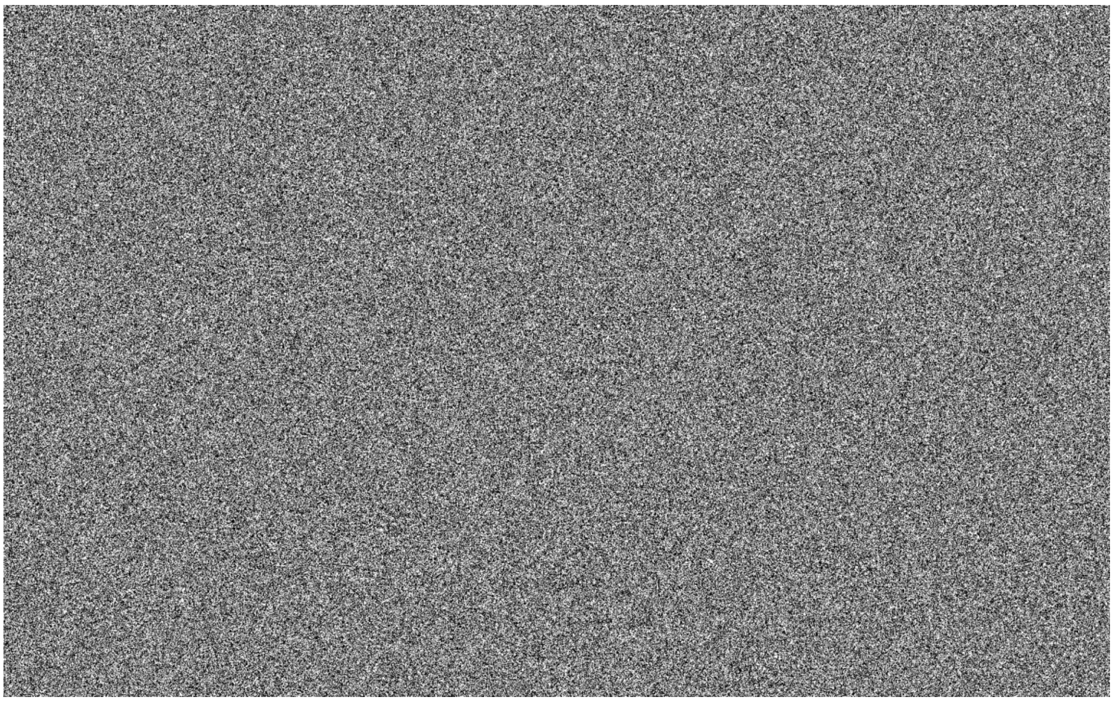


Fig 5.5 Cipher Image 1

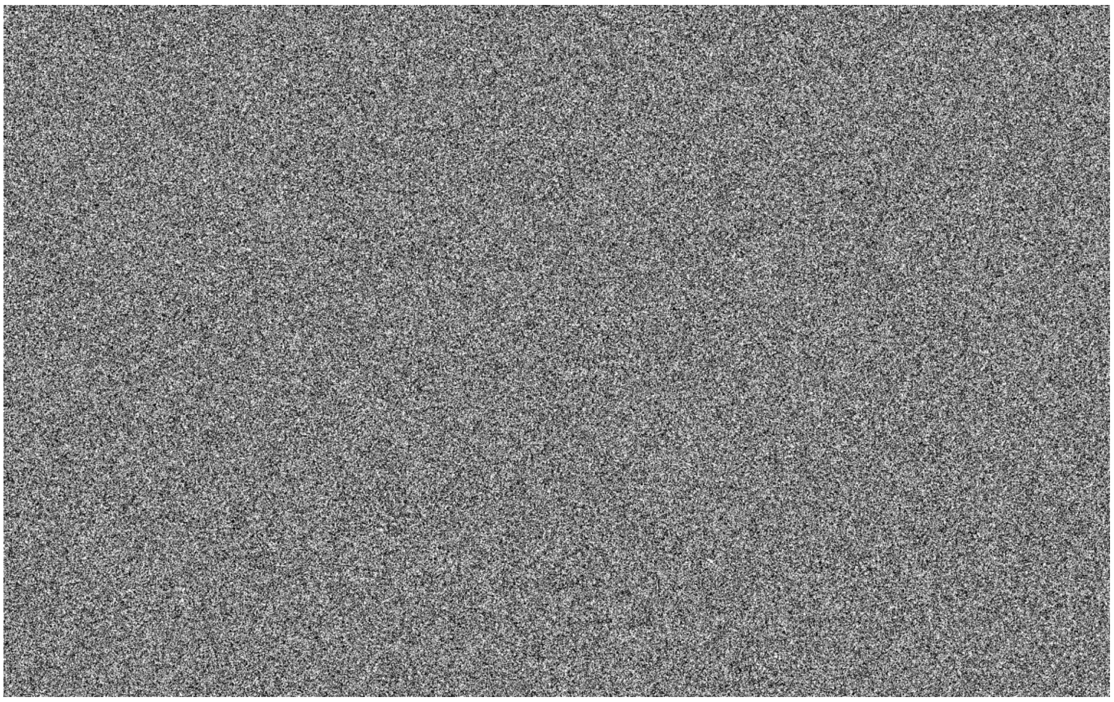


Fig 5.6 Cipher Image 2

CHAPTER 6) CONCLUSION

7.1 Declaration

I have successfully created and implemented the Application ‘Viscrypt’.

I hope that it serves its purpose and proves to be helpful for people long run.

7.2 Limitations

Despite endless efforts put in by me, Viscrypt still faces some drawbacks:

- This application does only deals with encryption.
- This application does not deal with decryption. For decryption, any third-party image editing tool with exclusion tool can be used.

7.3 Maintenance

In the maintenance phase, the developers must make changes to the hardware, softwa re and the documentation. It involves supporting its operation effectiveness for a peri od of time. It may include improving optimization, fixing bugs, improving security, o r addressing user reports.

Repositories and other management standards and procedures are employed to make sure that the modifications don’t lead to disruption of ongoing operations, or negative ly impact performance.

Besides major changes, developers also push routine updates which involves procedu res like requesting, evaluating, approving, testing, installing and documenting websit e modifications.

QA, auditing, security, and end-
user should be included in the update management processes. Before implementing a ny major updates, it is necessary to do risk and security reviews to decide whether up date should be finally implemented.

For maintenance of this application:

1. The project itself needs some feature updates, such as a decryption module.
2. Perform script optimization.
3. Perform rigorous testing.

7.4 Future Scope and Enhancement

1. I look forward to extend the features of this application to be able to decrypt c ipher images as well.
2. So far, its usability is minimum. I aim to improve the application to perform more complex encryption.
3. I am looking forward to implement what I have learned through this project t o other personal projects as well.

CHAPTER 8) BIBLIOGRAPHY

- Abdullah, A. M., 2017. Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data.Cryptography and Network Security, pp. 1-12."Location Based Encryption-Decryption Approach for Data Security"
- Anon., 2019. Advanced Encryption Standard. [Online] Available at: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
- Bradford, C., 2019. Storage Craft. [Online] Available at: https://blog.storagecraft.com/5-common-encryption-algorithms/[Accessed 14 October 2019]
- Abhilasha CP, N. K., 2016. Software Implementation of AES Encryption Algorithm. International Journal ofAdvanced Research in Computer Science and Software Engineering, 6(5), pp. 201-205.
- Shraddha Wade, A. G. A. K. V. D., 2017. Design Enhance AES Data Encryption and Decryption.International Journal of Scientific Research in Science and Technology, 3(2), pp. 136-138.
- Messer, P., 2018. Symmetric and Asymmetric Encryption. [Online] Available at: https://www.professormesser.com/security-plus/sy0-501/symmetric-and-asymmetric-encryption/