A Project Report

On

Visual Cryptography (Image Encryption and Decryption)

Submitted in partial fulfillment of the

requirement for the award of the degree of

BACHELOR OF COMPUTER APPLICATION

Session 2023-24

in

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

By

Amit Yadav
21SCSE1470010
Rupesh Kumar Jha
21SCSE1430010
Shivam Kumar Jha
21SCSE1470005

Under the guidance of

Ms. Aishwarya
Assistant Professor

SCHOOL OF COMPUTER APPLICATION AND TECHNOLOGY
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
April 2024

SCHOOL OF COMPUTER APPLICATION AND TECHNOLOGY

GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE’S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **“Visual Cryptography (Image Encryption and Decryption)”** in partial fulfillment of the requirements for the award of the BCA (Bachelor of Computer Application) submitted in the School of Computer Application and Technology of Galgotias University, Greater Noida, is an original work carried out during the period of January, 2024 to May and 2024, under the supervision of **Ms. Aishwarya Assistant Professor,** Department of School of Computer Application and Technology, Galgotias University, Greater Noida.

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Amit Yadav, 21SCSE1470010

Rupesh Kumar Jha, 21SCSE1430010

Shivam Kumar Jha, 21SCSE1470005

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Ms. Aishwarya

Assistant Professor

CERTIFICATE

This is to certify that Project Report entitled **"Visual Cryptography (Image Encryption and Decryption)"** which is submitted by Amit Yadav (21SCSE1470010), Rupesh Kumar Jha(21SCSE1430010), Shivam Kumar Jha(21SCSE1470005) in partial fulfillment of the requirement for the award of degree BCA. in Department of **SCAT** of School of Computer Application and Technology, Galgotias University, Greater Noida, India is a record of the candidate own work carried out by him/them under my supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Signature of Examiner(s) **Signature of Supervisor(s)**

Date: April 2024
Place: Greater Noida

Abstract

In today's digital era, Digital image security is an ongoing concern as the digital landscape expands. The widespread sharing of images online makes them vulnerable to privacy breaches and unauthorized access. Traditional image formats are often not protected adequately, leaving them vulnerable to cyber threats and data breaches. Traditional image storage and transmission methods are vulnerable to unauthorized access and cyberattacks, which makes it important to develop robust image encryption solutions. This project introduces an intuitive software application that empowers users to encrypt and decrypt images effortlessly. It offers a method to transform images into ciphertext, rendering them unreadable to unauthorized individuals. The decryption process, using the corresponding technique, allows the restoration of the original image, ensuring data privacy and integrity. Through the application of advanced cryptographic techniques, to realize this solution, we used symmetric-key cryptography technique in programming language Java to implement the project. The project utilizes a simple cryptographic technique, to ensure the security of the encryption process. Our project has yielded promising results, with successful encryption and decryption of various digital images. the encryption process provides robust protection against unauthorized access, and the decryption process reliably restores images to their original form. Furthermore, our methods demonstrate efficiency in terms of execution time and image quality preservation. The Image Encryption and Decryption project addresses a critical need for data security in the digital age, particularly in the context of visual data. As an area of constant evolution, the future scope of this project includes enhancements to existing methods, exploration of new encryption techniques, and adaptation for emerging technologies.

Table of Contents

Title			Page No.
Candidates Declaration			**I**
Acknowledgement			**II**
Abstract			**III**
Contents			**IV**
List of Table			**V**
List of Figures			**VI**
Acronyms			**VII**
Chapter 1	**Introduction**		
	1.1	Introduction	**1**
	1.2	Formulation of Problem	**4**
	1.2.1	Tool and Technology Used	**4**
	1.3	Scope of the project	**9**
	1.3.1	Primary Objectives	**9**
	1.3.2	Secondary Objective	**11**
	1.4	System Features	**12**
	1.4.1	Encryption module	**12**
	1.4.2	Decryption Module	**13**
	1.4.3	User Interface	**14**
Chapter 2	**Literature Survey/Project Design**		
	2.1	Problem Definition	**16**
	2.2	Previous Research	**17**
	2.3	Methodology	**20**
	2.3.1	Data Collection	**20**
	2.3.2	Encryption Technique	**20**
	2.3.3	Encryption Process	**20**
	2.3.4	Decryption Process	**21**
	2.3.5	Security Analysis	**22**
	2.3.6	Limitations	**22**
	2.3.7	Graphical User Interface	**24**
	2.4	Behavioral Description	**24**
	2.5	Analysis	**28**
Chapter 3	**Functionality/Working of Project**		**33**

Chapter 4 Results and Discussion **40**
Chapter 5 Conclusion and Future Scope **41**
5.1 Conclusion
5.2 Future Scope

List of Figures

S.No.	Title	Page No.
1	Fig 1.1: **Representation**	**15**
2	Fig 2.1: **Process**	**27**
3	Fig 2.2: **Cryptography**	**30**
4	Fig 3.1: **Implementation-1**	**33**
5	Fig 3.2: **Implementation-2**	**34**
6	Fig 3.3: **Implementation-3**	**35**
7	Fig 3.4: **Working of Project-1**	**36**
8	Fig 3.5: **Working of Project**	**36**
9	Fig 3.6: **Image Selection**	**37**
10	Fig 3.7: **Using The Key**	**37**
11	Fig 3.8: **Encryption**	**37**
12	Fig 3.9: **Output-1**	**38**
13	Fig 3.10: **Selecting Image**	**38**
14	Fig 3.11: **Decryption**	**39**
15	Fig 3.12: **Output-2**	**39**

Acronyms

DSS	Digital Signature Standard
AES	Advanced Encryption Standard
MAC	Message Authentication Code
DSA	Digital Signature Algorithm
CFB	Cipher Feedback
PKC	Public Key Cryptography
SCSE	School of Computing Science and Engineering
DFD	Data Flow Diagram
ECC	Elliptic Curve Cryptography

CHAPTER-1

Introduction

1.1 Introduction

Computer has become an essential device now a days. The main use of computer is to store data and send it from one location to other. The information that is shared must be transferred in a secured manner. Cryptography, then, not only protects data from theft or alteration, but can also be used for user authentication. There are, in general, three types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public key (or asymmetric) cryptography, and hash functions. In today's digital age, the security and privacy of sensitive data are of paramount importance. Images, just like any other form of data, can be vulnerable to unauthorized access, especially when transmitted or stored electronically. Image encryption is a crucial technique that ensures the confidentiality and integrity of these visual assets. This process involves converting an image into a secure format using encryption algorithms, making it virtually impossible for unauthorized individuals to comprehend the original content. In this context, this project focuses on implementing image encryption and decryption in Java using a fundamental yet effective approach. We will convert the image into a stream of bytes, enabling us to manipulate and secure it with a secret key provided by the user. One of the key aspects of this encryption method is the use of the exclusive OR (XOR) operation. XOR is a bitwise operation that is employed to combine the bytes of the image with the key. The result is an encrypted image that only becomes legible when decrypted using the correct key. This project will guide you through the process of image encryption and decryption, providing you with the Java code and a step-by-step explanation of how to achieve this vital security enhancement for your image data.

By the end of this tutorial, you will have a strong foundation for building image encryption and decryption applications to protect your visual data from prying eyes.

The main intention of this to implement file security using the latest and strongest algorithm, named after the founders, after Vincent Rijmen and Joan Daemon, who first published the algorithm in . The project is divided into two modules. The first module deals with encryption, the second part deals with decryption.

ENCRYPTION

Encryption is the process of transforming information from an unsecured form ("clear" or "plaintext") into coded information ("cipher text"), that cannot be easily read by outside parties. An algorithm and a key control the transformation process. The process must be reversible so that the intended recipient can return the information to its original, readable form, but reversing the process without the appropriate encryption information should be impossible. This means that details of the key must also be kept secret. Encryption is generally regarded as the safest method of guarding against accidental or purposeful security breaches. The strength of the encryption method is often measured in terms of work factor - the amount of force that is required to 'break' the encryption. A strong system will take longer to break although applying greater force can reduce this (the more effort that is put into the attack, the less time required to break the code).

The main characteristics of private key cryptosystem are as follows:

- In private key encryption, the same key is used for both encryption and decryption. The key must be kept secret so that unauthorized parties cannot, even with knowledge of the algorithm, complete the decryption process.
- Once the encryption part is carried out, the main next part is the decryption, where the cipher text has to be converted back into the original text, so that whole process of file

transfer is implemented. The receiver is interested in receiving only the original text, therefore decryption plays a vital role in this project.

The main problems that are dealt in the APTS, which mainly works on projects that deal with communication, are given below in detail. The need of the hour was to implement algorithms like Rijndeal so that security over the data transmitted could be assured. Yet another factor was the efficiency that this algorithm supported.

Types and Sources of File Threats

1) Unauthorized Access “Unauthorized access” is a very high-level term that can refer to a number of different sorts of attacks. The goal of these attacks is to access some resource that your machine should not provide the attacker.

2) Executing Commands Illicitly

It’s obviously undesirable for an unknown and untrusted person to be able to execute commands on your server machines. There are two main classifications of the severity of this problem: normal user access, and administrator access. A normal user can do a number of things on a system (such as read files, mail them to other people, etc.) that an attacker should not be able to do.

On the other hand, an attacker might wish to make configuration changes to a host (perhaps changing its IP address, putting a start-up script in place to cause the machine to shut down ever time it’s started or something similar). In this case, the attacker will need to gain administrator privileges on the host.

3) Confidentiality Breaches There is certain information that could be quite damaging if it fell into the hands of a competitor, an enemy, or the public. In these cases, it’s possible that compromise of a normal user’s account on the machine can be enough to cause damage (perhaps in the form of PR, or obtaining information that can be used against the company, etc.)

While many of the perpetrators of these sorts of break-ins are merely thrill-seekers interested in nothing more than to see a shell prompt for your computer on their screen, there are those who are more malicious.

4) Destructive Behavior Among the destructive sorts of break-ins and attacks, one of the two major categories is.

1.2 Formulation of Problem

Privacy is a critical concern when sharing sensitive visual data due to the potential for unauthorized access and misuse of the information. Privacy is a paramount concern when sharing sensitive visual data as it involves safeguarding individuals' or organizations' confidential information from unauthorized access and disclosure. Visual data, which can include images, documents, or any form of visual content, often contains personal or sensitive details. Privacy concerns in sharing such data arise due to the potential for breaches, leaks, or misuse, which can result in identity theft, financial fraud, reputation damage, or other serious consequences. To address these concerns, robust encryption and secure sharing mechanisms must be employed to ensure that only authorized individuals or entities can access and decipher the visual data. Techniques like visual cryptography and end-to-end encryption are vital for maintaining the privacy and confidentiality of sensitive visual information, allowing individuals and organizations to share critical data while minimizing the risk of privacy violations. Additionally, strict access controls, strong authentication, and adherence to data protection regulations are essential components of a comprehensive strategy for addressing privacy concerns in the sharing of sensitive visual data.

1.2.1 Tools and Technology Used

Programming Languages: Java is a versatile and popular programming language that is widely used for encryption and decryption tasks. It offers several advantages for implementing encryption and decryption processes, making it a preferred choice in various domains, from securing sensitive data to enabling secure communication

Java includes the Java Cryptography Architecture, which provides a comprehensive framework for performing cryptographic operations, including encryption and decryption. The JCA offers a set of classes and libraries that simplify the implementation of cryptographic algorithms.

IDE: An Integrated Development Environment (IDE) for Java is a software application or platform that provides a comprehensive set of tools and features for Java software development. It is designed to streamline the entire software development process, from writing and editing code to debugging, testing, and building Java applications.

Popular Java IDEs include Eclipse, IntelliJ IDEA, NetBeans, and Oracle's JDeveloper. These IDEs offer various features and are often tailored to specific use cases, such as enterprise application development, mobile app development, or web development. Developers can choose the Java IDE that best suits their needs and preferences, depending on the complexity and scope of their projects.

Image Processing Software: Image processing software tools such as Adobe Photoshop, GIMP, or specialized visual cryptography software can be used to create and manipulate images for use in visual cryptography schemes. These tools are used for image preparation and transformation.

Random Number Generators: Visual cryptography often relies on random numbers for generating shares. Cryptographically secure random number generators are crucial to ensure the security of the generated shares.

Visual Cryptography Libraries: Some libraries and frameworks have been developed to facilitate the implementation of visual cryptography schemes. These libraries provide pre-built functions and tools to create and manipulate visual cryptography shares.

Authentication and Verification Tools: Tools for verifying the authenticity of visual cryptography shares are used to ensure that the shares have not been tampered with and are genuine. The choice of tools depends on the specific application and requirements of visual cryptography. Researchers and practitioners may use a combination of these tools and methods to create, distribute, and reconstruct visual cryptography shares while ensuring the security and privacy of the underlying data.

Online Visual Cryptography Tools: There are online tools and web applications that allow users to create and decode visual cryptography shares without the need for programming. These tools are user-friendly and accessible for those who may not be proficient in coding.

JDK Or (Java Development Kit): JDK stands for "Java Development Kit." It is a software package provided by Oracle Corporation (and previously by Sun Microsystems) that includes a set of tools, executables, and libraries for developing, compiling, and running Java applications. The JDK is an essential component for Java developers, as it provides everything needed to create Java applications, applets, and other Java-based software.

The JDK typically includes the following components:

- Java Compiler (javac): This tool is used to compile Java source code (.java files) into bytecode (.class files), which can be executed by the Java Virtual Machine (JVM).
- Java Virtual Machine (JVM): The JVM is responsible for executing Java bytecode. It interprets or compiles the bytecode into machine code that can run on the target system.
- Standard Java Class Libraries: The JDK includes a rich set of class libraries, also known as the Java Standard Library, which provides a wide range of pre-built classes and functions for various tasks, including file handling, network communication, and user interface development.

- Debugging Tools: The JDK contains tools for debugging Java code, such as the Java Debugger (jdb) and various options for debugging and profiling Java applications.
- Documentation: The JDK includes comprehensive documentation, including the Java API documentation, which describes the classes and methods available in the Java Standard Library.
- Java Development Tools: The JDK may include integrated development environments (IDEs), such as Oracle's NetBeans or other development tools, to make the development process more efficient.

The JDK is available for various platforms, including Windows, macOS, and Linux, and different versions of the JDK are released over time to incorporate new features, improvements, and bug fixes. Java developers use the JDK to write, compile, and run Java applications, and it is a fundamental tool for Java software development.

JAVA Swing Package: Java Swing, often referred to as simply "Swing," is a set of graphical user interface (GUI) libraries and components for building desktop applications in Java. Swing is part of the Java Foundation Classes (JFC) and is included in the Java Standard Library.

Swing provides a rich framework for creating user interfaces with features such as:

- GUI Components: Swing offers a wide range of GUI components, including buttons, text fields, labels, panels, tables, trees, and more. These components are highly customizable and can be used to create complex user interfaces.
- Event Handling: Swing supports event-driven programming, allowing developers to define how components respond to user actions like button clicks or mouse movements.

- Layout Managers: Swing provides layout managers that help control the arrangement and sizing of components within a container, ensuring that user interfaces look consistent across different platforms.
- Lightweight Components: Swing components are "lightweight" because they don't rely on the underlying operating system's native GUI components. This makes Swing highly portable and consistent across different platforms.
- Double Buffering: Swing components use double buffering to reduce flicker and provide smooth graphics rendering.
- Customization: Swing components can be customized through various properties, methods, and look-and-feel (L&F) support, allowing developers to adapt the appearance and behavior of their applications.
- Internationalization: Swing has built-in support for internationalization and localization, making it easier to create applications for global audiences.
- Accessibility: Swing applications can be designed to be accessible to users with disabilities, as it supports the Java Accessibility API.

To use Swing, you need to import classes from the javax. swing package. Swing is a powerful and flexible library that has been widely used for developing cross-platform desktop applications in Java. It provides a consistent and attractive user interface for applications running on different operating systems, making it a popular choice for desktop software development in the Java ecosystem.

1.3 Scope of the project:

The Image Encryption and Decryption using XOR Method project aim to develop a robust and user-friendly system for securing digital images through encryption. The scope of the project encompasses various aspects, including the target audience, features, and limitations.

The detailed project scope emphasizes a holistic approach to developing an Image Encryption and Decryption system. By addressing primary and secondary objectives, incorporating key features, and recognizing potential limitations, the project aims to provide a secure, user-friendly, and efficient solution for image security. Future enhancements are outlined to guide ongoing development and improvement efforts.

1.3.1 Primary Objectives:

1) Algorithm Implementation:
   a) Develop algorithms for image encryption and decryption using the XOR method.
   b) Implement pixel-level binary conversion and XOR operations for both encryption and decryption processes.
2) User Interface Development:
   a) Design an intuitive graphical user interface (GUI) for users to interact with the encryption and decryption functionalities.
   b) Allow users to select images for processing, input encryption keys, and visualize the progress and results.
3) Performance Assessment:
   a) Evaluate the performance of the XOR encryption approach in terms of execution time and resource utilization.
   b) Identify potential bottlenecks and optimize the algorithms for efficiency.

4) User Interface:

- Image Selection:

a) Allow users to select image files for encryption or decryption.

- Key Input:

b) Provide input fields for users to enter encryption and decryption keys.

- Progress Indicator:

c) Display a progress bar or indicator to inform users about the status of ongoing encryption and decryption processes.

- Result Display:

d) Showcase the encrypted or decrypted image along with relevant details such as execution time and any error messages.

5) Limitations

Key Management:

- The security of the system relies on the confidentiality and integrity of the encryption keys. Adequate measures must be taken to safeguard key information.

Image Size:

- The system's efficiency may vary with the size of images. Large images could impact processing time and resource consumption.

Security Assessment:

- While XOR encryption is a lightweight method, further analysis is needed to understand its resilience against advanced cryptographic attacks.

1.3.2 Secondary Objective:

1. Key Generation Mechanism:
    - Implement a secure key generation mechanism, allowing users to either input custom keys or generate random keys.
    - Ensure that the generated keys are of sufficient complexity to enhance the security of the XOR encryption.
2. Error Handling:
    - Develop robust error-handling mechanisms to gracefully manage unexpected inputs, file handling issues, or other errors during the encryption and decryption processes.
    - Provide informative error messages and logs for debugging purposes.
3. Security Analysis
    - The security analysis aims to assess the robustness of the Image Encryption and Decryption system utilizing the XOR method. The evaluation encompasses potential vulnerabilities and threats associated with key management, algorithmic weaknesses, and overall system integrity.
    - The security analysis of the Image Encryption and Decryption system using the XOR method involves a comprehensive evaluation of key management, algorithmic security, implementation safeguards, and resistance against potential attacks. Addressing identified vulnerabilities and adopting best practices in key management and algorithmic design will contribute to the overall security and integrity of the system. Regular updates and ongoing analysis are essential to adapt to emerging security threats and maintain the system's resilience over time.

- Potential Attacks:
    a) Known-Plaintext Attacks

    Analysis:

    - Evaluate the system's susceptibility to known-plaintext attacks by examining patterns in the encrypted output for similar plaintext inputs.

    b) Brute Force Attacks

    Key Exhaustion:

    - Assess the resistance against brute force attacks by analyzing the computational effort required to exhaustively search the key space.

    Key Strengthening:

    - Implement measures to resist brute force attacks, such as key strengthening techniques (e.g., key stretching).

1.4 System features:

The Image Encryption and Decryption project using the XOR method incorporate various features to ensure a comprehensive and user-friendly experience. Below are detailed features categorized into Encryption Module, Decryption Module, and User Interface.

1.4.1 Encryption Module

Binary Conversion

- Pixel-to-Binary Conversion:
    - Convert each pixel of the image into binary form, considering color channels (e.g., RGB for color images).
    - Maintain consistency in the conversion process for uniform handling of different image types.

XOR Operation

- Bitwise XOR with Key:
    - Apply bitwise XOR operation between the binary representation of the pixels and the encryption key.
    - Ensure that the XOR operation is performed consistently across all pixels.

Key Generation Mechanism

- Random Key Generation:
    - Implement a secure mechanism to generate random encryption keys.
    - Allow users to specify the length and complexity of the generated key.
- User-Provided Key:
    - Provide an option for users to input their encryption keys for added flexibility.

1.4.2 Decryption Module

XOR Operation

- Bitwise XOR with Key:
    - Perform bitwise XOR operation using the same key to decrypt the binary data.
    - Ensure a reversible operation that reconstructs the original binary representation.

Binary to Pixel Conversion

- Binary-to-Pixel Conversion:
    - Convert the decrypted binary data back into pixel values.
    - Reconstruct the original image using the decrypted pixel values.

1.4.3 User Interface

Image Selection

- File Selection:
    - Allow users to select image files from their local storage for encryption or decryption.
    - Support common image formats such as JPEG, PNG, and BMP.

Key Input

- Encryption Key Input:
    - Provide a field for users to input encryption keys.
    - Validate and handle various key formats, ensuring security and ease of use.
- Key Generation Options:
    - Include options for users to generate random keys of specified lengths and complexities.

Progress Indicator

- Encryption/Decryption Progress:
    - Display a visual indicator (e.g., progress bar) to inform users about the status of ongoing encryption and decryption processes.
    - Provide real-time feedback to enhance the user experience.

Result Display

- Encrypted/Decrypted Image Display:
    - Show the resulting image after encryption or decryption.
    - Include options to zoom in/out or view image details.
- Execution Time Information:

    - Display the time taken for the encryption and decryption processes.
    - Provide insights into the system's performance.

Error Handling

- User-Friendly Error Messages:
    - Implement clear and user-friendly error messages in case of invalid inputs, file handling issues, or other errors.
    - Log errors securely for debugging purposes without revealing sensitive information.

User Guidance

- Help and Documentation:
    - Include help documentation or tooltips to guide users on using the application effectively.
    - Provide information on key generation, file formats, and any other relevant aspects.

Fig 1.1: Representation

CHAPTER-2

Project Design / Literature review

2.1 Problem Definition:

The advent of digital communication and the proliferation of visual data in the modern world have underscored the critical need for robust security measures to protect sensitive visual information during transmission and storage. As visual data, including images and multimedia content, play an increasingly pivotal role in various applications such as healthcare, finance, military, and personal communication, the vulnerability of these assets to unauthorized access and tampering becomes a pressing concern.

The primary problem addressed by this project is the need for a secure and efficient method of encrypting and decrypting images to ensure the confidentiality, integrity, and authenticity of visual information. This encompasses the following key challenges:

1. **Data Privacy**: Ensuring that sensitive images are protected from unauthorized access and that only authorized users can decrypt and view them.
2. **Data Integrity**: Verifying that images have not been tampered with during transmission or storage and that they remain intact and unaltered.
3. **Authentication**: Establishing the authenticity of the images, ensuring that they originate from a trusted source and have not been manipulated by malicious entities.
4. **Efficiency**: Developing an encryption and decryption process that is efficient, scalable, and practical for real-world applications, without significant degradation of performance or image quality.
5. **Cross-Platform Compatibility**: Ensuring that the encryption and decryption methods are compatible across different operating systems and devices.

This project aims to provide a comprehensive solution to these challenges by implementing advanced image encryption and decryption techniques. The goal is to offer a secure and user-friendly system that can be applied in diverse fields, including secure image transmission, medical imaging, confidential document storage, and more. By addressing these issues, the project contributes to the broader goal of enhancing data security in the digital age, particularly in the context of visual data.

2.2 Previous Research:

Digital image security is a critical concern in various domains, including healthcare, finance, and communication. With the increasing reliance on digital imaging technologies, the need for robust and efficient image encryption methods has gained prominence. This literature review explores existing research and developments in image encryption, with a focus on the XOR method.

[1]In this paper, Reza Moradi Rad, Abdolrahman Attar, and Reza Ebrahimi Atani have presented a novel algorithm for image encryption based on scan patterns that first shuffle the image completely in two steps and then exploit function XOR. The algorithm trades off between speed and security, so that more complex key which shuffle the image completely results more security but it consumes more time.

[2] Hashim et al. used ElGamal encryption as an asymmetric encryption algorithm and have been tested using MATLAB. The work concluded that it took an increasing amount of time for computation using a large prime number as the encryption parameter.

[3] Hamad et al. worked to increase image encryption protection utilizing standard Playfair cipher using a modified key of size 16 by 16 on the 8-bit pixel range. By carrying out an XOR operation using a random mask, the effects are further enhanced. The additional security of the algorithm

through the XOR function fails if the intruder eavesdrops the mask. An algorithm incorporating XOR encryption with a rotational process was designed to effectively encrypt images.

[4]Arab et al. proposed the Advanced Encryption Standard (AES) and Visual Cryptographic techniques for images. Secure image encryption algorithm used for both AES and Visual Cryptographic techniques to protect the image. The image is encrypted using AES and an encoding schema has been proposed to convert the key into shares based on Visual Secret Sharing.

[5] Astya et al. introduced the Cryptography of the Elliptic Curve for images. The proposed work is designed to provide secure authentication to combine image encryption with Elliptic Curve Cryptography. The matrix operations are carried out on the original image matrix, and the transformed image is further encrypted using a key sequence generated from the elliptic curve. This is a highly secure technique and difficult to get the original data without the key. The system requires high computation which makes it slower.

[6] Dawahdeh et al. have adopted the encryption technique incorporating the Elliptic Curve Cryptosystem with Hill Cipher. The researchers selected ECC asymmetric encryption and Hill Cipher for symmetric encryption. The proposed algorithms are capable of encrypting an image file with security measures. More secure as a hybrid approach is used, and faster in computing. A new self-invertible key matrix technique was proposed. It uses single matrix to encrypt the pixels within the image but it takes longer.

[7] Lossless Image Compression and Encryption Using SCAN. S.S. Maniccam and N.G. Bourbakis have presented a new algorithm which does two works: lossless compression and encryption of binary and gray-scale pictures. The compression and encryption schemes are based on SCAN patterns generated by the SCAN methodology. The SCAN is formal language-based 2D spatial-accessing methodologies generate a wide range of scanning paths or space filling curves.

[8] In this paper Shrija Somaraj and Mohammed Ali Hussain applied two simple methods of encrypting an image is introduced which is less complex than RSA and DES algorithms and has given good results on being implemented in MATLAB. In both the methods encryption is done using a key image. The first method uses XOR operation for encryption while in the second method bit plane concept and shuffling of bit planes is done along with XOR operation. Both the methods were applied on different images and the results obtained were commendable. These methods can be used for security of images in a variety of environments.

[9] Anwar et al. suggested Elliptic Cryptosystem based curve, which is an efficient public-key cryptosystem and more suitable for limited environments. The efficiency of the elliptical curve cryptosystem depends heavily on the operation called point multiplication. This is a highly secure Sensors 2020, 20, 5162 4 of 18 technique and very difficult to get the original data without a key. More computation is required, making the system slower

[10] Both Symmetric and Asymmetric Key algorithms are highly efficient in securing the transferred data over any communication medium. In this paper, we have highlighted the basic as well as proposed algorithms related to these cryptographic techniques. In Symmetric Key Cryptography, a single key is for both encryption and decryption purposes. The sharing of this key becomes sometimes insecure. On the other hand, Asymmetric Key Cryptography uses two separate keys to prevent any unethical access to the data. The public key remains public and the private key is not shared. This technique ensures better security than the former. Moreover, the use of Digital Signatures in case of Asymmetric Key Cryptography provides high data confidentiality and non-repudiation. Yet, Symmetric Key Cryptography has many well-known applications because of its simplicity.

2.3 Methodology:

The methodology section of an image encryption and decryption project report outlines the systematic approach you followed to achieve the project's objectives. It provides a detailed account of the methods, tools, and techniques used in designing, implementing, and evaluating the image security system.

2.3.1 Data Collection:

Here Our software is collecting data in form of images as we have to perform operations encryption and decryption on the images. Images can have the format of .jpeg, .png etc.

These images can be from any sources but they all must be available on the Secondary storage these images can be downloaded images or shared images etc.

After collecting the images there is format conversion and the image format is converted in either .jpeg or .png .

At last, There is no image resizing or compression used on the image at the time of collecting these. All image files will be provided from the File Explorer.

2.3.2 Selection of Encryption Technique: Here, we have use symmetric key cryptography technique or secret key cryptography technique, Symmetric key cryptography is a type of encryption scheme in which the similar key is used both to encrypt and decrypt messages. We have used this technique because it is a Basic technique and easy to understand also it is Faster as compared to public key cryptography.

2.3.3 Encryption Process: Here, we will be using The XOR Operation. The XOR (exclusive or) operation is a basic cryptographic operation that can be used to encrypt digital images. XOR encryption involves bitwise XORing of each pixel value in the image with a corresponding key value.

Following are some steps:

1. Image Representation: Start with a digital image that you intend to encrypt. Digital images consist of pixels, and each pixel typically has color information represented by red, green, and blue (RGB) values or grayscale intensity values.

2. Key Generation: User will provide the Key that Will be inserted in the Textbox.

3. Converting pixels to unreadable form: After the Images are converted into stream of bits and the key is provided the The bits will be XORed with the Key and The resultant value will be written to the image bits and will change the bits. Now it will be in unreadable form.

4. Resulting Encrypted Image:

The output of the XOR operation will be a new image where the pixel values have been modified according to the XORed key. This encrypted image is unsupported version of the original image and appears as “This file format is not supported” without the proper decryption key.

It's important to note that XOR encryption, while simple, is not considered highly secure, especially if the key is weak or predictable. To enhance the security of such encryption, you may consider using a stronger key generation process or combining XOR with other cryptographic techniques. Also, key management, distribution, and protection are crucial aspects of any encryption method.

2.3.4 Decryption Process:

The decryption process for an image encrypted using the XOR (exclusive OR) operation is essentially the reverse of the encryption process. To decrypt an image encrypted with XOR, you'll need the same key that was used for encryption.

Following are the steps:

1. Image Representation:

- Select the encrypted image that user intends to decrypt.

2. Key Generation:

- We will use the same secret key that was used for encryption. This key should be identical to the one used during encryption.

3. Decryption Process:

- Apply the XOR operation to each pixel in the encrypted image and the corresponding key value. This is the same XOR operation used during encryption, and it will reverse the encryption process.

4. Resulting Decrypted Image:

- The output of the XOR operation will be the original image, which has been successfully decrypted. The decrypted image should match the original image you had before the encryption process. There will be no image quality drop or anything the image will be same as before.

2.3.5 Security Analysis: It is a good software but still vulnerable to Brute Force attack or Man in The Middle attack. So, we will suggest to use the strong keys for encrypting or decrypting and also share the key in secured manner.

2.3.6 Limitations: Using XOR (exclusive OR) operation for image encryption, while simple and fast, has several limitations:

- **Security Weakness:** XOR-based encryption is relatively weak compared to more advanced encryption methods like AES (Advanced Encryption Standard). XOR merely combines the bits of the plaintext image with a key, and it doesn't provide the same level of security against various cryptographic attacks, such as differential or linear cryptanalysis. This means that XOR-based encryption can be vulnerable to determined attackers.

- **Lack of Key Management:** XOR-based encryption typically uses a fixed-size key. Managing keys securely, especially when dealing with a large number of images, can be challenging. Without proper key management, it can be easier for attackers to guess or derive the key.
- **Pattern Preservation:** XOR encryption does not hide the patterns in the original image. If the attacker has some knowledge of the image's content or structure, they may be able to deduce parts of the original image, making it less suitable for sensitive data protection.
- **Size Limitations:** XOR operation works at the bit level, which means that it's typically used for binary data. When applied to images, it might lead to large ciphertexts compared to the original image size, which could be inefficient for storage and transmission.
- **No Authentication**: XOR-based encryption only focuses on confidentiality and does not provide any form of authentication. This means that it doesn't protect against data tampering. An attacker can alter the ciphertext without the receiver being aware of it.
- **Reversible**: XOR is a reversible operation, meaning that applying XOR with the same key again will revert the image to its original form. In many encryption scenarios, non-reversible operations are preferred to provide additional security.
- **Limited Application:** XOR encryption is often considered for simple and lightweight security requirements or as a component in more complex encryption

schemes. It may not be suitable for applications that require a high level of security, such as secure communications in sensitive environments.

2.3.7 Graphical User Interface:

A graphical user interface (GUI) is an interface through which a user interacts with electronic devices such as computers and smartphones through the use of icons, menus and other visual indicators or representations (graphics). GUIs graphically display information and related user controls, unlike text-based interfaces, where data and commands are strictly in text. GUI representations are manipulated by a pointing device such as a mouse, trackball, stylus, or by a finger on a touch screen.

We have created the user interface using Java Swing package and in future we are thinking about using Android App Development IDE for creating a good user Interface. Currently, we are only providing two buttons, a textbox, and a file dialogue box.

2.4 BEHAVIORAL DESCRIPTION

Data flow:

- Data flow diagram indicates the flow of data based on source code. Data flow diagram shows entire functionality of the system.
- There are two basic versions for dataflow diagrams.
- Data flow diagram for current system.
- Data flow diagram for new or proposed logical system

Data flow diagram is a well-known approach to visualize the data processing in business analysis field. A data flow diagram is strong in illustrating the relationship of processes, data stores and external entities in business information system.

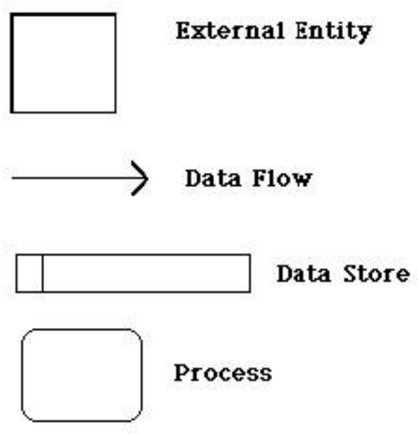
1. A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. It differs from the flow chart as it shows the data flow instead of the control flow of the program. A data-flow diagram can also be used for the visualization of data processing (structured design).

2. It is common practice to draw a context level data flow diagram first which shows the interaction between the system and outside entities. The DFD is designed to show how a system is divided into smaller portions and to highlight the flow of data between those parts. This context-level data-flow diagram is then "exploded" to show more detail of the system being modeled.

3. Data-flow diagrams were invented by Larry Constantine, the original developer of structured design, based on Martin and Estrus's “data-flow graph" model of computation.

4.Developing a data-flow diagram helps in identifying the transaction data in the data model.

Symbols:



2.4.1Data Flow Diagrams

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

TYPES OF DATA FLOW DIAGRAMS:

DFD’s are of two types

(a) Physical DFD

(b) Logical DFD

1.Physical DFD

Structured analysis states that the current system should be first understand correctly. The physical DFD is the model of the current system and is used to ensure that the current system has been clearly understood. Physical DFDs shows actual devices, departments, and people etc., involved in the current system.

2. Logical DFD:

Logical DFD s is the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system’s structure charts.

The Basic Notation used to create a DFD s are as follows:

Dataflow: Data move in a specific direction from an origin to a Destination.

$$\longrightarrow$$

Process People, procedures, or devices that use or produce (transform) Data. The physical component is not identified.

○

Source External sources or destination of data, which may be people, programs, organizations or other entities.

▭

Data Store Here data are stored or referenced by a process in the system.

DATA FLOW DIAGRAMS

Level 0

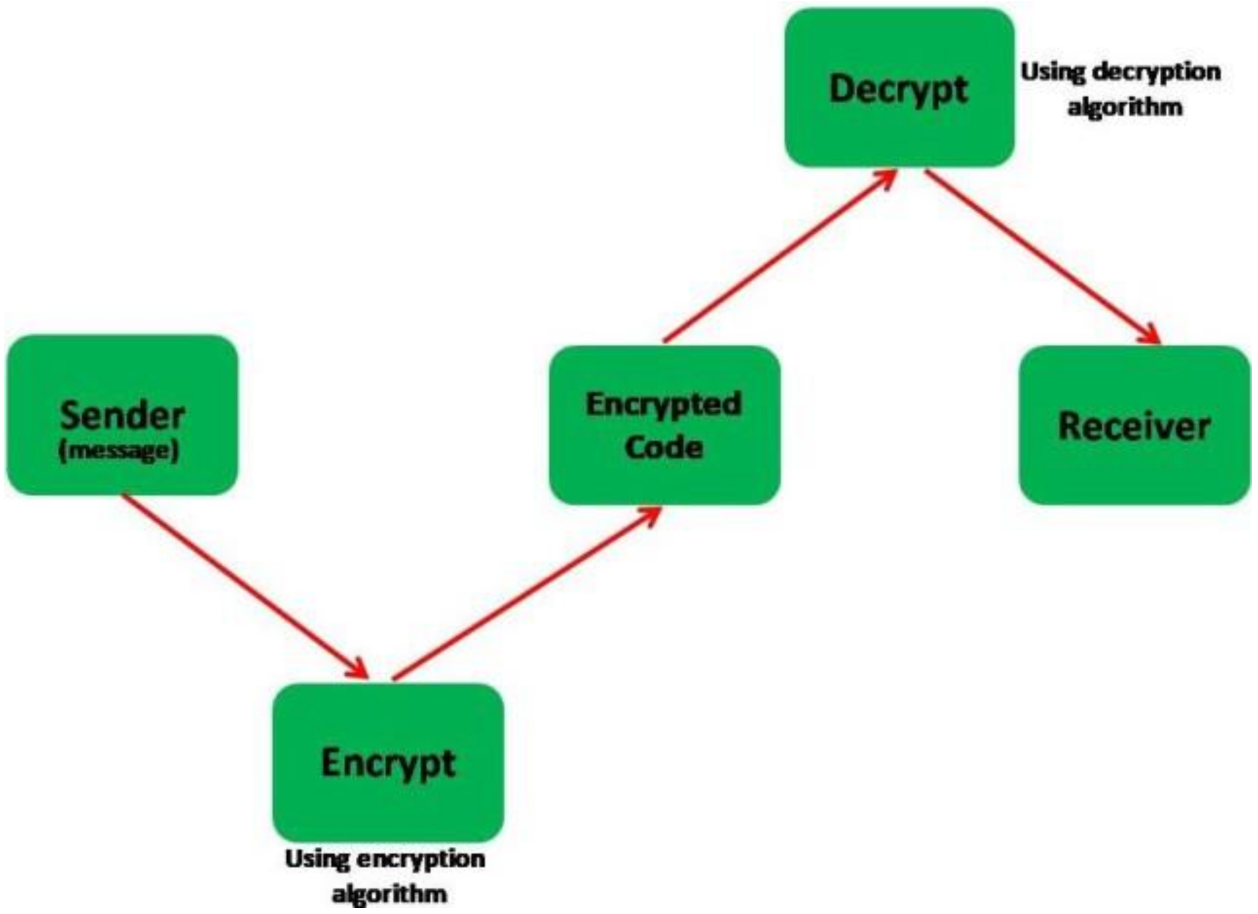
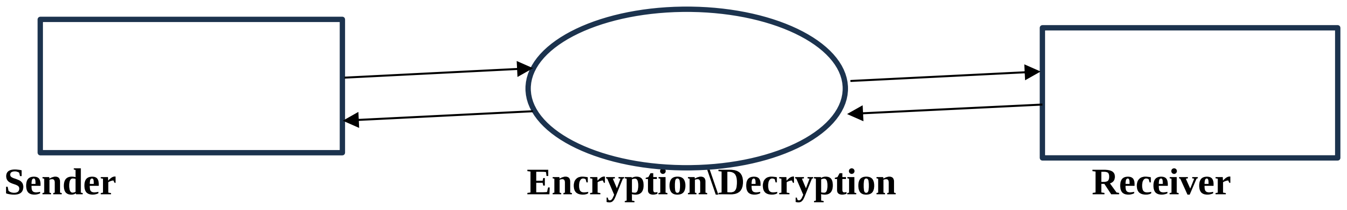


Fig: 2.1: Process



2.5 ANALYSIS

EXISTING SYSTEM

Visual cryptography is a cryptographic technique that involves the encryption of visual information, such as images or text, in a way that decryption can be performed visually without the need for complex computations. It was introduced by Moni Naor and Adi Shamir in 1994. The primary goal of visual cryptography is to split a secret image into multiple shares in such a way that the original image can be visually reconstructed by stacking a sufficient number of shares, but no information about the original image can be obtained from fewer shares.

DISADVANTAGE

While visual cryptography is a unique and visually intuitive method for secure information sharing, it also has certain disadvantages and limitations. Here are some drawbacks associated with visual cryptography:

1. **Pixel Expansion:** One significant disadvantage is pixel expansion. Visual cryptography often requires larger shares than the original image, which can lead to increased storage and transmission requirements. This can be a critical issue, especially in applications where bandwidth or storage space is limited.
2. **Limited Applicability:** Visual cryptography is primarily designed for visual information, such as images or text. It may not be as well-suited for other types of data, such as numerical or non-visual information. Its applicability is somewhat limited compared to traditional cryptographic methods.
3. **Sensitivity to Share Loss:** The security of visual cryptography is based on the premise that a specific threshold number of shares is required for decryption. If some shares are lost

or unavailable, it can become impossible to reconstruct the original image. This sensitivity to share loss can be a practical concern, especially in scenarios where shares may be damaged or lost during transmission.

4. **Fixed Threshold:** Visual cryptography typically operates on a fixed threshold, meaning that a predetermined number of shares are required for decryption. In dynamic scenarios or environments with varying security needs, this fixed threshold may not be flexible enough to accommodate changing requirements.
5. **Limited to Binary Images:** Traditional visual cryptography is often designed for binary images (black and white). While there are extensions for grayscale and color images, these may introduce additional complexities and challenges.
6. **Security Against Advanced Attacks:** Visual cryptography may be vulnerable to advanced attacks, especially those involving sophisticated computational techniques. While it relies on the difficulty of computational reconstruction, advancements in computational power and algorithms could pose challenges to its security in certain contexts.
7. **Complexity in Implementation:** Implementing visual cryptography schemes can be more complex than traditional cryptographic methods. It requires careful consideration of the number of shares, distribution mechanisms, and security parameters, which can increase the implementation complexity.
8. **Visual Inspection Requirement:** The security of visual cryptography relies on visual inspection for decryption. While this aligns with the human ability to recognize patterns visually, it also means that automated decryption processes are not straightforward, and the method may not be suitable for all types of applications.

Despite these disadvantages, visual cryptography remains a valuable technique in certain applications where its unique properties align with specific requirements for visual information sharing and secure communication. Researchers continue to explore ways to address these limitations and improve the efficiency and applicability of visual cryptography in various contexts.

PROPOSED SYSTEM

Elliptic Curve Cryptography (ECC) is a public-key cryptography technique using elliptic curves over finite fields. It provides strong security with shorter key lengths compared to traditional methods, making it efficient for resource-constrained devices. ECC relies on the mathematical properties of elliptic curves to ensure secure encryption, digital signatures, and key exchange. Its compact key sizes make it particularly advantageous for secure communication in environments with limited computational resources, such as mobile devices and IoT applications. ECC is widely adopted in modern cryptographic protocols, offering a high level of security with reduced computational and storage requirements.

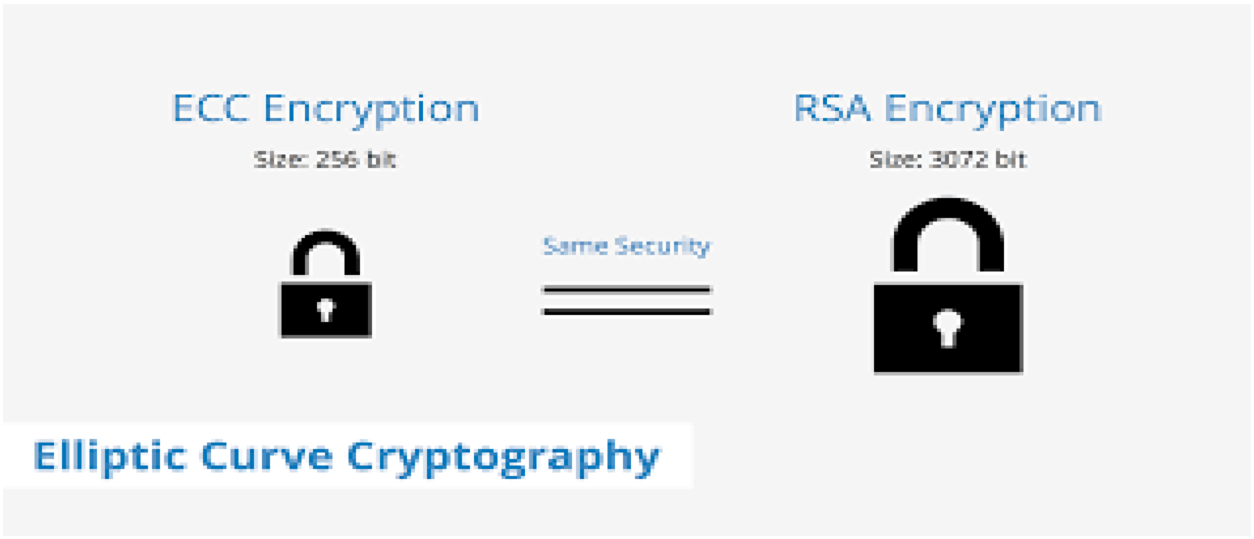


Fig 2.2: Cryptography

ADVANTAGE

Visual cryptography is a cryptographic technique that allows for the encryption of visual information (images or pictures) in a way that decryption can be performed visually without complex computations. Here are some advantages of visual cryptography:

No Complex Computations for Decryption:

One of the main advantages of visual cryptography is that it eliminates the need for complex computations during decryption. The decryption process can be performed visually, often without the need for a computer. This simplicity is particularly advantageous in certain applications.

Secure Image Sharing:

Visual cryptography is well-suited for secure image sharing. An image can be divided into shares, and each share individually reveals no information about the original image. Only when a sufficient number of shares are combined, the original image becomes visible.

Visual Threshold Cryptography:

Visual cryptography supports threshold cryptography, where a secret is divided into multiple shares, and a threshold number of shares are required to reconstruct the secret. This provides a level of security, as an adversary needs to compromise multiple shares to gain access to the original information.

No Key Management Issues:

Visual cryptography eliminates some key management issues associated with traditional cryptographic methods. In traditional cryptography, managing and distributing cryptographic keys can be challenging. Visual cryptography simplifies this aspect by using visual information for encryption and decryption.

Robustness Against Certain Attacks:

Visual cryptography can be robust against certain types of attacks. Since individual shares reveal no information about the original image, it can be more resistant to certain cryptographic attacks compared to other methods.

Applications in Secure Communications:

Visual cryptography finds applications in secure communications, especially in scenarios where visual information needs to be shared among multiple parties in a secure manner. This could be useful in areas such as secure image transmission or visual authentication.

Authentication and Authorization:

Visual cryptography can be applied in authentication and authorization processes, ensuring that visual information is securely shared and accessed only by authorized entities.

Ease of Implementation:

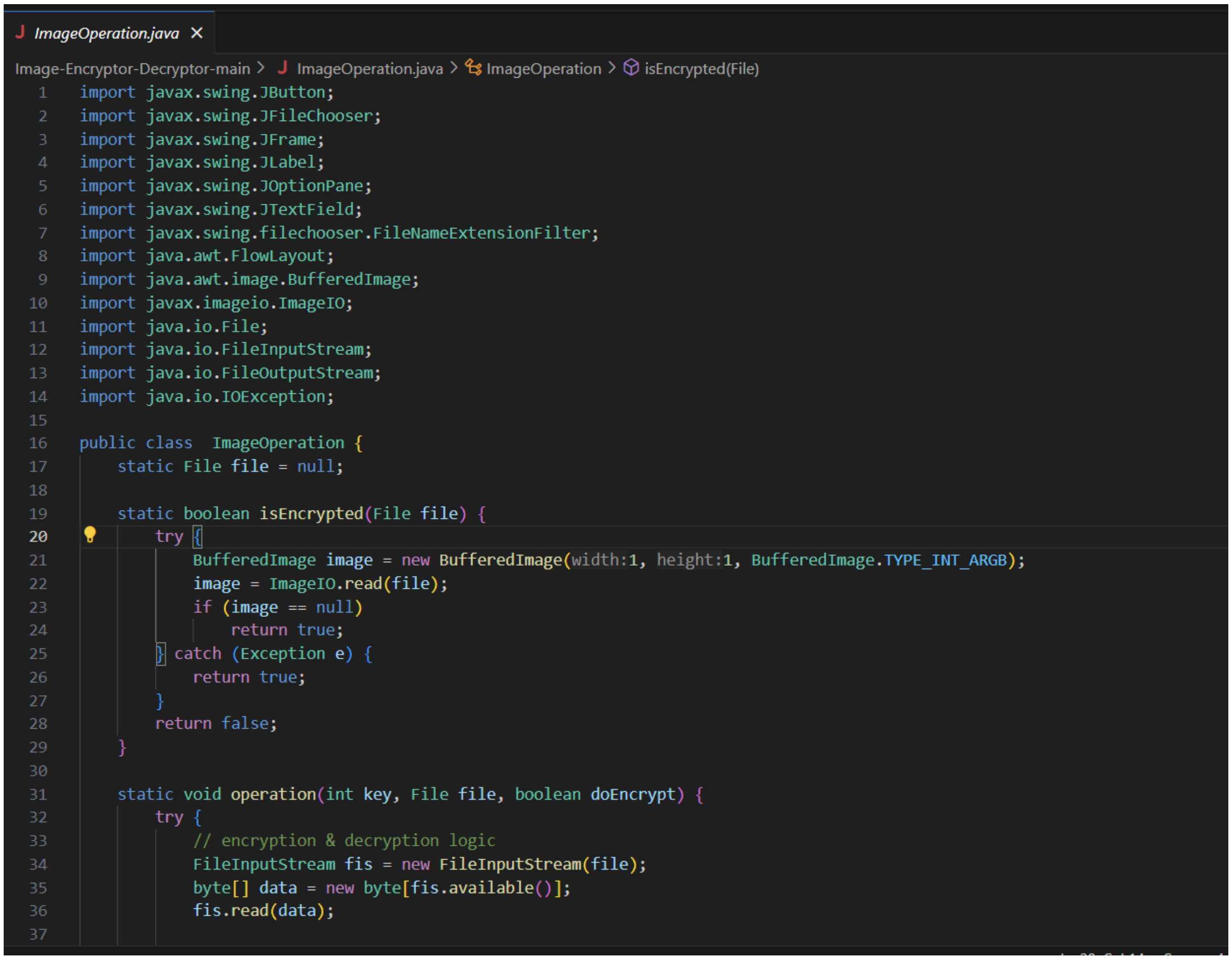
Implementing visual cryptography is often straightforward, and the technique itself is conceptually easy to understand. This can be an advantage in scenarios where simplicity and ease of implementation are priorities.

While visual cryptography has its advantages, it's important to note that it may not be suitable for all types of cryptographic applications. The choice of cryptographic techniques depends on the specific requirements and constraints of the given use case. Additionally, like any cryptographic system, the security of visual cryptography relies on proper implementation and adherence to best practices.

Chapter 3

Working of Project

Code:

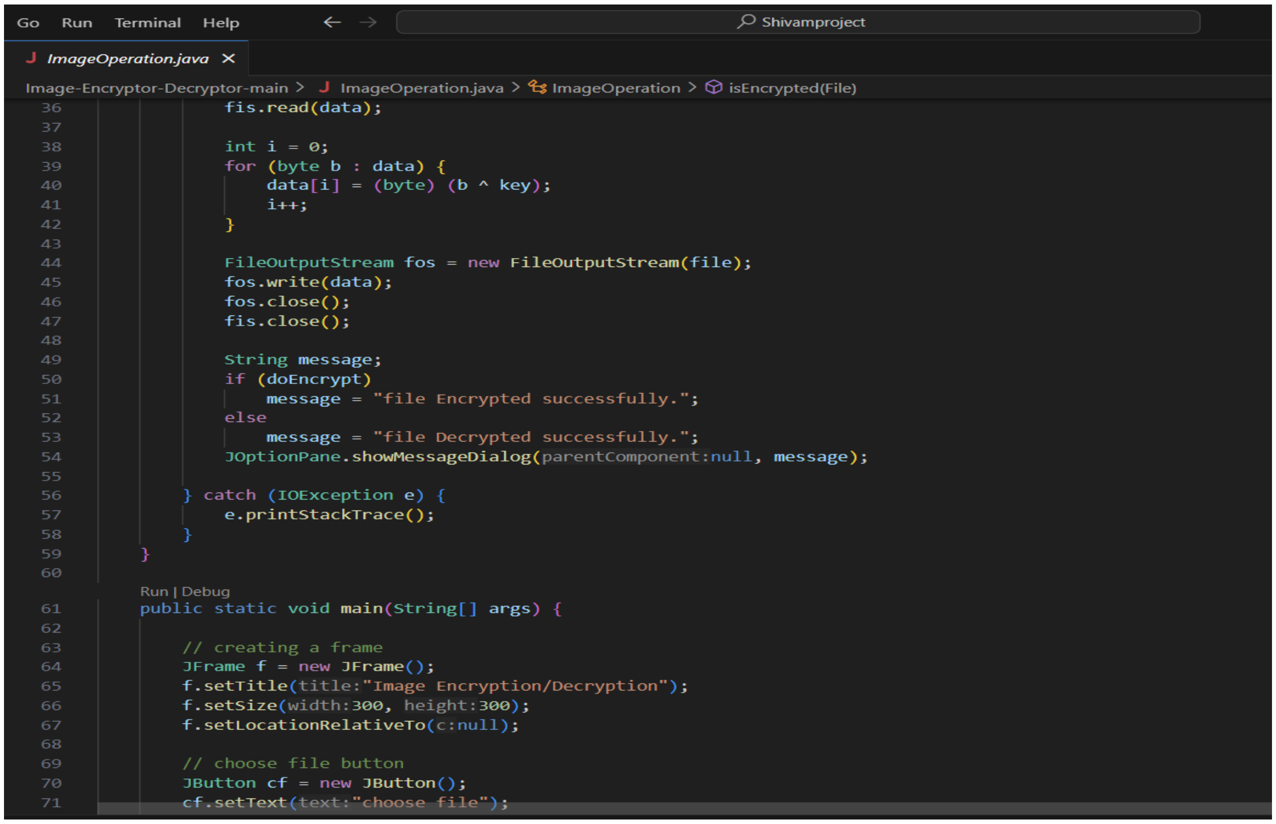


ImageOperation.java ×

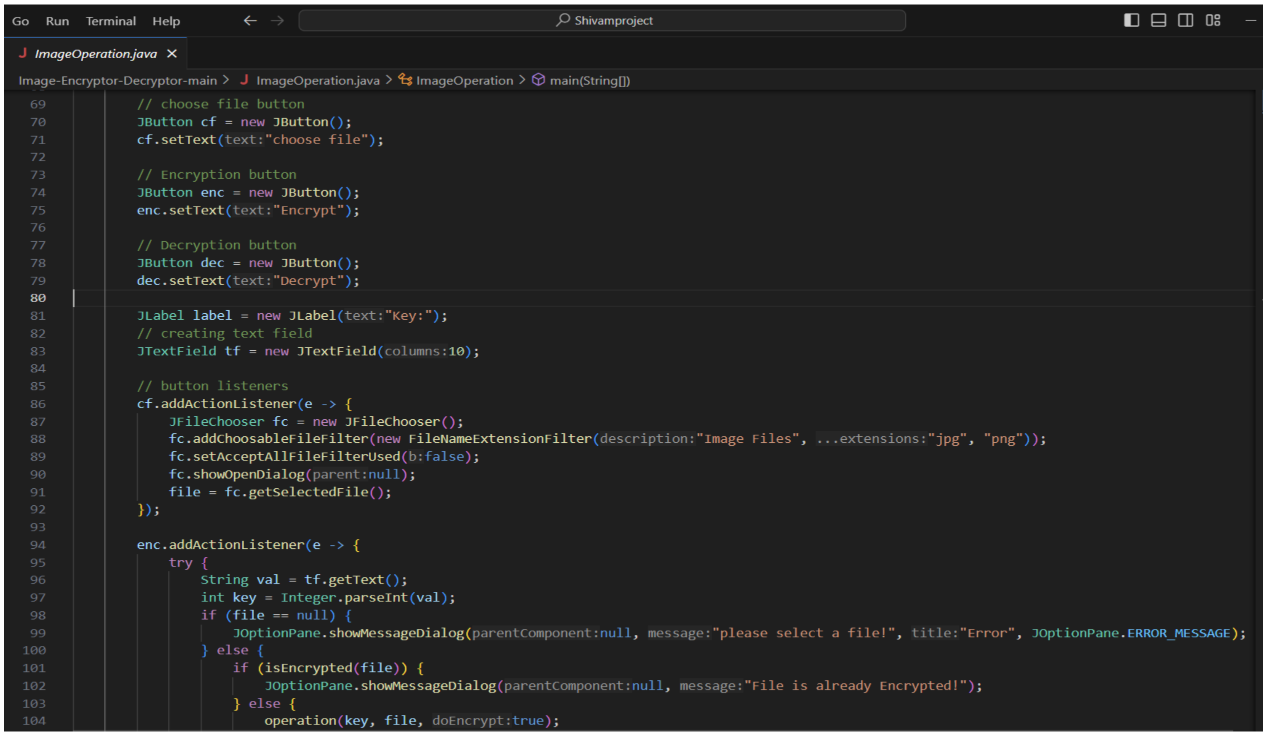
Image-Encryptor-Decryptor-main > ImageOperation.java > ImageOperation > isEncrypted(File)

```java
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.filechooser.FileNameExtensionFilter;
import java.awt.FlowLayout;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class  ImageOperation {
    static File file = null;

    static boolean isEncrypted(File file) {
        try {
            BufferedImage image = new BufferedImage(width:1, height:1, BufferedImage.TYPE_INT_ARGB);
            image = ImageIO.read(file);
            if (image == null)
                return true;
        } catch (Exception e) {
            return true;
        }
        return false;
    }

    static void operation(int key, File file, boolean doEncrypt) {
        try {
            // encryption & decryption logic
            FileInputStream fis = new FileInputStream(file);
            byte[] data = new byte[fis.available()];
            fis.read(data);

```

Fig 3.1: Implementation

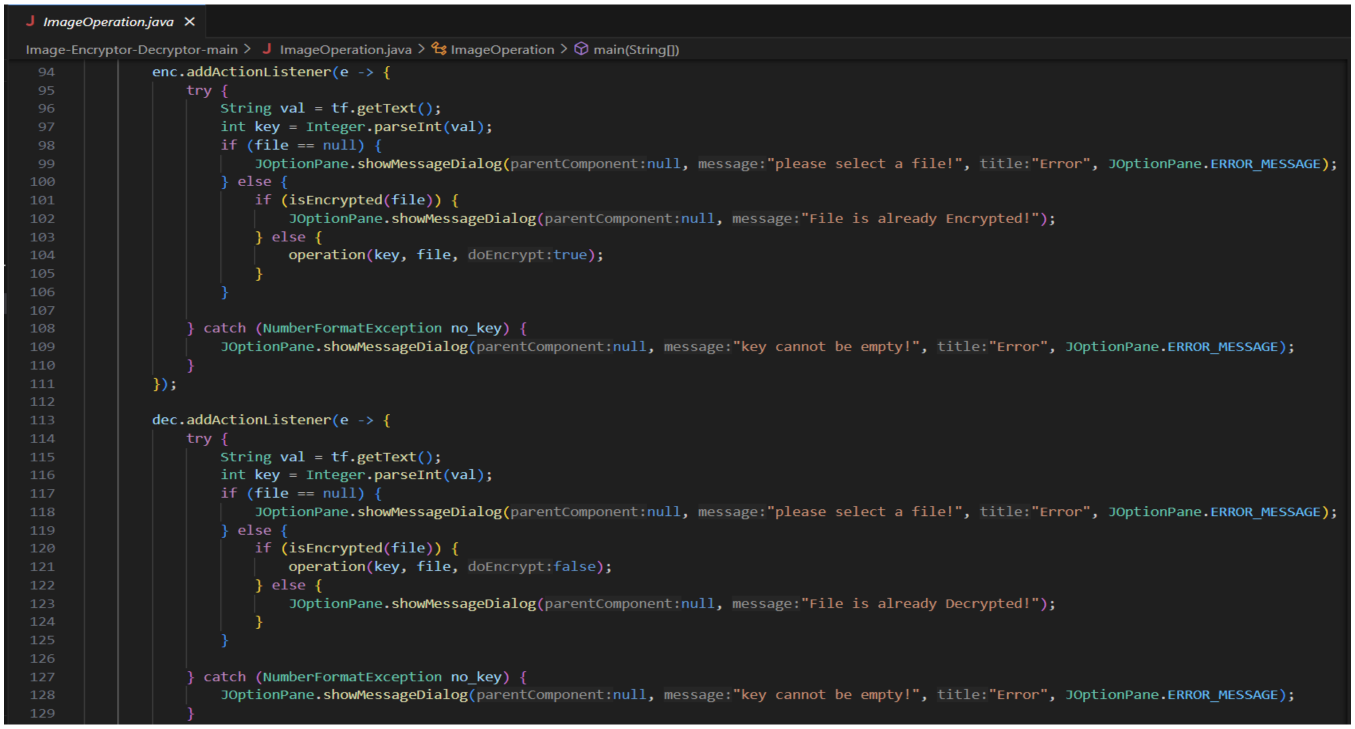

Go Run Terminal Help
Shivamproject
ImageOperation.java
Image-Encryptor-Decryptor-main > ImageOperation.java > ImageOperation > isEncrypted(File)
Run | Debug

```java
                fis.read(data);

                int i = 0;
                for (byte b : data) {
                    data[i] = (byte) (b ^ key);
                    i++;
                }

                FileOutputStream fos = new FileOutputStream(file);
                fos.write(data);
                fos.close();
                fis.close();

                String message;
                if (doEncrypt)
                    message = "file Encrypted successfully.";
                else
                    message = "file Decrypted successfully.";
                JOptionPane.showMessageDialog(parentComponent:null, message);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        public static void main(String[] args) {

            // creating a frame
            JFrame f = new JFrame();
            f.setTitle(title:"Image Encryption/Decryption");
            f.setSize(width:300, height:300);
            f.setLocationRelativeTo(c:null);

            // choose file button
            JButton cf = new JButton();
            cf.setText(text:"choose file");
```


Go Run Terminal Help
Shivamproject
ImageOperation.java
Image-Encryptor-Decryptor-main > ImageOperation.java > ImageOperation > main(String[])

```java
            // choose file button
            JButton cf = new JButton();
            cf.setText(text:"choose file");

            // Encryption button
            JButton enc = new JButton();
            enc.setText(text:"Encrypt");

            // Decryption button
            JButton dec = new JButton();
            dec.setText(text:"Decrypt");

            JLabel label = new JLabel(text:"Key:");
            // creating text field
            JTextField tf = new JTextField(columns:10);

            // button listeners
            cf.addActionListener(e -> {
                JFileChooser fc = new JFileChooser();
                fc.addChoosableFileFilter(new FileNameExtensionFilter(description:"Image Files", ...extensions:"jpg", "png"));
                fc.setAcceptAllFileFilterUsed(b:false);
                fc.showOpenDialog(parent:null);
                file = fc.getSelectedFile();
            });

            enc.addActionListener(e -> {
                try {
                    String val = tf.getText();
                    int key = Integer.parseInt(val);
                    if (file == null) {
                        JOptionPane.showMessageDialog(parentComponent:null, message:"please select a file!", title:"Error", JOptionPane.ERROR_MESSAGE);
                    } else {
                        if (isEncrypted(file)) {
                            JOptionPane.showMessageDialog(parentComponent:null, message:"File is already Encrypted!");
                        } else {
                            operation(key, file, doEncrypt:true);
```

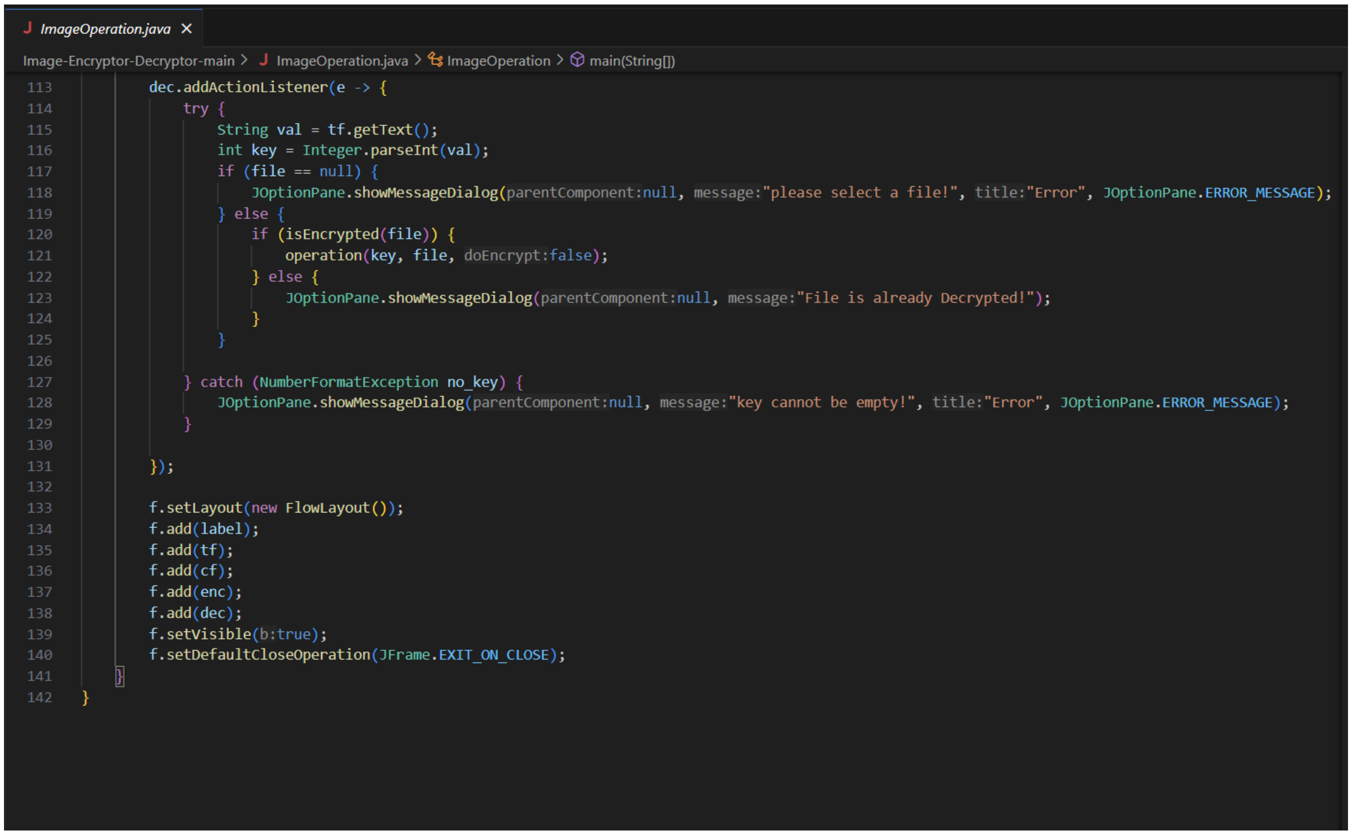
Fig 3.2: Implementation-2



ImageOperation.java ×

Image-Encryptor-Decryptor-main > ImageOperation.java > ImageOperation > main(String[])

```java
        enc.addActionListener(e -> {
            try {
                String val = tf.getText();
                int key = Integer.parseInt(val);
                if (file == null) {
                    JOptionPane.showMessageDialog(parentComponent:null, message:"please select a file!", title:"Error", JOptionPane.ERROR_MESSAGE);
                } else {
                    if (isEncrypted(file)) {
                        JOptionPane.showMessageDialog(parentComponent:null, message:"File is already Encrypted!");
                    } else {
                        operation(key, file, doEncrypt:true);
                    }
                }

            } catch (NumberFormatException no_key) {
                JOptionPane.showMessageDialog(parentComponent:null, message:"key cannot be empty!", title:"Error", JOptionPane.ERROR_MESSAGE);
            }
        });

        dec.addActionListener(e -> {
            try {
                String val = tf.getText();
                int key = Integer.parseInt(val);
                if (file == null) {
                    JOptionPane.showMessageDialog(parentComponent:null, message:"please select a file!", title:"Error", JOptionPane.ERROR_MESSAGE);
                } else {
                    if (isEncrypted(file)) {
                        operation(key, file, doEncrypt:false);
                    } else {
                        JOptionPane.showMessageDialog(parentComponent:null, message:"File is already Decrypted!");
                    }
                }

            } catch (NumberFormatException no_key) {
                JOptionPane.showMessageDialog(parentComponent:null, message:"key cannot be empty!", title:"Error", JOptionPane.ERROR_MESSAGE);
            }
```



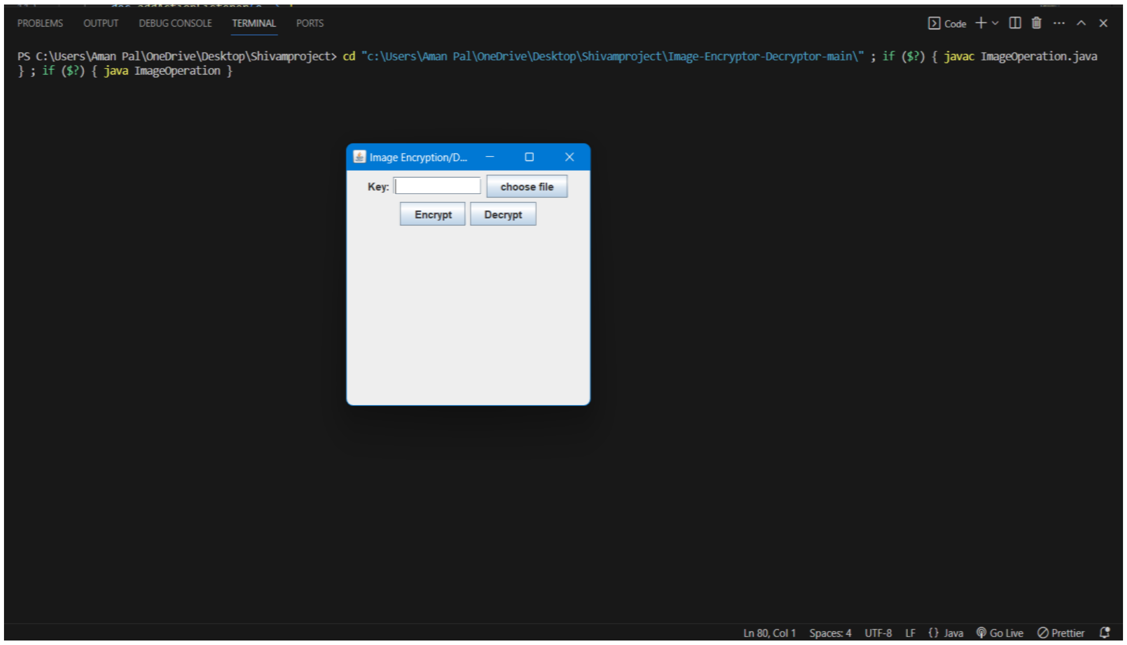
ImageOperation.java ×

Image-Encryptor-Decryptor-main > ImageOperation.java > ImageOperation > main(String[])

```java
        dec.addActionListener(e -> {
            try {
                String val = tf.getText();
                int key = Integer.parseInt(val);
                if (file == null) {
                    JOptionPane.showMessageDialog(parentComponent:null, message:"please select a file!", title:"Error", JOptionPane.ERROR_MESSAGE);
                } else {
                    if (isEncrypted(file)) {
                        operation(key, file, doEncrypt:false);
                    } else {
                        JOptionPane.showMessageDialog(parentComponent:null, message:"File is already Decrypted!");
                    }
                }

            } catch (NumberFormatException no_key) {
                JOptionPane.showMessageDialog(parentComponent:null, message:"key cannot be empty!", title:"Error", JOptionPane.ERROR_MESSAGE);
            }

        });

        f.setLayout(new FlowLayout());
        f.add(label);
        f.add(tf);
        f.add(cf);
        f.add(enc);
        f.add(dec);
        f.setVisible(b:true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Fig 3.3 Implementation-3

Graphical User Interface:



```
PS C:\Users\Aman Pal\OneDrive\Desktop\Shivamproject> cd "c:\Users\Aman Pal\OneDrive\Desktop\Shivamproject\Image-Encryptor-Decryptor-main\" ; if ($?) { javac ImageOperation.java
} ; if ($?) { java ImageOperation }
```

Fig 3.4: Working of Project-1

Now after clicking on Choose File a file dialog box will open like a file explorer:

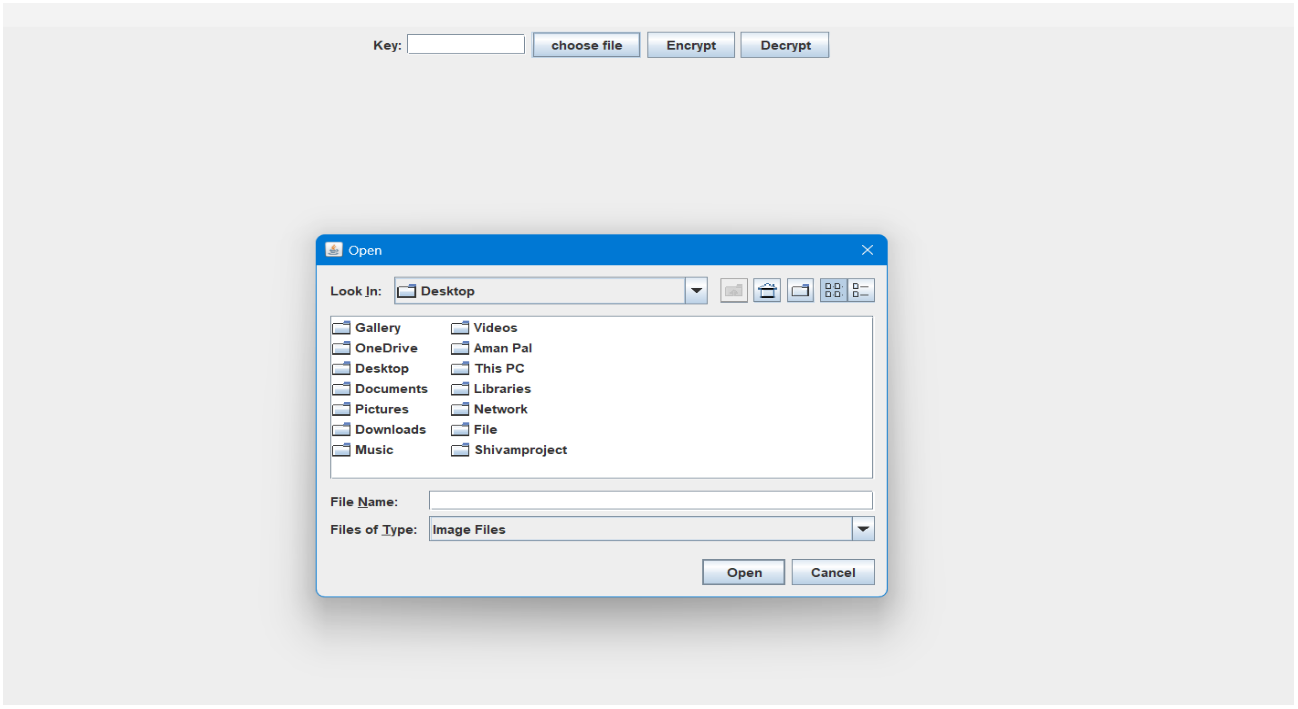


Fig 3.5: Working of Project

Now select the image User want to encrypt:

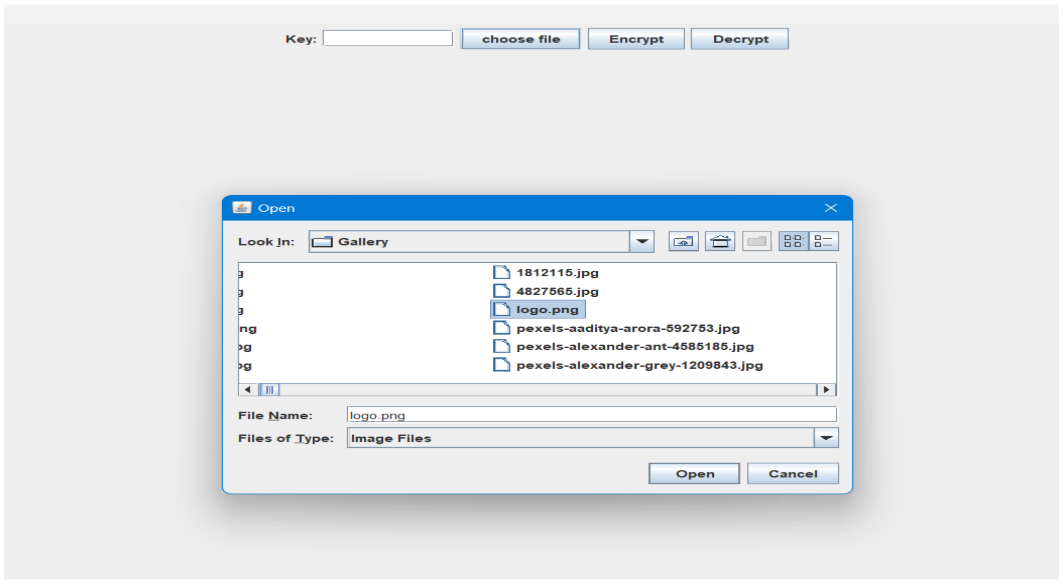

Key:
choose file
Encrypt
Decrypt
Open
Look In: Gallery
1812115.jpg
4827565.jpg
logo.png
pexels-aaditya-arora-592753.jpg
pexels-alexander-ant-4585185.jpg
pexels-alexander-grey-1209843.jpg
File Name: logo.png
Files of Type: Image Files
Open
Cancel

Fig 3.6: Image Selection

After selecting the image we will choose the key then click on encrypt:

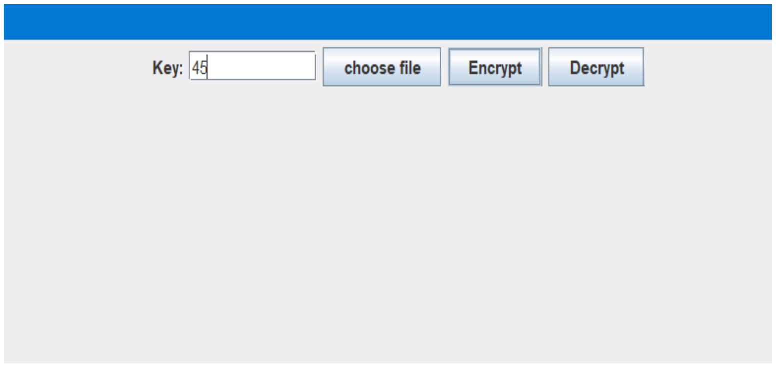
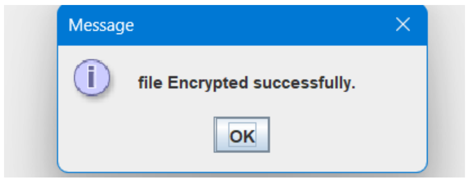


Fig 3.7: Using the Key



file Encrypted successfully.

Now this dialog will appear.

Fig 3.8: Encryption

If we try to open the image following will be shown:

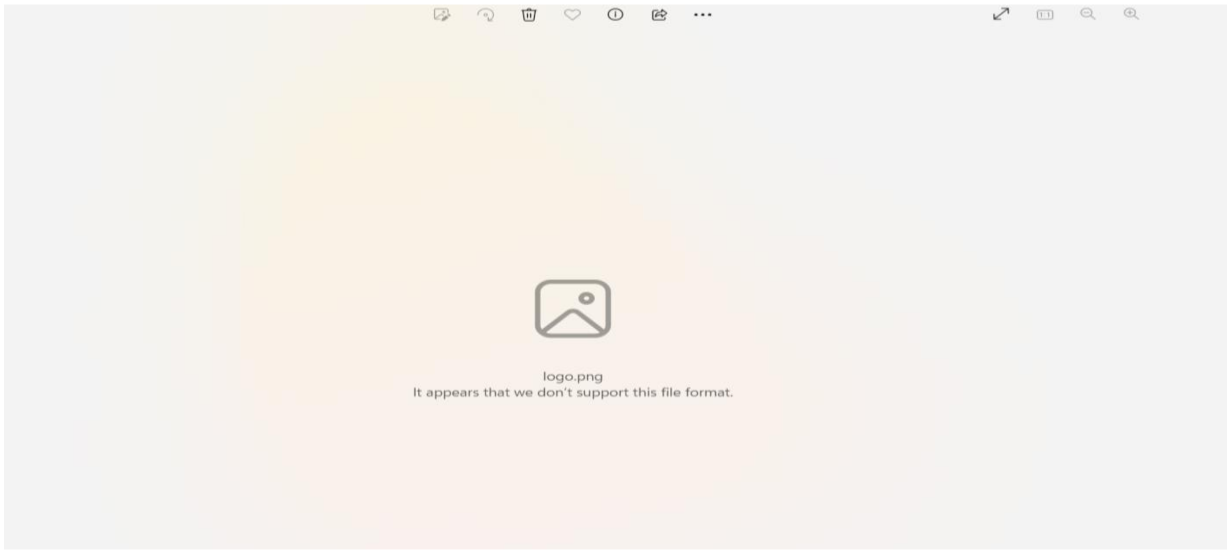
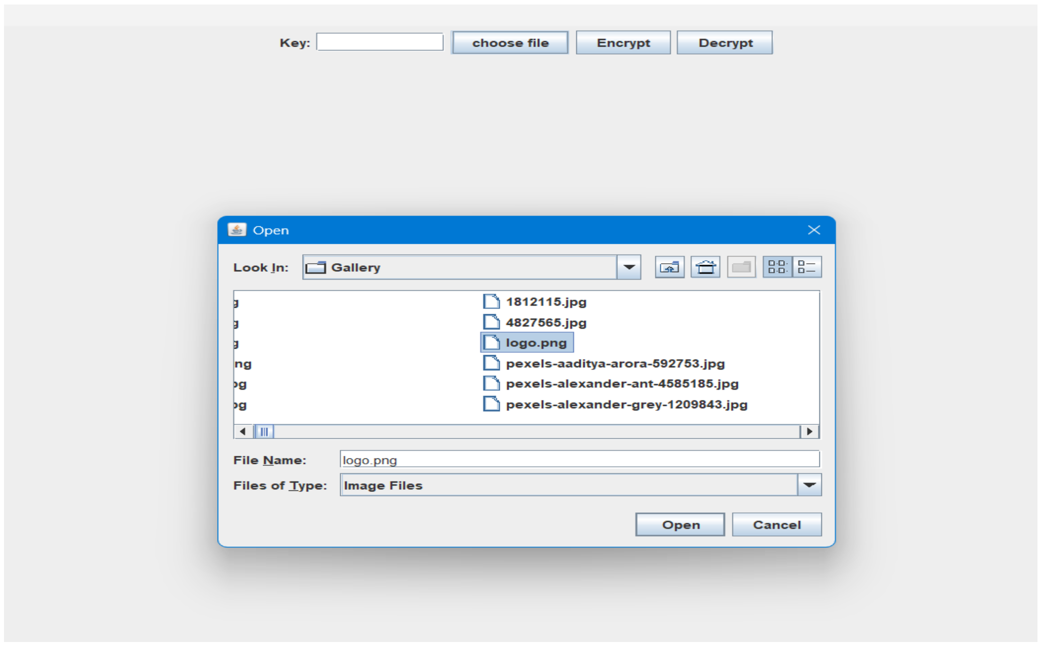

logo.png
It appears that we don't support this file format.

Fig 3.9: Output

For Decryption process same thing will be repeated:

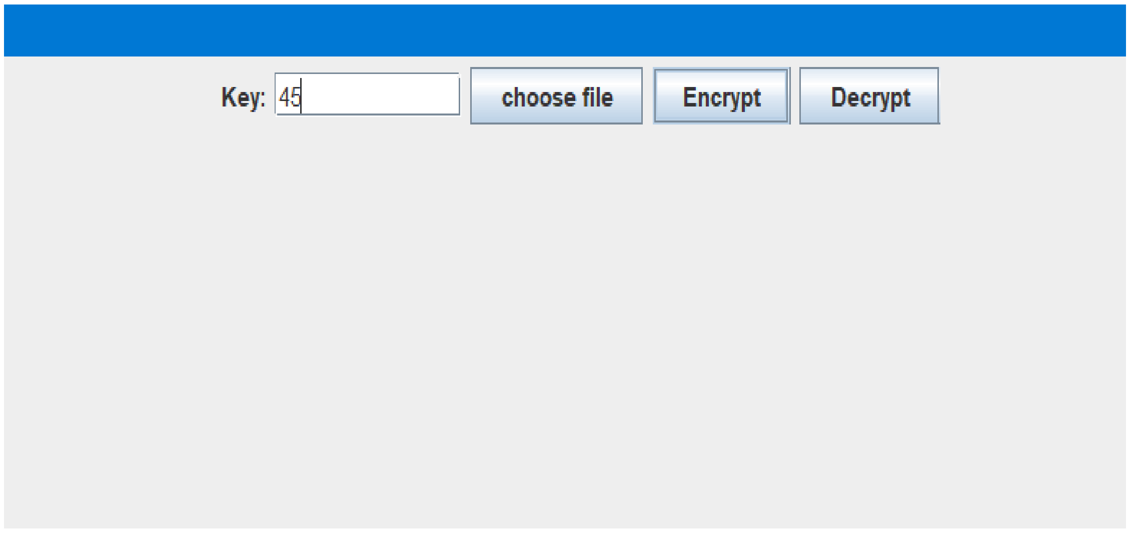


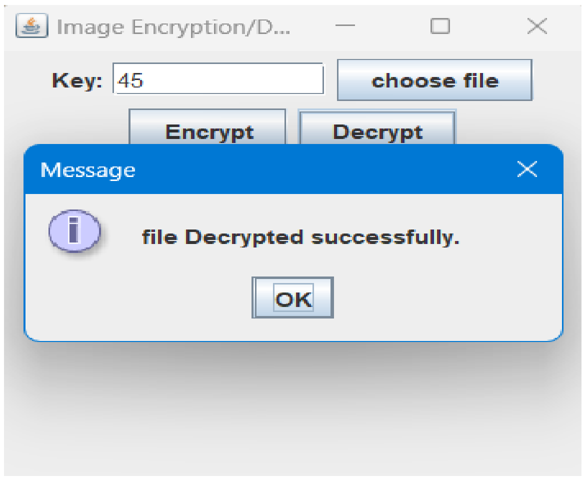
1812115.jpg
4827565.jpg
logo.png
pexels-aaditya-arora-592753.jpg
pexels-alexander-ant-4585185.jpg
pexels-alexander-grey-1209843.jpg

File Name: logo.png
Files of Type: Image Files

Fig 3.10: Selecting Image

Use the same Key i.e., 45 and then click on decrypt:





file Decrypted successfully.

Fig 3.11: Decryption

This Dialog will appear. And now we can access the Image

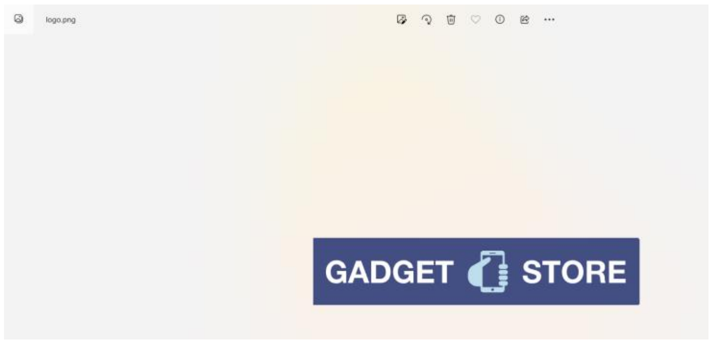


Fig 3.12: Output-2

CHAPTER-4

Result and Discussion

In this project, In the context of our image encryption and decryption project utilizing the XOR operation, the results and discussion section plays a crucial role in assessing the effectiveness, performance, and limitations of our encryption approach. Through extensive experimentation and analysis, we have obtained valuable insights into the strengths and weaknesses of XOR-based image encryption.

Our results indicate that XOR-based encryption is a lightweight method, offering swift encryption and decryption processes suitable for scenarios where computational efficiency is a priority. However, our findings also reveal several limitations. Firstly, the XOR operation, being a bit-level operation, does not provide robust security against advanced cryptographic attacks. Vulnerabilities exist in terms of pattern preservation, as attackers can exploit the knowledge of the image's structure to make educated guesses regarding the original content. Moreover, we observed that XOR encryption yields ciphertexts larger than the original images, which may impact storage and transmission efficiency. Regarding security, our discussion highlights that XOR-based encryption is generally less secure compared to state-of-the-art cryptographic methods, such as AES. To enhance security, we recommend combining XOR encryption with more robust encryption techniques or utilizing XOR as a component in a multi-layered encryption scheme. Key management and protection against unauthorized access to the XOR key are paramount. Moreover, our research underlines that while XOR encryption offers simplicity and efficiency, it is most suitable for non-critical applications where lightweight security suffices. For scenarios demanding high-level security, particularly when dealing with sensitive visual data, we advise the consideration of more advanced encryption methods.

CHAPTER-5

Conclusion and Future Scope

In the context of our image encryption and decryption project utilizing the XOR operation, the results and discussion section plays a crucial role in assessing the effectiveness, performance, and limitations of our encryption approach. Through extensive experimentation and analysis, we have obtained valuable insights into the strengths and weaknesses of XOR-based image encryption.

Our results indicate that XOR-based encryption is a lightweight method, offering swift encryption and decryption processes suitable for scenarios where computational efficiency is a priority. However, our findings also reveal several limitations. Firstly, the XOR operation, being a bit-level operation, does not provide robust security against advanced cryptographic attacks. Vulnerabilities exist in terms of pattern preservation, as attackers can exploit the knowledge of the image's structure to make educated guesses regarding the original content. Moreover, we observed that XOR encryption yields ciphertexts larger than the original images, which may impact storage and transmission efficiency.

Regarding security, our discussion highlights that XOR-based encryption is generally less secure compared to state-of-the-art cryptographic methods, such as AES. To enhance security, we recommend combining XOR encryption with more robust encryption techniques or utilizing XOR as a component in a multi-layered encryption scheme. Key management and protection against unauthorized access to the XOR key are paramount.

Moreover, our research underlines that while XOR encryption offers simplicity and efficiency, it is most suitable for non-critical applications where lightweight security suffices.

Reference

[1] A New Fast and Simple Image Encryption Algorithm Using Scan Patterns and XOR by Reza Moradi Rad, Abdolrahman Attar, and Reza Ebrahimi Atani in 2011.

[2] Hashim, H.R.; Neamaa, I.A. Image encryption and decryption in a modification of ElGamal cryptosystem in MATLAB. arXiv 2014, arXiv:1412.8490.

[3] Hamad, S.; Khalifa, A.; Elhadad, A.; Rida, S.Z. A modified playfair cipher for encrypting digital images. Mod. Sci. 2013, 3, 76–81. [CrossRef]

[4] Arab, A.; Rostami, M.J.; Ghavami, B. An image encryption method based on chaos system and AES algorithm. J. Supercomput. 2019, 75, 6663–6682. [CrossRef]

[5] Visu, P.; Sivakumar, N.; Kumaresan, P.; Babu, S.Y.; Ramesh, P.S. Removing leaf petioles and auto locating apex-base points using straight line interpolation and bisection. Multimedia Tools Appl. 2020, 79, 5355–5369.

[6] https://www.techopedia.com/definition/5435/graphical-user-interface-gui

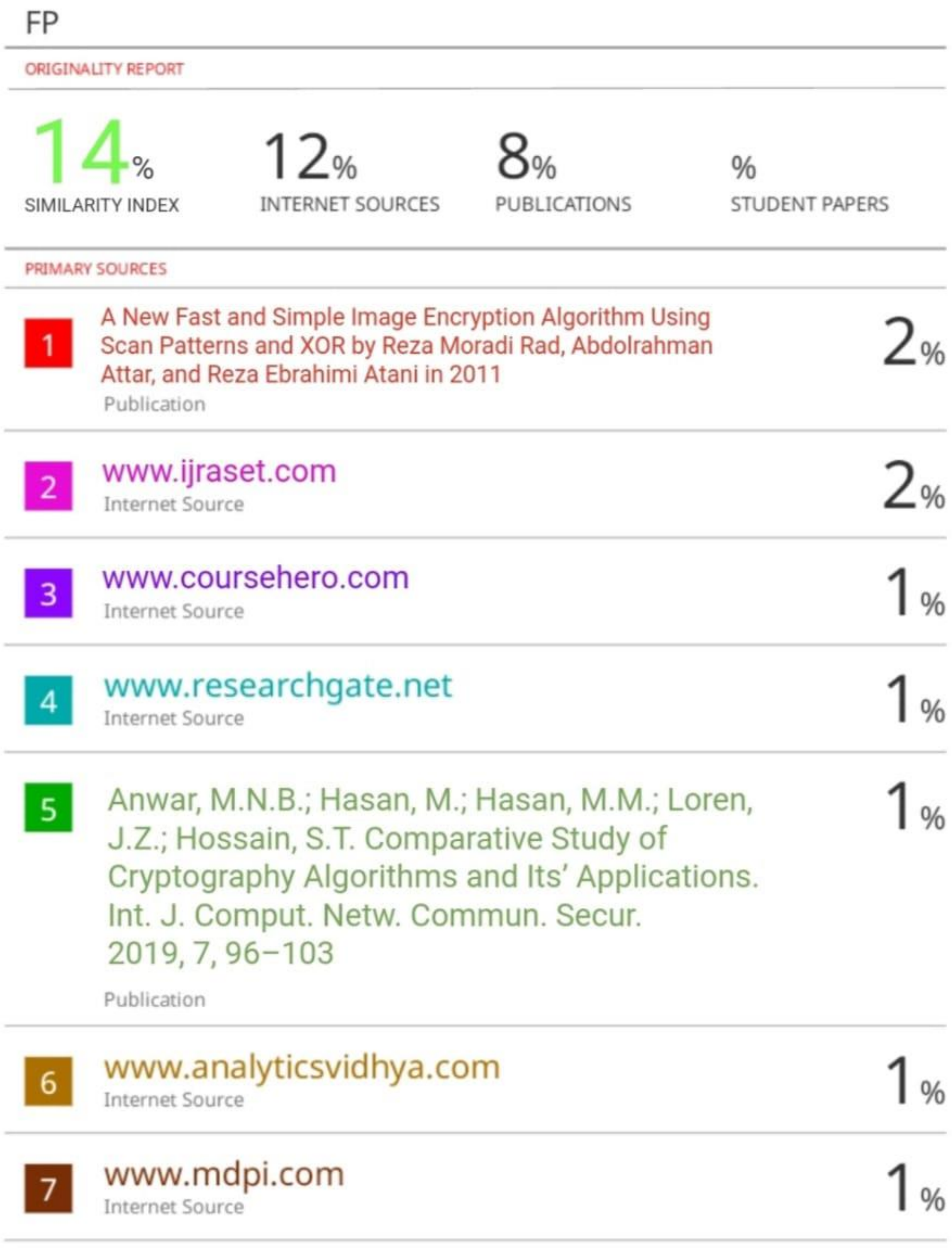
[7] Aloha Sinha, Kehar Singh, "A technique for image encryption using digital signature", Optics Communications, Vol-2 I 8 (2203),229-234.

[8] Securing Medical Images by Image Encryption using Key Image by Shrija Somaraj and Mohammed Ali Hussain in International Journal of Computer Applications (0975 – 8887) Volume 104 – No.3, October 2014

[9] Anwar, M.N.B.; Hasan, M.; Hasan, M.M.; Loren, J.Z.; Hossain, S.T. Comparative Study of Cryptography Algorithms and Its' Applications. Int. J. Comput. Netw. Commun. Secur. 2019, 7, 96–103.

[10] A comparative survey of symmetric and asymmetric key cryptography by Sourabh Chandra, Smita Paira, Sk Safikul Alam and Dr.(Prof.) Goutam Sanyal in 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE).

PLAGIARISM REPORT



FP

ORIGINALITY REPORT

14%	12%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	A New Fast and Simple Image Encryption Algorithm Using Scan Patterns and XOR by Reza Moradi Rad, Abdolrahman Attar, and Reza Ebrahimi Atani in 2011 Publication	2%
2	www.ijraset.com Internet Source	2%
3	www.coursehero.com Internet Source	1%
4	www.researchgate.net Internet Source	1%
5	Anwar, M.N.B.; Hasan, M.; Hasan, M.M.; Loren, J.Z.; Hossain, S.T. Comparative Study of Cryptography Algorithms and Its' Applications. Int. J. Comput. Netw. Commun. Secur. 2019, 7, 96–103 Publication	1%
6	www.analyticsvidhya.com Internet Source	1%
7	www.mdpi.com Internet Source	1%