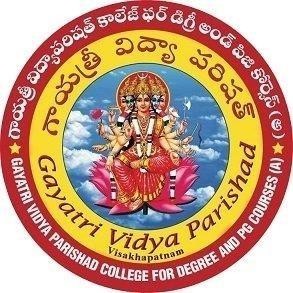
**GAYATRI VIDYA PARISHAD COLLEGE FOR DEGREE AND PG COURSES (A)**

**(Affiliated to Andhra University) Rushikonda, Visakhapatnam Department of Computer Applications**



“Creating human excellence for a better society”

MISSION

VISION

“Unfold into a world class organization with strong academic and research base, producing responsible citizens to cater to the changing needs of the society.”

**El Gamal Cryptosystem for image encryption and decryption**

A Project report submitted in partial fulfilment of

the requirement for the Award of the Degree of

**Master of Computer Applications**

Submitted By

**GUDIVADA SAI SUDHEER**

(Regd.No. PG212202021)

Under the Esteemed guidance of

**P. VENKATA RAO**

ASSOCIATE PROFESSOR



Department of Computer Applications

**GAYATRI VIDHYA PARISHAD COLLEGE**

**FOR DEGREE AND PG COURSES (AUTONOMOUS)**

**(Affiliated to Andhra University)**

**RUSHIKONDA, VISAKHAPATNAM**

**2021-2023**

**GAYATRI VIDHYA PARISHAD COLLEGE FOR DEGREE AND PG COURSES (AUTONOMOUS)**

#### (Affiliated to Andhra University)

**RUSHIKONDA, VISAKHAPATNAM**

**School of Computer Applications**



**C E R T I F I C A T E**

This is to certify that the project report titled “**El Gamal Cryptosystem for image encryption and decryption”** is the bona-fide record of project work carried out by **GUDIVADA SAI SUDHEER (Regd.No.** PG212202021**).** as a student of this college, during the academic year 2022- 2023, in partial fulfillment of the requirement for the award of the degree of Mas ter of Computer Applications.

#### Project Guide Director of MCA

**P. VENKATA RAO Prof. I. S. Pallavi**

**External Examiner**

**DECLARATION**

**I, GUDIVADA SAI SUDHEER** hereby declares that the project report entitled “**El Gamal Cryptosystem for image encryption and decryption”,** is an original work done at **Gayatri Vidya Parishad College for Degree and PG Courses (Autonomous), Visakhapatnam,** submitted in partial fulfillment of the requirements for the award of Master of Computer Applications, Gayatri Vidya Parishad College for Degree and PG Courses(A), affiliated to Andhra University. I assure that this project is not submitted in any other University or college.

**GUDIVADA SAI SUDHEER**

**ACKNOWLEGMENT**

I consider this as a privilege to thank all those people who helped me a lot for successful completion of the project **“El gamal cryptosystem for image encryption and decryption”.**

I would like to thank, **Principal of Gayatri Vidya Parishad College for Degree and PG Courses (A),** **Prof. S. Rajani**, who has provided full-fledged lab and infrastructure for successful completion of my project work.

I would like to thank our ever-accommodating **Director of MCA, Prof. I.S.Pallavi,** who has obliged in responding to every request though she is busy with her hectic schedule of administration and teaching and suggested me with several changes a lot in completion of this project.

I would like to extend my sincere thanks to **Mr. P.Venkata Rao,Associate Professor**,**Department of computer applications** his constant guidance and supervision as well as providing his immense support in completing this project, although he is very busy with his works.

I thank all the **Teaching & Non-Teaching staff** who had been a constant source of support and help during the tenure of completion of this project.

**GUDIVADA SAI SUDHEER**

**EL GAMAL CRYPTOSYSTEM**

**FOR IMAGE**

**ENCRYPTION AND DECRYPTION**

**ABSTRACT**

**ABSTRACT**

El Gamal cryptosystem is one of the well-known public-key algorithms for its ability to generate different ciphertexts for the same plaintext on successive runs. However, this algorithm results in a ciphertext occupying a larger memory space than its plaintext due to its encryption nature. As a result, it is pretty infeasible to use data that require their encrypted form to have the same, such as image data. To overcome this issue, propose a new El Gamal cryptosystem that can be used for any digital data message, including images and text.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CONTENTS** | | | | | | | | | | |
|  | **CHAPTER** | | |  | |  | | | | **Page No.** |
| **1** | | | **. Introduction** | | |  | |  |  |
|  | | | 1.1 | Network Security | | | | | 1 |
|  | | | 1.2 | Principles in network security | | | | | 2 |
|  | | |  | 1.2.1 | | Confidentiality | | | 2 |
|  | | |  | 1.2.2 | | Integrity | | | 2 |
|  | | |  | 1.2.3 | | Availability | | | 2 |
|  | | |  | 1.2.4 | | Non-Repudiation | | | 2 |
|  | | |  | 1.2.5 | | Authentication | | | 3 |
|  | | | 1.3 | Cryptography | | | | | 3 |
|  | | |  | 1.3.1 | | Cryptographic System | | | 3 |
|  | | |  | 1.3.2 | | Crypto Analysis | | | 4 |
|  | | |  | 1.3.3 | | El gamal cryptography | | | 4 |
|  | | |  | 1.3.4 | | Symmetric Key and Asymmetric Key | | | 4 |
| **2** | | | **Literature survey** | | | | | |  |
|  | | | 2.1 | Introduction | | | | | 8 |
|  | | | 2.2 | History | | | | | 8 |
|  | | | 2.3 | Current system | | | | | 9 |
|  | | | 2.4 | Proposed system | | | | | 9 |
|  | | | 2.5 | About python | | | | | 9 |
|  | | |  | 2.5.1 | | Advantages of Python | | | 9 |
|  | | |  | 2.5.2 | | Characterstics of python | | | 10 |
|  | | |  | 2.5.3 | | Applications of Python | | | 10 |
|  | | |  | 2.5.4 | | Python-GUI Programming (Tkinter) | | | 11 |
|  | | |  | 2.5.5 | | Tkinter Programming | | | 11 |
|  | | |  | 2.5.6 | | Tkinter Widgets | | | 11 |
|  | | | 2.6 | Requirement Elicitation | | | | | 13 |
|  | | |  | 2.6.1 | | Functional Requirements | | | 13 |
|  | | |  | 2.6.2 | | Non- Functional Requirements | | | 13 |
| **3** | | | **UML modeling** | | | | | |  |
|  | | | 3.1 | Introduction to UML | | | | | 16 |
|  | | | 3.2 | Goals of UML | | | | | 16 |
|  | | | 3.3 | UML standard diagrams | | | | | 17 |
|  | | |  | 3.3.1 | | Structural diagrams | | | 17 |
|  | | |  | 3.3.2 | | Behavioral diagrams | | | 17 |
|  | | | 3.4 | UML diagrams | | | | | 18 |
|  | | |  | 3.4.1 | | Use case diagram | | | 18 |
|  | | |  |  | | 3.4.1.1 | | Actors | 18 |
|  | | |  |  | | 3.4.1.2 | | Use Cases | 19 |
|  | | |  | 3.4.2 | | Sequence Diagram | | | 21 |
|  | | |  |  | | 3.4.2.1 | | Sequence Diagram for Sender | 22 |
|  | | |  |  | | 3.4.2.2 | | Sequence Diagram for Receiver | 22 |
|  | | |  | 3.4.3 | | State Chart Diagram | | | 23 |
|  | | |  |  | | 3.4.3.1 | | State Chart Diagram for Sender | 23 |
|  | | |  |  | | 3.4.3.2 | | State Chart Diagram for Receiver | 24 |
| **4** | | | **Design** | | | | | |  |
|  | | | 4.1 | Design and Goals | | | | | 26 |
|  | | | 4.2 | Encryption and Decryption Algorithms | | | | | 27 |
|  | | | 4.3 | Examples | | | | | 27 |
| **5** | | | **Coding** | | | | | |  |
|  | | | 5.1 | Coding approach | | | | | 31 |
|  | | | 5.2 | Information handling | | | | | 31 |
|  | | | 5.3 | Programming style | | | | | 31 |
|  | | | 5.4 | Verification and Validation | | | | | 32 |
|  | | | 5.5 | Field level validation | | | | | 32 |
|  | | | 5.6 | Source code | | | | | 32 |
| **6** | | | **Testing** | | | | | |  |
|  | | | 6.1 | Testing activities | | | | | 64 |
|  | | | 6.2 | Types of Testing | | | | | 64 |
|  | | |  | 6.2.1 | | Unit Testing | | | 64 |
|  | | |  | 6.2.2 | | Integration Testing | | | 65 |
|  | | |  | 6.2.3 | | Validation Testing | | | 65 |
|  | | |  | 6.2.4 | | System Testing | | | 65 |
|  | | |  | 6.2.5 | | Usability Testing | | | 65 |
|  | | |  | 6.2.6 | | Performance Testing | | | 65 |
|  | | |  | 6.2.7 | | Installation Testing | | | 66 |
|  | | | 6.3 | Various Types of Testing | | | | | 66 |
|  | | |  | 6.3.1 | | White Box Testing | | | 66 |
|  | | |  | 6.3.2 | | Black Box Testing | | | 66 |
|  | | | 6.4 | Testing Plan | | | | | 66 |
|  | | | 6.5 | Test Case Result | | | | | 66 |
| **7** | | | **Screens** | | | | | | 68 |
| **8** | | | **Conclusion** | | | | | | 71 |
| **9** | | | **References** | | | | | | 72 |
| **10** | | | **Appendix** | | | | | | 73 |

**INTRODUCTION**

1. **INTRODUCTION**

## **1.1 Network security:**

In an organization several computers are connected in network. This network may connect to internet. So, the message transmission is done between the computers. Here, we need the network security to protect our message transmissions. This is the criteria of providing network security.

Network security involves the authorization of access to data in a network, which is controlled by the network administrator. Users choose or assigned to username and password or other authenticating information that allows them access to information and programs within their authority. Network security covers a verity of computer networks, both public and private, that are used in everyday jobs conducting transactions and communications among business, government agencies and individuals. Networks can be private, such as within a company, and others which might be open to public access. Network security is involved in organizations, enterprises, and other types of institutions. It does as its title explains: It secures the network, as well as protecting and overseeing operations being done. The most common and simple way of protecting a network resource is by assigning it a unique name and a corresponding password.

The provision and policies adopted by a network administrator to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources. Network security involves the authorization of access to data in a network, which is controlled by the network administrator. Uses can choose or are assigned and ID and password or other authenticating information that allows them access to information and program within their authority.

Network security covers a variety of computer networks, both public and private, that are used in everyday jobs conducting transaction and communications among business, government agencies and individuals. Network can be private, such as within a company, and others which might be open to public access. Network security is involved in organizations, enterprises, and other type of institutions.

The networks are computer networks, both public and private, that are used every day transactions and communications among businesses, government agencies and individuals. The networks are comprised of “nodes’, which are “client” terminals (individual user PCs), and one or more “servers” and/or “host” computers. They are linked by communication system, some of which might be private, such as within a company and others which might be open to public access. The obvious example of a network system that is open to public access is the Internet, but many private networks also utilize publicly-accessible communications.

Today, most companies host computers can be accessed by their employees whether in their offices over a private communications network, or from their homes or hotel rooms while on the road through normal telephone lines.

## **1.2 Principles in Network Security:**

To provide adequate protection of network resources, the procedures and technologies that you deploy need to guarantee three things, sometimes referred to as the CIA triad: the following are some useful security services.

## **1.2.1 Confidentiality:**

Confidentiality is roughly equivalent to privacy. Measures undertaken to ensure confidentiality are designed to prevent sensitive information from reaching the wrong people, while making sure that the right people can in fact get it: Access must be restricted to those authorized to view the data in question. It is common, as well, for data to be categorized according to the amount and type of damage that could be done should it fall into unintended hands. More or less stringent measures can then be implemented according to those categories.

Sometimes safeguarding data confidentiality may involve special training for that privy to such documents. Such training would typically include security risks that could threaten this information. Training can help familiarize authorized people with risk factors and password-related best practices and information about social engineering methods, to prevent them from bending data-handling rules with good intensions and potentially disastrous results.

## **1.2.2 Integrity:**

Integrity involves maintaining the consistency, accuracy, and trustworthiness of data over its entire life cycle. Data must not be changed in transit, and steps must be taken to ensure that data cannot be altered by unauthorized people. These measures include file permissions and user access controls. Version control may be used to prevent erroneous changes or accidental deletion by authorized users from becoming a problem. In addition, some means must be in place to detect any changes in data that might occur as a result of non-human-caused events such as an electromagnetic pulse or server crash. Some data might include checksums, even cryptographic checksums, for verification of integrity. Backups or redundancies must be available to restore that affected data to its state.

## **1.2.3 Availability:**

Availability of information refers to ensuring that authorized parties can access the information when needed. Information only has value if the right people can access it at the right times. Denying access to information has become a very common attack nowadays. Almost every week you can find news about high profile website access to the resources of the website. Such downtime can be very costly. Other factors that could lead to lack of availability to valuable information may include accidents such as power outages or natural disasters such as floods.

## **1.2.4 non-repudiation:**

The creator/sender of the information cannot deny at alter stage his or her intentions in the creation or transmission of the information. In law, non-repudiation implies one’s intention to fulfil their obligation to a contract.

It is important to note that while technology such as cryptographic system can assist in non-repudiation efforts, the concept is at its core legal concept transcending the realm of technology. It is not, for instance, sufficient to show that the message a digital signature signed with the sender’s private key, and thus only the sender could have sent the message and nobody else could have altered it in transit. The alleged sender could in return demonstrate that the digital signature algorithm is vulnerable or flawed or prove that his signing key has been compromised.

## **1.2.5 Authentication:**

The sender and receiver can confirm their identity and the origin/destination of the information. In computing, e-Business, and information security, it is necessary to ensure that the data, transaction, communication or documents (electronic or physical) are genuine. It is also important for authenticity to validate that both parties involved are who they claim to be. Some information security systems incorporate authentication features such as “digital signatures”, which give evidence that the message data is genuine and was sent by someone possessing the proper signing key.

## **1.3 Cryptography:**

Cryptography was concerned totally with message encryption, i.e., the conversion of message from an intelligible form into unintelligible one and reverse again at the other end, rendering it unreadable by an unauthorized person without the knowledge of secret key (decryption key). In the modern age of technology cryptography is becoming a more and more central topic within computer science. As there is a need for more secure cryptographic schemes, the application of graph theory is going to increase for the development of secure encryption algorithms. R. Yadhu have proposed a selective encryption mechanism using message specific key and spanning tree concept of graph theory. The mechanism provides protection of privacy in communication as it avoids the formation of self-loops and parallel edges and key is exchanged only among the authenticated persons only. Graph theory has a great contribution in the development of various encryption techniques. In this paper we propose a scheme for secure communication using prime weighted graph.

Cryptography is the art and science of secure data communications over insecure channels. It is the study of method of sending messages in disguised form so that only the intended recipients can remove the disguise and interpret the message. Historically, the major consumers of cryptography were military and intelligence organizations. Today, however, cryptography is everywhere! Security mechanisms that rely on cryptography are an integral part of almost any computer system. Users rely on cryptography every time they access a secured website. Cryptographic methods are used to enforce access control in multi-user operating systems, and to prevent thieves from extracting trade secrets from stolen laptops. Software protection methods employ encryption, authentication, and other tools to prevent copying.

## **1.3.1 CRYPTOGRAPHIC SYSTEM**

A cryptographic system is any computer system that involves cryptography. Such systems include for instance, a system for secure electronic mail which might include methods for digital signatures, cryptographic hash functions, key management techniques, and so on. Cryptographic systems are made up of cryptographic primitives and are usually rather complex. Because of this, breaking a cryptosystem is not restricted to breaking the underlying cryptographic algorithms; usually, it is far easier to break the system as a whole.

**1.3.2 CRYPTOANALYSIS**

Cryptoanalysis refers to the art and science of analyzing information systems in order to study the hidden aspects of the systems. Cryptanalysis is used to breach cryptographic security systems and gain access to the contents of encrypted messages, even if the cryptographic key is unknown.

**1.3.3El Gamal Cryptography**

El Gamal cryptography is a public-key encryption algorithm named after its inventor, Taher El Gamal. It is based on the computational difficulty of solving the discrete logarithm problem in finite fields or elliptic curve groups.

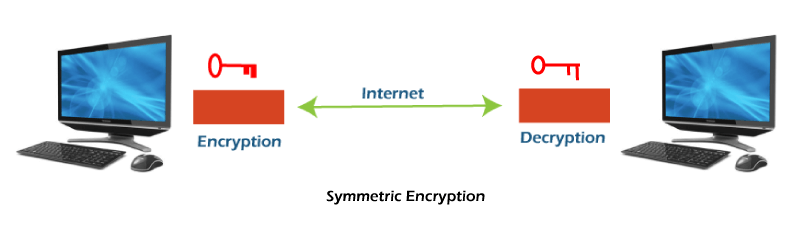
El Gamal encryption is a probabilistic encryption scheme that allows secure communication between two parties, commonly referred to as the sender and the recipient. The algorithm involves the use of a public key for encryption and a private key for decryption.

El Gamal encryption provides semantic security, meaning that even if an adversary obtains multiple ciphertexts, it is computationally infeasible to determine the corresponding plaintext without knowledge of the private key.

El Gamal cryptography has applications in secure communication protocols, digital signatures, and key exchange protocols. It offers an alternative to other public-key encryption schemes like RSA, providing different security properties and computational characteristics.

## **1.3.4 Symmetric Key:**

Symmetric encryption encrypts and decrypts the information using a single password. In this encryption technique, the message is encrypted with a key, and the same key is used for decrypting the message. It is the simplest and commonly known encryption technique. It makes it easy to use but less secure.

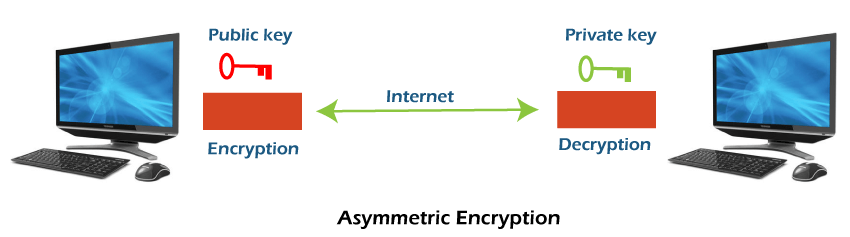


It is called symmetric encryption because the same key is responsible for encrypting or decrypting the data. The single key used in symmetric encryption is used to encrypt plain text into ciphertext, and that same key is used to decrypt that ciphertext back into plain text.

Symmetric encryption is also called secret key encryption. The algorithm behind the symmetric encryption executes faster and less complex, so it is the preferred technique to transmit the data in bulk.

**Asymmetric Key:**

Asymmetric encryption uses two keys for encryption and decryption. It is based on the technique of public and private keys. A public key, which is interchanged between more than one user. Data is decrypted by a private key, which is not exchanged. It is slower but more secure. The public key used in this encryption technique is available to everyone, but the private key used in it is not disclosed.

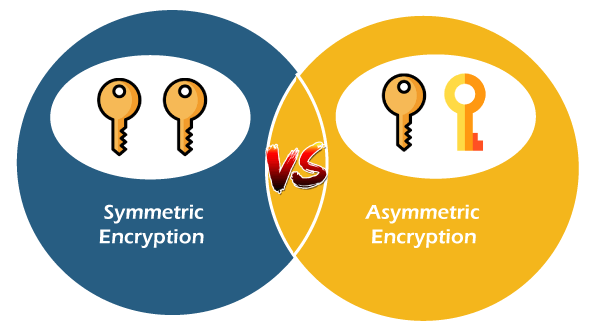


The drawback of this encryption is that it takes more time than the symmetric encryption process. Asymmetric encryption is slower than secret-key encryption because, in secret key encryption, a single shared key is used to encrypt and decrypt the message, while in public-key encryption, two different keys are used, both related to each other by a complex mathematical process. Therefore, we can say that encryption and decryption take more time in public-key encryption.

In asymmetric encryption, a message that is encrypted using a public key can be decrypted by a private key, while if the message is encrypted by a private key can be decrypted by using the public key. Asymmetric encryption is widely used in day-to-day communication channels, especially on the internet.

That's about the description of both encryption techniques. Both encryption techniques have their own benefits and limitations, but from a security perspective, asymmetric encryption is a better choice. Now, let's see the comparison chart between both techniques. We are comparing asymmetric and symmetric encryption based on some characteristics.

**Symmetric encryption v/s Asymmetric encryption**



### 

**LITERATURE SURVEY**

## **2.Literature Survey**

## **2.1 Introduction:**

El Gamal encryption is a public-key cryptosystem. It uses asymmetric key encryption for communicating between two parties and encrypting the message.

This generates a set of bytes with the same approximately equal to the number of bytes in the given message during the encryption process. This is accomplished by using the public key information from the user to whom the encrypted message is to be sent.

The security of the El Gamal cryptosystem is based on the intractability of the discrete Logarithm problem. This problem involves finding the exponent of a given number in a finite field, which is computationally expensive foe large fields.

**2.2 History:**

The El Gamal cryptosystem was developed by Taher El Gamal in 1985 while he was working at the Stanford Research Institute (now SRI International). El Gamal’s goal was to create a public-key encryption scheme that could provide the same level of security as the widely used RSA algorithm but with a different mathematical foundation.

El Gamal’s work was influenced by the Diffie-Hellman key exchange protocol, which had been proposed a decade earlier. Diffie-Hellman introduced the concept of public-key cryptography, where encryption and decryption keys are distinct. El Gamal expanded upon this idea and developed a full-fledged cryptosystem based on the computational intractability of the discrete logarithm problem.

The El Gamal cryptosystem was first described in El Gamal’s paper titled "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," which was published in the IEEE Transactions on Information Theory in 1985. The paper outlined the encryption algorithm, the key generation process, and the signature scheme associated with the El Gamal cryptosystem.

El Gamal's cryptosystem gained recognition for its security properties and versatility. It offered encryption, digital signatures, and key exchange capabilities, making it suitable for various cryptographic applications. The security of the El Gamal cryptosystem is based on the presumed difficulty of calculating discrete logarithms in finite fields.

Despite its contributions and widespread usage, the El Gamal cryptosystem is known to be computationally more expensive compared to some other encryption algorithms, particularly in terms of processing time and key size. Consequently, it is often used in combination with other cryptographic techniques or in scenarios where the security benefits of public-key encryption are essential, such as key exchange and digital signatures.

## **2.3 Current System:**

El Gamal cryptosystem can be defined as the cryptography algorithm that uses the public and private key concepts to secure communication between two systems. It can be considered the asymmetric algorithm where the encryption and decryption happen by using public and private keys. However, this algorithm results in a ciphertext occupying a larger memory space than its plaintext due to its encryption nature. As a result, it is pretty infeasible to use data that require their encrypted form to have the same such as image data.

## **2.4 Proposed System:**

The proposed approach mainly tests image data, consisting of three stages: key pair generation, image encryption, and image decryption. First, we generate as many random bytes as required for encrypting or decrypting images using the sender or receiver's public key information. Then use an XOR operation between each pixel in the image and each randomly generated byte to obtain the encrypted or decrypted image.

**2.5 About Python**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

**2.5.1 Advantages of Python?**

Python is a high-level, interpreted, interactive, and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is a MUST for students and working professionals to become a great Software Engineer especially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports an Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**2.5.2 Characteristics of Python**

Following are important characteristics of Python Programming −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**2.5.3 New Approach for building window Software**

The Python Framework simplifies Windows development. It provides developers with a single approach to build both desktop applications sometimes called smart client applications and Web-Based applications. It also developers to use the same tools and skills to develop software for a verity of system ranging from handled smart phones to large server installations.

**2.5.4 Applications of Python**

As mentioned before, Python is one of the most widely used language over the web. I'm going to list few of them here:

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

# **2.5.5 Python - GUI Programming (Tkinter):**

Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

* **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
* **wxPython** − This is an open-source Python interface for wxWindows http://wxpython.org.
* **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine http://www.jython.org.

There are many other interfaces available, which you can find them on the net.

# **2.5.6 Tkinter Programming**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

* Import the Tkinter module.
* Create the GUI application main window.
* Add one or more of the above-mentioned widgets to the GUI application.
* Enter the main event loop to take action against each event triggered by the user.

# **2.5.7 Tkinter Widgets**

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table –

**Button**: The Button widget is used to display buttons in your application.

**Canvas**: The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.

**Check button**: The Check button widget is used to display a number of options as checkboxes. The user can select multiple options at a time.

**Entry:** The Entry widget is used to display a single-line text field for accepting values from a user.

**Frame:** The Frame widget is used as a container widget to organize other widgets.

**Label**: The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

**Listbox**: The List box widget is used to provide a list of options to a user.

**Menu button:** The Menu button widget is used to display menus in your application.

**Menu:** The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.

**Message**: The Message widget is used to display multiline text fields for accepting values from a user.

**Radiobutton** : The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.

**Scale:** The Scale widget is used to provide a slider widget.

**Scrollbar**: The Scrollbar widget is used to add scrolling capability to various widg-ets, such as list boxes.

**Text**: The Text widget is used to display text in multiple lines.

**Top level**: The Top-level widget is used to provide a separate window container.

**Spin box**: The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.

**PanedWindow**: A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.

**LabelFrame**: A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.

**tkMessageBox**:This module is used to display message boxes in your applications.

**2.6 REQUIREMENT ELICITATION:**

Requirement is the feature the system must have a constraint that it must satisfy to be accepted by the clients. Requirements engineering aims at defining the requirements of the system under construction. It includes two main activities namely Requirements Elicitation and Analysis.

Requirements Elicitation focuses on describing the process of the system. The client, the developer and the users identify the problem. Such a definition is called Requirement Specification. This specification is structured and formulized during analysis to produce an Analysis Model.

Requirements Elicitation and Analysis focuses only on the user’s view of the system. Requirements Elicitation includes the following activities:

**2.6.1 Functional Requirements:**

It describes the interactions between the system and its environment independent of its implementation.

The functional requirements are:

**Encryption:**

Input: Original image

Output: Cipher image.

**Decryption:**

Input: Cipher image

Output: Original image

**2.6.2 NON-FUNCTIONAL REQUIREMENTS:**

Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc. During this activity, developers, users and clients agree on aspects like performance of system, documentation, resources, security and its quality.

**Usability :** The GUI of this system provides easy access to the user can get best results. It is also providing help to know how to use the system.

**Reliability :** This system is reliable to encrypt or decrypt every message given by user.

**Portability**  : This system can runs on any platform.

**Implementation Using**: Python Programming tkinter module.

**Robustness** : This system functions in the presence of invalid inputs as displaying message that to provide correct details.

. . **HARDWARE REQUIREMENTS:**

The Minimum hardware requirements to run this system are:

* 1. **Processor** : Intel core i3 and above
  2. **Ram** : 4.00GB
  3. **System Type** : 64/32-bit Operating System
  4. **Hard Disk** : 500GB
  5. **Monitor**  : Standard
  6. **Keyboard** : Standard
  7. **Mouse**  : Standard

**SOFTWARE REQUIREMENTS:**

The Minimum software requirements to run this system are:

|  |  |  |
| --- | --- | --- |
| **1.** | **Front end Design** | **:** Python 3.8.3 IDLE |
| **2.** | **Operating System** | **:** Windows 10 |
| **3.** | **Back-end Design** | **:**  Python Programming Language |

**UML MODELING**

### **3. UML MODELING**

### **3.1 Introduction to UML**

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software system. UML was created by the Object Management Group (OMG) and UML 1.0 specification drift was proposed to the OMG in January 1997.

OMG is continuously making efforts to create a truly industry standard.

* UML stands for Unified Modeling Language.
* UML is different from the other common programming language such as C++, java, COBOL, etc.
* UML is a pictorial language used to make software blueprints.
* UML can be described as a general-purpose visual modeling language to visualize, specify, construct and document software system.
* Although UML is generally used to model software system, it is not limited within this boundary. It is generally used to model software system as well. For example, the process flows in a manufacturing unit, etc.

UML is not a programming language, but tools can be used to generate code in various language using UML diagrams. UML has a direct relation with object-oriented analysis and design. After some standardization, UML has become an OMG standard.

### **3.2 Goals of UML**

A picture is worth a thousand words, this idiom absolutely fits describing UML. Object-oriented concepts were introduced much earlier than UML. At that point of time, there were no standard methodologies to organize and consolidate the Object-oriented development. It was then that UML came into picture.

There are number of goals for developing UML but the most important is to define some general-purpose modeling language, which all models can use and it also need to be made simple to understand and use.

UML diagram are not only made for developers but also for business users, common people, and anybody interested to understand the system. The system can be a software or non-software system. Thus, it must be clear that UML is not a development method rather it accompanies with processes to make it a successful system. In conclusion, the goal of UML can be defined as a simple modeling mechanism to model all possible practical system in today’s complex environment.

### 

### **3.3 UML standard diagrams:**

The elements are like components which can be associated in diverse ways to make a complete UML picture, which is known as diagram. Thus, it is very important to understand the different diagrams to implement the knowledge in real-life system. Any complex system is best understood by making some kind of diagrams or pictures. These diagrams have a better impact on our understanding. If we look around, we will realize that the diagram is not a new concept but it is used widely in different forms in different industries. We prepare UML diagram to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system. You can also create your own set of diagrams to meet your requirements. Diagrams are generally made in an incremental and iterative way. There are two broad categories of diagram and they are again divided into subcategories:

* Structural Diagrams
* Behavioural Diagrams

### **3.3.1 Structural Diagrams**

The structural diagram represents the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable. These static parts are represented by classes, interfaces, object, components, and nodes. The four structural diagrams are:

* Class diagram
* Object diagram
* Component diagram
* Deployment diagram

### **3.3.2 Behavioural Diagrams**

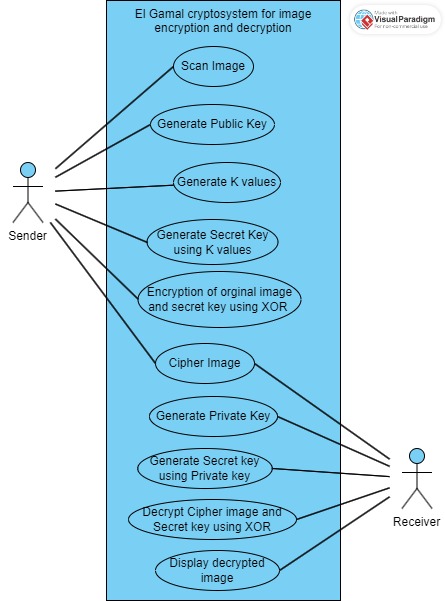
Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered. Behavioural diagram captures the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system. UML has the following five types of behavioural diagrams:

* Use case diagram
* Sequence diagram
* Collaboration diagram
* State chart diagram
* Activity diagram

### **3.4 UML Diagrams**

### **3.4.1 Use Case Diagram:**

Use case describes the behaviour of the system as seen from the actor’s point of view. A use case diagram can portray then different types of users of a system and the many ways that they interact with system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well. Actors initiate the use cases for accessing system’s functionality. When actors and use cases exchange information, they are said to Communicate.



**Figure 3.4.1 Use case diagram (Sender and Receiver)**

### **3.4.1.1 Actors**

In this use case diagram, the sender will enter the plain text then encrypt with the key generated using Pythagorean triple algorithm then the cipher text is obtained. Now the receiver will choose the cipher text then decrypt by using the key generated at the sender side. Now plain text is generated at receiver side.

There are two main actors 1) sender 2) receiver

**Sender:** The sender will enter the Original image then the encryption process will be performed by using Public and Private keys.

**Receiver:** The receiver will browse the cipher image using a private key we can generate Orginal image .

**3.4.1.2 Use Cases**

**Plain Image :** Sender will enter the plain Image with respect to the requirement.

**Keys**  : Sender will enter key

**Encryption :** Sender will Browse the plain image and s key encryption and apply plain image with public and private keys then we will get cipher Image.

**Decryption** : we can browse the cipher Image here and enter private keys we can get Orginal image.

**Description**

|  |  |
| --- | --- |
| **Use case Name** | Sender and receiver |
| **Participating Actors** | Sender and receiver will Encrypt and Decrypt the images |
| **Entry Condition** | Sender Enter the Original image |
| **Flow of Events** | 1. Enter the Original image  2. Display the Original Image on the screen |
| **Exit Condition** | Display the plain text on the screen. |

**Table 3.1 : Sender Use case**

### 

### Use case for Key Generation

|  |  |
| --- | --- |
| **Use case Name** | Key Generation |
| **Participating Actors** | Sender |
| **Entry Condition** | Find Values that should we have |
| **Flow of Events** | Calculating key value using the formulas |
| **Exit Condition** | Then key will be generated |

**Table 3.2 Key generation Use case**

### 

### Use case for Encryption

|  |  |
| --- | --- |
| **Use case Name** | Encryption |
| **Participating Actors** | Sender |
| **Entry Condition** | Enter the plain image |
| **Flow of Events** | 1. Enter the plain image.  2. Convert plain into numbers.  3. Encrypt the message by pressing the encrypt button.  4. Save the Cipher image to the Receiver. |
| **Exit Condition** | Obtain the cipher image |

**Table 3.3 Encryption Use case**

### 

### Use case for Decryption

|  |  |
| --- | --- |
| **Use case Name** | Decryption |
| **Participating Actors** | Receiver |
| **Entry Condition** | Select the cipher image |
| **Flow of Events** | 1. Select cipher image. 2. Convert cipher image into numbers. 3. Decrypt the cipher image by pressing the button. 4. We will obtain the plain image. |
| **Exit Condition** | Obtain the Original image |

**Table 3.4 Decryption Use Case**

### 

### Encryption Scenario:

|  |  |
| --- | --- |
| **Scenario Name** | Encryption |
| **Participating Actors** | Sender |
| **Flow of Events** | 1. Enter the plain Image. 2. Plain text is converted into cipher image. 3. Send the cipher image to the receiver. |

**Table 3.5: Encryption Scenario**

Decryption Scenario:

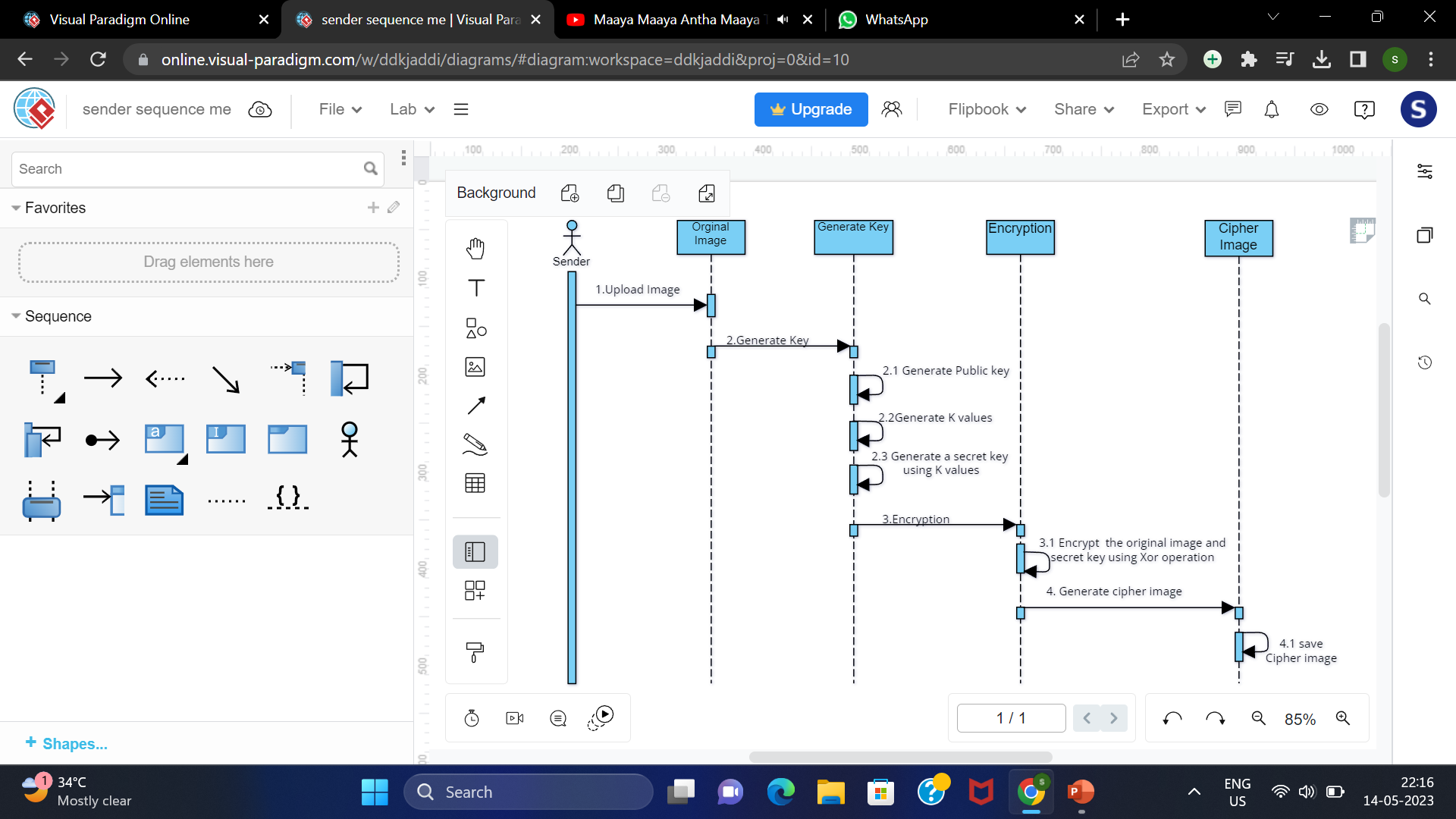
|  |  |
| --- | --- |
| **Scenario Name** | Decryption |
| **Participating Actors** | Receiver |
| **Flow of Events** | 1. Receiver will decrypt the cipher image. 2. Used the key sent by the sender. |

### 

### **3.4.2 Sequence Diagram**:

Interaction between object can be described by means of sequence diagrams. An object interacts with another object by sending messages. The reception of a message by an object triggers the execution of an operation, which in turn may send messages to other objects. Arguments may be passed along with a message and are bound to the parameters of the executing operation in the receiving object.

**3.4.2.1 Sequence diagram for sender:**

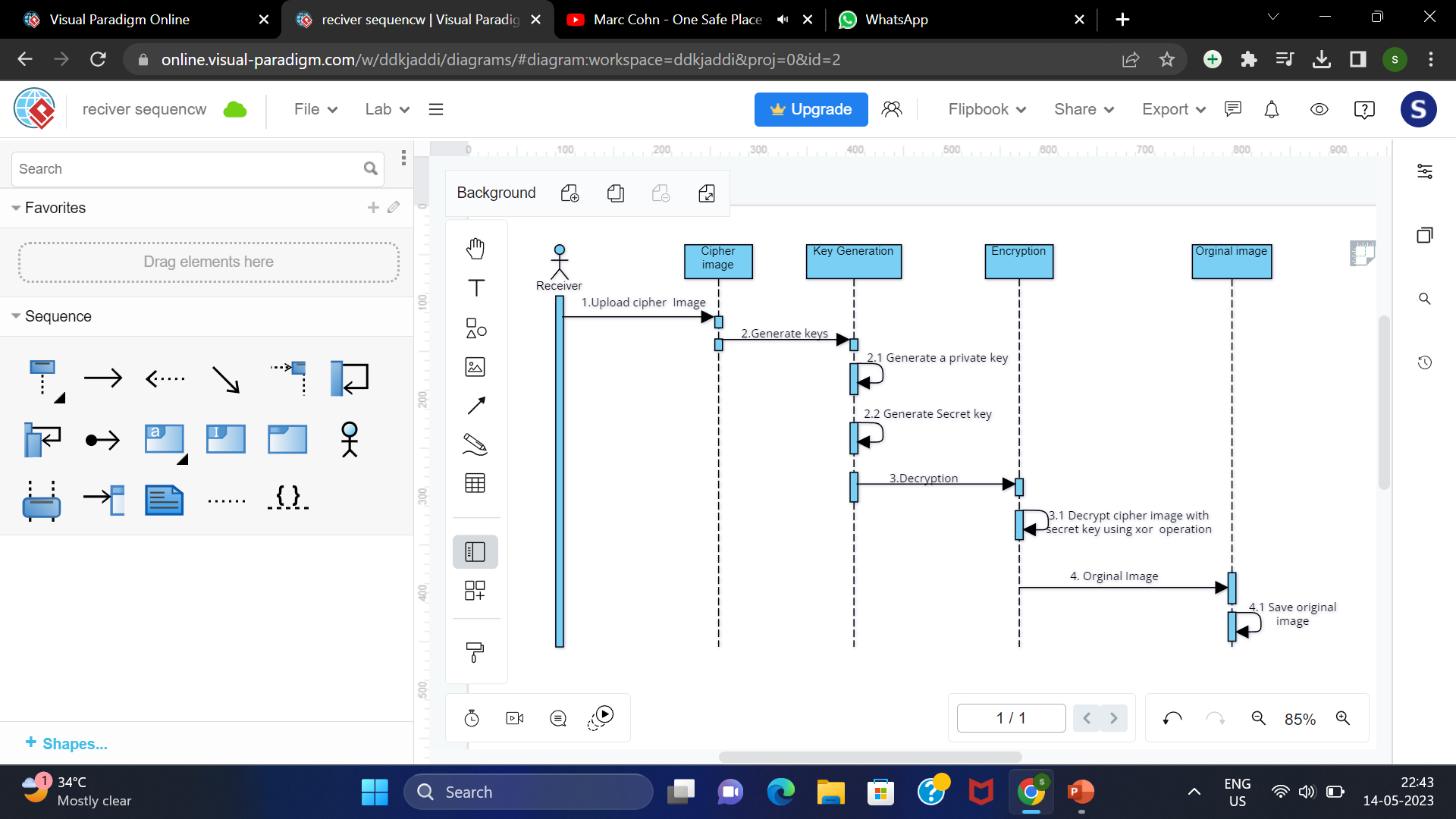


**Figure 3.4.2.1: Senders Sequence Diagram**

### Description:

**Sender:** The sender will enter the plain image then the encryption process will be performed by using public and private keys. It generates a secret key to perform the Xor operation then we get the Cipher Image.

### **3.4.2.2 Sequence diagram for receiver:**

****

**Figure 3.4.2.2 Receivers Sequence Diagram**

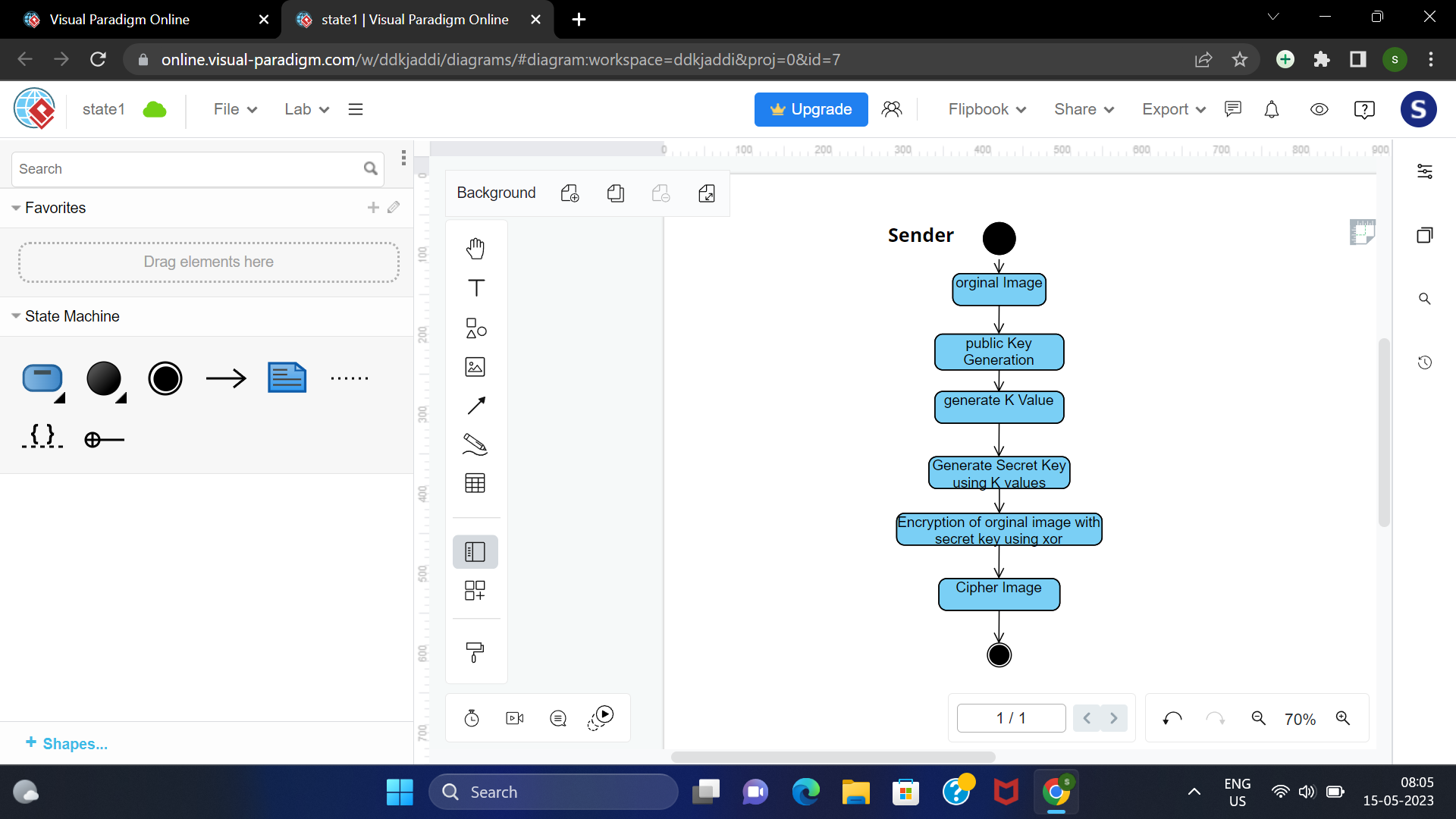
### Description:

**Receiver:** The receiver will browse the ciphertext and private key used to generate the secret key and perform the Xor operation using we get the plain Image.

**3.4.3 State-Chart Diagram:**

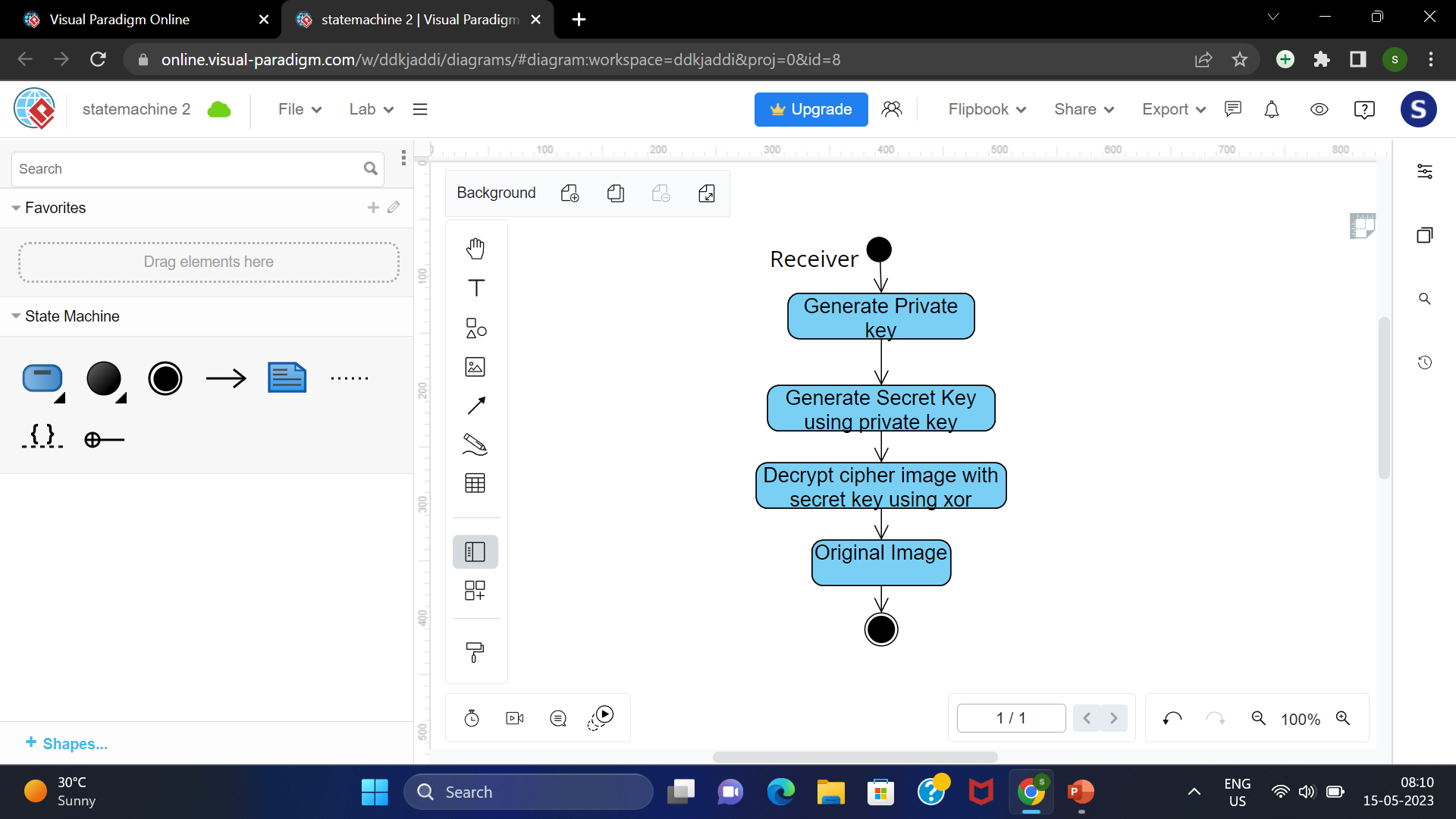
State diagrams are used to describe the behavior of a system. State diagram describe all the possible state of an object as events occur. Each diagram usually represents objects of a single class and track the different state of its objects through the system. Not all classes will require a state diagram and state diagram are not useful for describing the collaboration of all objects in a use case. State diagram have very few elements.

**3.4.3.1 State chart diagram for sender:**



**Sender:** The sender will enter the plain image then the encryption process will be performed by using public and private keys .It generates secret key to perform Xor operation then we get the Cipher Image.

**3.4.3.2 State chart diagram for Receiver:**



**Receiver:** The receiver will browse the ciphertext and private key used to generate secret key perform the Xor operation using we get the plain Image.

**DESIGN**

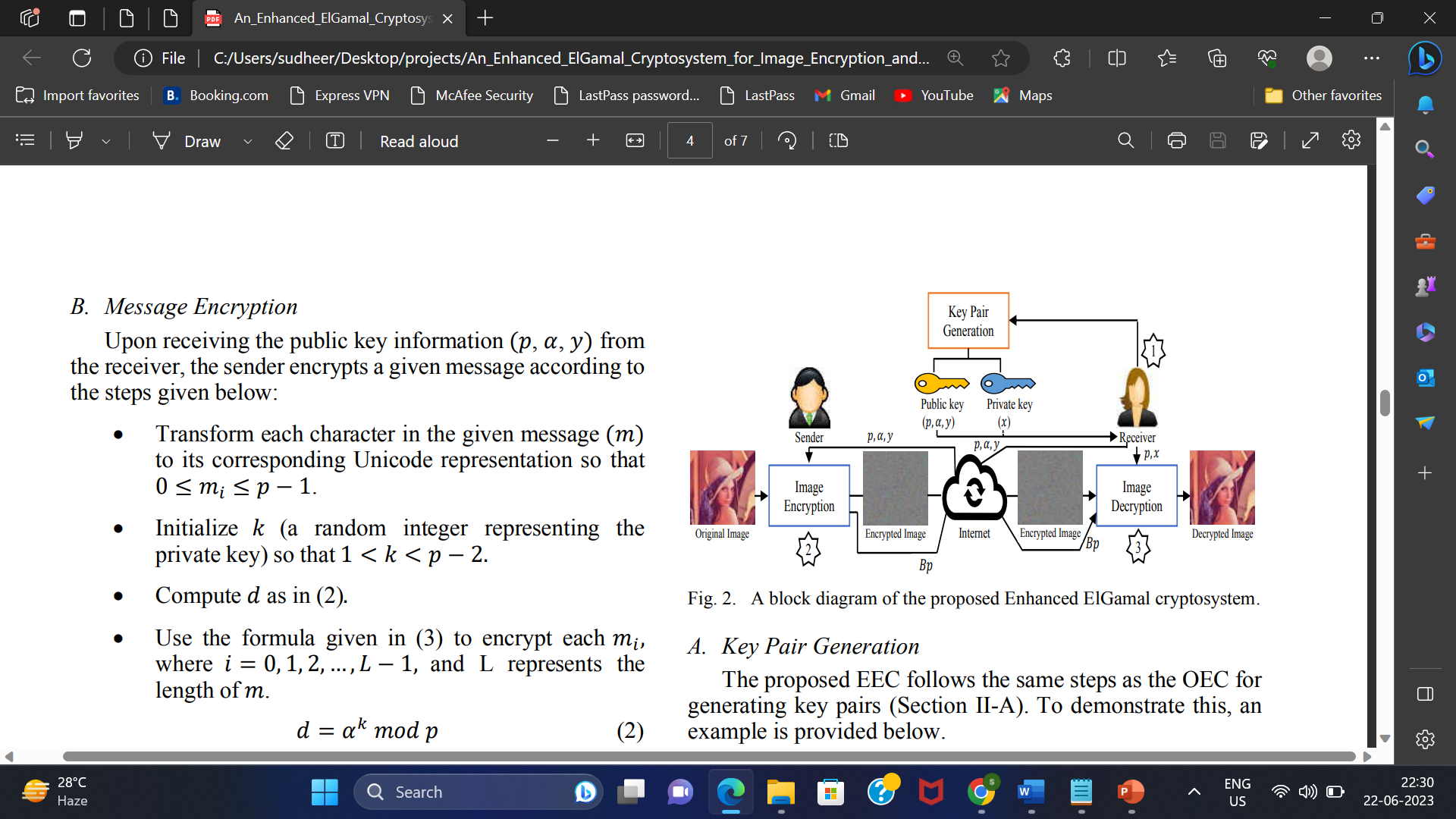
# **4. DESIGN**

## **4.1 Design and Goals :**

Encryption is the process of transforming plaintext into the cipher text where plain text is the input to the encryption process and cipher text is the output of the encryption process. Decryption is the process of transforming cipher text into the plain text where cipher text is the input to the decryption and plain text is the output of the decryption process. There are various encryption algorithms exit classified as symmetric and asymmetric encryption algorithms.

**Encryption and Decryption**

Data that can be read and understood by anyone without any special knowledge about it is called orginal image. The method of disguising the orginal image in such a way as to hide the information is called encryption. Encrypting orginal image results in unreadable gibberish called cipher image. You use encryption to ensure that information is hidden from anyone for whom it is not intended, even those who can see the encrypted data. To prevent the encrypted message from having a larger size than its original message during the encryption process, we propose to perform an XOR operation between each byte of the original message and one byte of a randomly generated set of bytes at a time.



**Conventional Cryptography:**

The process of reverting cipher image to its original image is called decryption. we intend to generate as many bytes as required for encrypting or decrypting any given image using the public key information provided by the sender or the receiver. Accordingly, both parties can obtain the same shared secret key, which helps them encrypt or decrypt any given image.

**4.2 ENCRYPTION ALGORITHM**

Input: im ,p, a and y.

Output: im and Bp.

Step1: Read im ,p, a and y.

Step2. Initialize Bp and fsk as follows :Bp=[] and fsk=[].

Step3: Initialize im with im zeros , so its shape and type are identical to im.

Step4: Select k(a random integer) such that 1<k<p-2

Step5: Compute d as in (2).

Step6: Append d into Bp.

Step7: Compute sk as in (6)

Step 8: Convert the sk into a set of bytes and then append each byte separately into fsk.

Step9: If len(fsk) is less than len(im) go to step 4

Step 10: Compute im as in (7) to obtain an encrypted image

**DECRYPTION ALGORITHM**

Input: im , Bp , p and x

Output: im

Step1: Read im , Bp, p and x.

Step2: Initialize fsk as follows: fsk=[]

Step3: Initialize im with im zeros, so its shape and type are identical to im

Step4: Divide Bp into a set of blocks as follows: Bp1,Bp2,…Bpn-1.

Step5: For each block Bpi , do the following:

1)Compute sk as in(8)

2)Convert the sk into a set of bytes and then append each byte

separately into fsk

Step6: Compute imijc as in (9) to obtain the original image

**4.3Example:**



**Key Generation:**

P = 263

α =131 where 1< α<p-1

X =9

Y = "α" ^x mod p

=131^9 mod26

Y =9

**Encryption :**

=5, =3, =4, =10, =7, =2, =9, =8, =3 where 1<K<P-2

**sk =**

= = 239 = = 30 = = 190

= = 50 = = 177 = = 157

= = 174 = = 69 = = 30

**fsk= [239,30,190,50,177,157,174,69,30]**

**d =**

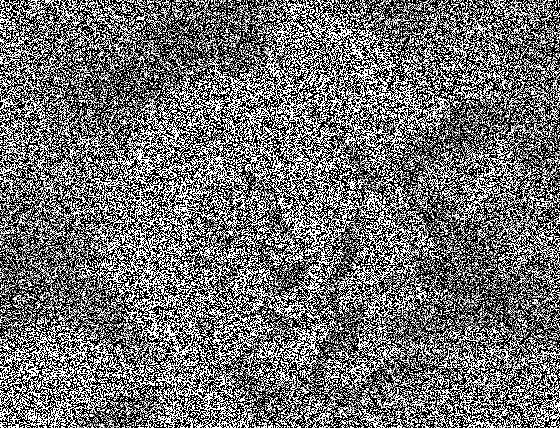
4

**Bp=[189,230,148,216,113,66,94,75,230]**

^239 = 227 ^ 30 = 0 ^190 = 181

^ 50 = 1 ^177 = 231 ^157 = 251

^174 = 245 ^ 69 = 23 ^30 = 27

****

**Decryption:**

**Sk =**

= 239 = 30 = 190

= 50 = 177 = 157

= 174 = 69 = 30

^239 = 12 30 = 30 190 = 11

50 = 51 177 = 86 157 = 102

^174 = 91 69 = 82 30 = 5



**CODING**

**5. CODING**

The goal of coding or programming phase is to translate the design of the system produced during the phase into code in a given programming language, which can be executed by a computer and the performs the computation specified by the design.

The coding phase affects both testing and maintenance. The goal of coding is not to reduce the implementation cost, but the goal should be to reduce the cost of later phase. In other words, the goal is not to simplify the job of programmer. Rather the goal should be to simplify the job of the tester and maintainer.

## **5.1 Coding Approach:**

There are two major approaches for coding any software system. They are top-Down approach and bottom up approach.

Bottom-up approach can suit for developing the object-oriented systems. During system design phase of reduce the complexity. We decompose the system into appropriate number of subsystems, for which objects can be modelled independently. These objects exhibit the way the subsystems perform their operations.

Once object have been modeled they are implemented by means of coding. Even though related to the same system as the objects are implemented of each other the Bottom-Up approach is more suitable for coding these objects.

In this approach, we first do the coding of objects independently and then we integrate these modules into one system to which they belong. In this project, top-Down approach is followed. For registration and Login. User will clicks and stores the intruder information intensely.

## **5.2 Information Handling:**

Any software system requires some amount of information during its operation selection of appropriate data structures can help us to produce the code so that objects of the system can better operate with the available information decreased complexity.

In this project, Encryption and decryption will not be possible if the image fields are vacant. System will not have any default values. User must specify each secret file name in encryption and locate all required operations in decryption.

## **5.3 Programming Style:**

Programming style deals with act of rules that a programmer must follow so that the characteristics of coding such as Traceability, Understands the ability, Modifiability, and Extensibility can be satisfied.

In the current system, we followed the coding rules for naming the variables and methods. The system is developed in a very interactive and users friendly manner.

**5.4 Verification and Validation:**

Verification is the process of checking the product built is right. Validation is the process of checking whether the right product is built. During the Development of the system coding for the object has been thoroughly verified from various aspects regarding their design, in the way they are integrated and etc. The various techniques that have been followed for validation discussed in testing the current system. Validations applied to the entire system at two levels:

## **5.5 Field level validation:**

Validations at the level of individual controls are also applied whenever necessary. System pops up appropriate and sensuous dialogs whenever necessary.

**5.6 Source Code:**

import cv2

import numpy as np

from cv2 import \*

from PIL import Image

from numpy import asarray

from tkinter import \*

from tkinter import messagebox

from PIL import ImageTk,Image

from tkinter.filedialog import asksaveasfile

from tkinter.filedialog import askopenfile

import tkinter as tk

from tkinter import ttk

from tkinter.colorchooser import askcolor

from tkinter import filedialog

import random

def clearText():

ee1.delete(0, END)

ee2.delete(0, END)

ee3.delete(0, END)

ee4.delete(0, END)

e5.delete(0, END)

'''def clearTextE():

e2.delete(0, END)

e3.delete(0, END)

e4.delete(0, END)'''

key=0

y=0

def raise\_proposed(f):

f.tkraise()

def raise\_frame(f):

f.tkraise()

def finish():

r.destroy()

def start1():

raise\_frame(f1)

def start2():

raise\_frame(f2)

def start3():

raise\_frame(f3)

def start4():

raise\_frame(f4)

def start5():

raise\_frame(f5)

def start6():

raise\_frame(f6)

def start7():

raise\_frame(f7)

def start8():

raise\_frame(f8)

def start9():

raise\_frame(f9)

def start10():

raise\_frame(f10)

def start11():

raise\_frame(f11)

def start12():

raise\_frame(f12)

def start13():

raise\_frame(f13)

def start14():

raise\_frame(f14)

#----------------------------------------------------------------------------------------------------------------------------

cc=0

filename=""

def upload\_file():

global img,filename,cc

try:

cc=1

f\_types = [('Jpg Files', '\*.jpg')]

filename = filedialog.askopenfilename(filetypes=f\_types)

print(filename)

img = ImageTk.PhotoImage(file=filename)

img11.configure(image=img,width=260,height=200)

img11.image=img

img = Image.open(filename)

numpydata = asarray(img)

print(numpydata)

exe1.insert(0,numpydata)

except:

pass

def generateSecKey():

global key

key = np.load('data.npy') # load

e3.insert(0,key)

def upload\_file2():

global img,filename,cc

try:

cc=1

f\_types = [('Jpg Files', '\*.jpg')]

filename = filedialog.askopenfilename(filetypes=f\_types)

print(filename)

img = ImageTk.PhotoImage(file=filename)

img1.configure(image=img,width=360,height=300)

except:

pass

def upload\_file3():

global img,filename,cc

try:

cc=1

f\_types = [('Jpg Files', '\*.jpg')]

filename = filedialog.askopenfilename(filetypes=f\_types)

print(filename)

img = ImageTk.PhotoImage(file=filename)

img2.configure(image=img,width=360,height=300)

except:

pass

#--------------------------------------Decryption-1 -------------------------------------------------------------------------------

ck=1

def upload\_file1():

global img,filename,ck

if(ck==1):

try:

ck=1

f\_types = [('Jpg Files', '\*.png')]

filename = filedialog.askopenfilename(filetypes=f\_types)

img = ImageTk.PhotoImage(file=filename)

img2D.configure(image=img,width=360,height=300)

except:

pass

else:

messagebox.showinfo('Message Box','Click the previous button')

def fun14():

global vj,dec,bi,bi1,sig,val,ck

if(ck==1):

path= 'Images\img7.jpg'

try:

ck=2

fin = open(path, 'rb')

image = fin.read()

fin.close()

image = bytearray(image)

val=''

for i, values in enumerate(image):

image[i] = values

if(i<300):

val=val+str(D8B(values))+' '

if(i%22==0):

val=val+'\n'

fin = open(path, 'wb')

fin.write(image)

fin.close()

#print('Encryption Done...')

x16.set(val)

except Exception:

print('Error caught : ', Exception.\_\_name\_\_)

else:

messagebox.showinfo('Message Box','Click the previous button')

def Secret\_Key():

key = np.load('data.npy') # load

e3.insert(0,key)

def fun15():

global length,ck

if(ck==2):

try:

ck=3

file=askopenfile(mode='r',filetypes=[('All files','\*.txt')])

if file is not None:

length=file.read()

x17.set(str(length))

except:

pass

else:

messagebox.showinfo('Message Box','Click the previous button')

#----------------------------------------------------------------------------------------------------------------------------

def encrypt\_image():

global filename

global key

global y

img = Image.open(filename)

numpydata = asarray(img)

print(numpydata)

img = Image.open(filename)

numpydata = asarray(img)

image\_input = cv2.imread(filename, 0)# 'C:/Users/aakas/Documents/flower.jpg'

(x1, y) = image\_input.shape

image\_input = image\_input.astype(float) / 255.0

# print(image\_input)

yval123=int(ee4.get())

mu, sigma = 0, 0.1 # mean and standard deviation

key = np.random.normal(mu, sigma, (x1, y)) + np.finfo(float).eps

np.save('data.npy', key) # save

print("key is ",key)

# load the image and convert into

# numpy array

image\_encrypted = image\_input / key

e5.insert(0,(image\_encrypted))

np.save('image.npy', image\_encrypted) # save

#fn='image\_encrypted.jpg'

fnn = asksaveasfile(mode='w', defaultextension=".jpg")

if fnn is None:

return

print('Path ',fnn.name)

cv2.imwrite(fnn.name, image\_encrypted \* 255)

img = ImageTk.PhotoImage(file=fnn.name)

img111.configure(image=img,width=260,height=200)

img111.image=img

def decrypt\_image():

global filename

global image\_output

file1 = open("datas.npy","r+", encoding="utf-8")

key = np.load('data.npy') # load

#e3.insert(0,key)

dd=file1.read()

dd1=dd.split('==')

print("DD ",dd1)

#p=e1.get()

pk=e2.get()

if int(pk)==int(dd1[2]):

image\_encrypted = np.load('image.npy') # load

#image\_input = cv2.imread(filename, 0)

#(x1, y) = image\_input.shape

#image\_encrypted = image\_input.astype(float)

key = np.load('data.npy') # load

image\_output = image\_encrypted \* key

image\_output \*= 255.0

cv2.imwrite('image\_output.jpg', image\_output)

cv2.imwrite('image\_output.png', image\_output)

print('done')

e4.insert(0,image\_output)

#fname='C:/Users/sudheer/Desktop/Sudheerfinal/image\_output.jpg'

#img = ImageTk.PhotoImage(file='image\_output.png')

#img2.configure(image=img,width=360,height=300)

#upload\_file3()

import tkinter as tk

window = tk.Tk()

window.geometry('200x200')

img = tk.PhotoImage(file='image\_output.png') # has to be `file=`

tk.Label(image=img).pack()

window.after(5000, window.destroy) # `destroy` without `()`

window.mainloop()

else:

messagebox.showinfo('Message Box','Pls Check the keys')

def Y\_value():

global y

global pkey

p=ee1.get()

a=ee2.get()

pkey=ee3.get()

if p=='' or a=='' or pkey=='':

messagebox.showinfo('Message Box','Pls Enter the Values')

else:

p=int(p)

a=int(a)

ctr=0

for i in range(1,p+1):

if p%i==0:

ctr=ctr+1

if ctr!=2:

messagebox.showinfo('Message Box','P Value Should be Primary')

else:

pkey=int(pkey)

y=(a\*\*pkey)%p

print("Y Value is ",y)

print(p)

print(a)

print(pkey)

data=str(p)+"=="+str(a)+"=="+str(pkey)

with open('datas.npy', 'w', encoding="utf-8") as f:

f.write(data)

ee4.insert(0,str(y))

def K\_value():

global key

img = Image.open(filename)

numpydata = asarray(img)

image\_input = cv2.imread('test.jpg', 0)# 'C:/Users/aakas/Documents/flower.jpg'

(x1, y) = image\_input.shape

image\_input = image\_input.astype(float) / 255.0

# print(image\_input)

mu, sigma = 0, 0.1 # mean and standard deviation

key = np.random.normal(mu, sigma, (x1, y)) + np.finfo(float).eps

np.save('data.npy', key) # save

print("key is ",key)

e5.insert(0,key)

def Secret\_KeyD():

global key

e3.insert(0,key)

def image\_values():

global image\_output

e4.insert(0,image\_output)

def save\_cipher():

f\_types = [('Jpg Files', '\*.jpg')]

filename = filedialog.askopenfilename(filetypes=f\_types)

r = Tk()

f1 = Frame(r)

f2 = Frame(r)

f3 = Frame(r)

f4 = Frame(r)

f5 = Frame(r)

f6 = Frame(r)

f7 = Frame(r)

f8 = Frame(r)

f9 = Frame(r)

f10 = Frame(r)

f11 = Frame(r)

f12 = Frame(r)

f13 = Frame(r)

f14 = Frame(r)

f1.place(x = 0,y = 0,height=825, width=1360)

f2.place(x = 0,y = 0,height=825, width=1360)

f3.place(x = 0,y = 0,height=825, width=1360)

f4.place(x = 0,y = 0,height=825, width=1360)

f5.place(x = 0,y = 0,height=825, width=1360)

f6.place(x = 0,y = 0,height=825, width=1360)

f7.place(x = 0,y = 0,height=825, width=1360)

f8.place(x = 0,y = 0,height=825, width=1360)

f9.place(x = 0,y = 0,height=825, width=1360)

f10.place(x = 0,y =0,height=825, width=1360)

f11.place(x = 0,y =0,height=825, width=1360)

f12.place(x = 0,y =0,height=825, width=1360)

f13.place(x = 0,y =0,height=825, width=1360)

f14.place(x = 0,y =0,height=825, width=1360)

lab1001 = Label(f5)

lab1001.place(x=350,y=420)

rgbc=StringVar()

trk=StringVar()

x1=StringVar()

x2=StringVar()

x3=StringVar()

x4=StringVar()

x5=StringVar()

x6=StringVar()

x7=StringVar()

x8=StringVar()

x9=StringVar()

x10=StringVar()

x11=StringVar()

x12=StringVar()

x13=StringVar()

x14=StringVar()

x15=StringVar()

x16=StringVar()

x17=StringVar()

x18=StringVar()

x19=StringVar()

x20=StringVar()

x21=StringVar()

x22=StringVar()

x23=StringVar()

x24=StringVar()

x25=StringVar()

x26=StringVar()

x27=StringVar()

x28=StringVar()

x29=StringVar()

from math import sqrt

# Returns True if n is prime

def isPrime( n):

# Corner cases

if (n <= 1):

return False

if (n <= 3):

return True

# This is checked so that we can skip

# middle five numbers in below loop

if (n % 2 == 0 or n % 3 == 0):

return False

i = 5

while(i \* i <= n):

if (n % i == 0 or n % (i + 2) == 0) :

return False

i = i + 6

return True

""" Iterative Function to calculate (x^n)%p

in O(logy) \*/"""

def power( x, y, p):

res = 1 # Initialize result

x = x % p # Update x if it is more

# than or equal to p

while (y > 0):

# If y is odd, multiply x with result

if (y & 1):

res = (res \* x) % p

# y must be even now

y = y >> 1 # y = y/2

x = (x \* x) % p

return res

# Utility function to store prime

# factors of a number

def findPrimefactors(s, n) :

# Print the number of 2s that divide n

while (n % 2 == 0) :

s.add(2)

n = n // 2

# n must be odd at this point. So we can

# skip one element (Note i = i +2)

for i in range(3, int(sqrt(n)), 2):

# While i divides n, print i and divide n

while (n % i == 0) :

s.add(i)

n = n // i

# This condition is to handle the case

# when n is a prime number greater than 2

if (n > 2) :

s.add(n)

# Function to find smallest primitive

# root of n

def findPrimitive( n) :

s = set()

# Check if n is prime or not

if (isPrime(n) == False):

return -1

# Find value of Euler Totient function

# of n. Since n is a prime number, the

# value of Euler Totient function is n-1

# as there are n-1 relatively prime numbers.

phi = n - 1

# Find prime factors of phi and store in a set

findPrimefactors(s, phi)

# Check for every number from 2 to phi

for r in range(2, phi + 1):

# Iterate through all prime factors of phi.

# and check if we found a power with value 1

flag = False

for it in s:

# Check if r^((phi)/primefactors)

# mod n is 1 or not

if (power(r, phi // it, n) == 1):

flag = True

break

# If there was no power with value 1.

if (flag == False):

return r

# If no primitive root found

return -1

def GenAValue():

p=ee1.get()

if p=='':

messagebox.showinfo('Message Box','Pls Enter the P Value')

else:

p=int(p)

proot=findPrimitive(p)

ee2.insert(0,str(proot))

#-----------------------abstract------frame-2--------------------

ph33=ImageTk.PhotoImage(Image.open("Images\myabs.jpg"))

lab1 = Label(f2,image=ph33).place(x = 0, y = 0)

b7 = Button(f2, text = "Back",fg="black",bg="#FFFACD",font = "Helvetica 12 bold",height=1,width=10,command=start10).place(x = 550, y = 570)

#b8 = Button(f2, text = "Next",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command = start10).place(x = 1100, y = 60)

#-----------------------Proposed------frame-2--------------------

ph3=ImageTk.PhotoImage(Image.open("Images\slide4.jpg"))

lab1 = Label(f14,image=ph3).place(x = 0, y = 0)

b7 = Button(f14, text = "Back",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command=start1).place(x = 800, y = 580)

b8 = Button(f14, text = "Proceed",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command = start10).place(x = 600, y = 580)

#------------------StartPage-----------frame 1----------------------------------------------------------------

ph1=ImageTk.PhotoImage(Image.open("Images\Slide1.jpg"))

lab1 = Label(f1,image=ph1).place(x = -10, y = -10)

#b9 = Button(f10, text = "Proceed",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command = start14).place(x = 00, y = 450)

#but1 = Button(f1, text = "Contents",fg="black",bg="#FFFACD", font = "Helvetica 18 bold",height=1,width=10, command = start10).place(x = 500, y = 550)

but2 = Button(f1, text = "Close",fg="black",bg="#FFFACD", font = "Helvetica 18 bold",height=1,width=10,command=finish).place(x = 800, y = 550)

but3 = Button(f1, text = "Proceed",fg="black",bg="#FFFACD", font = "Helvetica 18 bold",height=1,width=10,command=start14).place(x = 600, y = 550)

#------------------------Menu-----frame-2--------------------

ph7=ImageTk.PhotoImage(Image.open("Images\empty.jpg"))

lab1 = Label(f10,image=ph7).place(x = 0, y = 0)

#ph70=ImageTk.PhotoImage(Image.open("Images\ED.jpg"))

#lab1 = Label(f10,image=ph70).place(x = 300, y = 340)

lab1 = Label(f10,text="El Gamal CRYPTOSYSTEM FOR IMAGE ENCRYPTION AND DECRYPTION", font = "Helvetica 24 bold").place(x = 100, y = 200)

b7 = Button(f10, text = "Home",fg="black",bg="#FFFACD", font = "Helvetica 20 bold",height=1,width=13,command=start1).place(x = 550, y = 350)

b8 = Button(f10, text = "Abstract",fg="black",bg="#FFFACD", font = "Helvetica 20 bold",height=1,width=13,command = start2).place(x = 550, y = 420)

#b9 = Button(f10, text = "Proposed",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command = proposed).place(x = 550, y = 280)

b10 = Button(f10, text = "Encryption",fg="black",bg="#FFFACD", font = "Helvetica 20 bold",height=1,width=13,command=start3).place(x = 550, y = 490)

b11 = Button(f10, text = "Decryption",fg="black",bg="#FFFACD", font = "Helvetica 20 bold",height=1,width=13,command = start6).place(x = 550, y = 560)

#-----------------------abstract------frame-2--------------------

#ph3=ImageTk.PhotoImage(Image.open("Images\Slide3.jpg"))

#lab1 = Label(f2,image=ph3).place(x = 0, y = 0)

b7 = Button(f2, text = "Back",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command=start1).place(x = 100, y = 760)

b8 = Button(f2, text = "Next",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command = start3).place(x = 1100, y = 760)

#-----------------------------frame-3--------------------

lab1 = Label(f3,image=ph7).place(x = 0, y = 0)

lab1 = Label(f3,text="Encryption", font = "Helvetica 34 bold").place(x = 100, y = 200)

b1 = Button(f3,text='Browse Image',font = "Helvetica 12 bold",command = upload\_file).place(x = 100, y = 300)

img11 =Label(f3)

img11.place(x=450,y=300)

#l1123 = Label(f3,text='Enter P Value',font = "Helvetica 12 bold").place(x = 200, y = 350)

exe1 = Entry(f3,font = "Helvetica 12 bold")

exe1.place(x = 250, y = 300)

l1 = Label(f3,text='Enter P Value',font = "Helvetica 12 bold").place(x = 100, y = 350)

ee1 = Entry(f3,font = "Helvetica 12 bold")

ee1.place(x = 250, y = 350)

l2 = Button(f3,text='Enter a Value',font = "Helvetica 12 bold",command=GenAValue).place(x = 100, y = 400)

ee2 = Entry(f3,font = "Helvetica 12 bold")

ee2.place(x = 250, y = 400)

l3 = Label(f3,text='Enter Private Key',font = "Helvetica 12 bold").place(x = 100, y = 450)

ee3 = Entry(f3,font = "Helvetica 12 bold")

ee3.place(x = 250, y = 450)

ll4 = Button(f3,text='Y Value',font = "Helvetica 12 bold",command =Y\_value ).place(x = 100, y = 500)

ee4 = Entry(f3,font = "Helvetica 12 bold")

ee4.place(x = 250, y = 500)

l5 = Button(f3,text='Cipher Values',font = "Helvetica 12 bold",command = K\_value).place(x = 100, y = 550)

e5 = Entry(f3,font = "Helvetica 12 bold")

e5.place(x = 250, y = 550)

#l6 = Label(f3,text='Secrete Key',font = "Helvetica 12 bold").place(x = 200, y = 600)

#e6 = Entry(f3,font = "Helvetica 12 bold")

#e6.place(x = 350, y = 600)

Button(f3,text='Encrypt&Save',fg="black",bg="#FFFACD",font = "Helvetica 12 bold",command=encrypt\_image).place(x = 660, y = 550)

img111 =Label(f3)

img111.place(x=900,y=320)

#tr = Label(f3,textvariable=trk,font = "Helvetica 12 bold")

#tr.place(x = 700, y = 650)

#Button(f3,text='Save',font = "Helvetica 12 bold",command=save\_cipher).place(x = 760, y = 550)

b7 = Button(f3, text = "Back",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command=start10).place(x = 860, y = 550)

#b8 = Button(f3, text = "Next",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command = start4).place(x = 1100, y = 760)

bb7 = Button(f3, text = "Clear",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command=clearText).place(x = 600, y = 600)

#-----------------------------frame-4--------------------

ph17=ImageTk.PhotoImage(Image.open("Images\empty.jpg"))

lab1 = Label(f4,image=ph17).place(x = 0, y = 0)

#lab1 = Label(f4,text="Encryption-2", font = "Helvetica 24 bold").place(x = 100, y = 200)

#Button(f4,text='Create ASCII values for message is to be sent.',font = "Helvetica 12 bold",command=fun2).place(x = 150, y = 150)

L1 = Label(f4,textvariable=x2,anchor='w',font = "Helvetica 12 bold",width=90).place(x = 150, y = 200)

L1 = Label(f4,textvariable=x3,anchor='w',font = "Helvetica 12 bold",width=90).place(x = 150, y = 250)

#Button(f4,text='Now adds this digits with Armstrong number',font = "Helvetica 12 bold",command=fun3).place(x = 150, y = 300)

L1 = Label(f4,textvariable=x4,anchor='w',font = "Helvetica 12 bold",width=90).place(x = 150, y = 350)

L1 = Label(f4,textvariable=x5,anchor='w',font = "Helvetica 12 bold",width=90).place(x = 150, y = 400)

L1 = Label(f4,textvariable=x6,anchor='w',font = "Helvetica 12 bold",width=90).place(x = 150, y = 450)

#Button(f4,text='Now adds this digits with Password',font = "Helvetica 12 bold",command=fun4).place(x = 150, y = 500)

L1 = Label(f4,textvariable=x7,anchor='w',font = "Helvetica 12 bold",width=90).place(x = 150, y = 550)

L1 = Label(f4,textvariable=x8,anchor='w',font = "Helvetica 12 bold",width=90).place(x = 150, y = 600)

L1 = Label(f4,textvariable=x9,anchor='w',font = "Helvetica 12 bold",width=90).place(x = 150, y = 650)

b7 = Button(f4, text = "Prev",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command=start3).place(x = 100, y = 760)

b8 = Button(f4, text = "Next",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command = start5).place(x = 1100, y = 760)

#Button(f4,text='Reset Button',font = "Helvetica 12 bold",command=funR).place(x = 1190, y = 500)

#-----------------------------frame-5--------------------

#-----------------------------frame-6--------------------

lab1 = Label(f6,image=ph17).place(x = 0, y = 0)

lab1 = Label(f6,text="Decryption", font = "Helvetica 12 bold").place(x = 50, y = 250)

b1 = Button(f6,text='Browse Cipher Image',font = "Helvetica 12 bold",command = upload\_file2).place(x = 200, y = 300)

img1 =Label(f6)

img1.place(x=700,y=300)

'''

l1 = Label(f6,text='Enter P Value',font = "Helvetica 12 bold").place(x = 200, y = 150)

e1 = Entry(f6,font = "Helvetica 12 bold")

e1.place(x = 350, y = 150)

'''

l2 = Label(f6,text='Enter Private Key',font = "Helvetica 12 bold").place(x = 200, y = 350)

e2 = Entry(f6,font = "Helvetica 12 bold")

e2.place(x = 350, y = 350)

l3 = Button(f6,text='Secert Key',font = "Helvetica 12 bold",command =Secret\_Key).place(x = 200, y = 400)

e3 = Entry(f6,font = "Helvetica 12 bold")

e3.place(x = 350, y = 400)

l4 = Button(f6,text='image values',font = "Helvetica 12 bold",command =image\_values).place(x = 200, y = 450)

e4 = Entry(f6,font = "Helvetica 12 bold")

e4.place(x = 350, y = 450)

Button(f6,text='Decrypt',font = "Helvetica 12 bold",command=decrypt\_image).place(x = 200, y = 500)

img2 =Label(f6)

img2.place(x=200,y=500)

b7 = Button(f6, text = "Back",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command=start10).place(x = 700, y = 600)

'''lab1 = Label(f6,image=ph17).place(x = 0, y = 0)

lab1 = Label(f6,text="Decryption", font = "Helvetica 12 bold").place(x = 50, y = 250)

b1 = Button(f6,text='Browse Cipher Image',font = "Helvetica 12 bold",command = upload\_file2).place(x = 200, y = 300)

img1 =Label(f6)

img1.place(x=700,y=300)

l1 = Label(f6,text='Enter P Value',font = "Helvetica 12 bold").place(x = 200, y = 150)

e1 = Entry(f6,font = "Helvetica 12 bold")

e1.place(x = 350, y = 150)

l2 = Label(f6,text='Enter Private Key',font = "Helvetica 12 bold").place(x = 200, y = 330)

e2 = Entry(f6,font = "Helvetica 12 bold")

e2.place(x = 350, y = 330)

l3 = Button(f6,text='Secert Key',font = "Helvetica 12 bold",command =Secret\_Key).place(x = 200, y = 550)

e3 = Entry(f6,font = "Helvetica 12 bold")

e3.place(x = 350, y = 550)

l4 = Button(f6,text='image values',font = "Helvetica 12 bold",command =image\_values).place(x = 200, y = 650)

e4 = Entry(f6,font = "Helvetica 12 bold")

e4.place(x = 350, y = 650)

Button(f6,text='Decrypt',font = "Helvetica 12 bold",command=decrypt\_image).place(x = 200, y = 750)

img2 =Label(f6)

img2.place(x=200,y=750)

b7 = Button(f6, text = "Home",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command=start10).place(x = 350, y = 850)

#bb7 = Button(f6, text = "Clear",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command=clearText).place(x = 600, y = 600)'''

#-----------------------------frame-7--------------------

#-----------------------------frame-8--------------------

ph12=ImageTk.PhotoImage(Image.open("Images\empty.jpg"))

lab1 = Label(f8,image=ph12).place(x = 0, y = 0)

b7 = Button(f8, text = "Prev",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command=start7).place(x = 100, y = 760)

b8 = Button(f8, text = "Next",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command = start9).place(x = 1100, y = 760)

#-----------------------------frame-9--------------------

#ph13=ImageTk.PhotoImage(Image.open("Images\Slide12.jpg"))

#lab1 = Label(f9,image=ph13).place(x = 0, y = 0)

b7 = Button(f9, text = "Prev",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command=start8).place(x = 100, y = 760)

b8 = Button(f9, text = "Next",fg="black",bg="#FFFACD", font = "Helvetica 12 bold",height=1,width=10,command = start1).place(x = 1100, y = 760)

#-----------------------------End--------------------

raise\_frame(f1)

r.geometry("1360x825+100+0")

r.title("el gamal image encryption AND DECRYPTION")

r.mainloop()

**TESTING**

# **6.TESTING**

Testing is the process of finding differences between the expected behavior specified by system models and the observed behavior of the system. Testing is a critical role in quality assurance and ensuring the reliability of development and these errors will be reflected in the codes the application should be thoroughly tested and validated.

Unit testing finds the differences between the object design model and its corresponding components. Structural testing finds differences between the system design model and a subset of integrated subsystems. Functional testing finds differences between the use case model and the system.

Finally, performance testing, finds differences between non-functional requirements and actual system performance. Form modeling point of view, testing is the attempt of falsification of the system with respect to the system models. The goal of testing is to design tests that exercise defects in the system and to reveal problems.

## **6.1 Testing Objectives**

Testing a large system is a complex system is a complex activity and like any complex activity. It must be breaking into smaller activities. Thus, incremental testing was performed on the project i.e., components and subsystems of the system were tested separately before integrating them to from the subsystem for system testing.

**6.2 Types of Testing**

## **6.2.1 Unit Testing**

Unit testing focuses on the building blocks of the software system that is the objects and subsystem. There are three motivations behind focusing on components. First unit testing reduces the complexity of overall test activities allowing focus on smaller units of the system, second unit testing makes it easier to pinpoint and correct faults given that few components are involved in the rest. Third unit testing allows parallelism in the testing activities, that is each component are involved in the test. Third unit testing allows parallelism in the testing activities that is each component can be tested independently of one another.

In this system each module of segment display, encryption and decryption are treated as individual units are tested individually. The following are some unit testing techniques.

## 

## **6.2.2 Integration Testing**

Integrated testing defects faults that have not been detected. During unit testing by focusing on small group on components two or more components are integrated and tested and once tests do not reveal any new faults, additional components are added to the group. This procedure allows testing of increasing more complex parts on the system while keeping the location of potential faults relatively small. I have used the following approach to implements and integrated testing.

Top-down testing strategy unit tests the components of the top layer and then integrated the components of the next layer down. When all components of the next layer have been tested together, the next layer is selected. This was repeated until all layers are combined and involved in the test. In this project, we first perform individual testing on the modules; Encryption and decryption which are individually verified are integrated for making a perfect project.

## **6.2.3 Validation Testing**

The system completely assembled as package, the interfacing have been uncovered and corrected, and a final series of software tests are validation testing. The validation testing is nothing but validation success when system functions in a manner that can be reasonably expected by the customer. The system validation had done by series of Black-box test methods.

In this project, validation is performed on each individual control. In encryption and decryption, if the receiver browses the cipher text and the appropriate key and the cipher text are not match then the system will generate the wrong plaintext. If the key and the cipher text are matched, then system will generate the plain text.

## **6.2.4 System Testing**

System testing of software is a testing conducted on a complete, integrated system to evaluate the system’s compliance with its specified requirements.

## **6.2.5 Usability testing**:

It is technique used to evaluate a product by testing it on users. It finds differences between the functional requirements and the system. With minimum knowledge of network security is enough for to use our project.

## **6.2.6 Performance testing:**

It covers a broad range of engineering or functional evaluations to meet measurable performance characteristics. The coding of the system involves a simple but effective method that can perform well even the user submits text with high security.

## **6.2.7 Installation testing**

It is a kind of quality assurance work in the software industry that focuses on what customers will need to do to install and set up the new software successfully. The system is installed in the target environment.

**6.3 Various types of Testing**

**6.3.1 White Box Testing**

We have given invalid click points while login. At that time the system should give response as Invalid click points and password didn't match. The system responds as we haveAfter entering click points the system should compare with the o registered click point values and it should return message as password matched if the click points are matched with the expected.

**6.3.2 Black Box Testing**

After entering click points the system should compare with the o registered click point values and it should return message as password matched if the click points are matched with the registered click points. The system responds as we expected.

## **6.4 Testing Plan**

Testing accounts for 45-75% of the typical project effort. It is also one of the most commonly underestimated activities on a project. A test plan is a document that answers the basic questions about your testing effort.

**6.5 Test Case Result**

**Test Case-1: Encryption Process**

**Test Case Type:** Black Box Testing

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Expected Value** | **Observed value** | **Result** |
| Enter the plain text for encryption process. | The plain text should be displayed on the screen. | Plain text is displayed. | Plain text |
| User clicks on encryption | Encryption process is done will get the Cipher text. | Will get the cipher text. | Cipher text. |

**Table 6.5.1: Test Case-1: Encryption Process**

**Test Case-2: Decryption Process**

**Test Case id: 02**

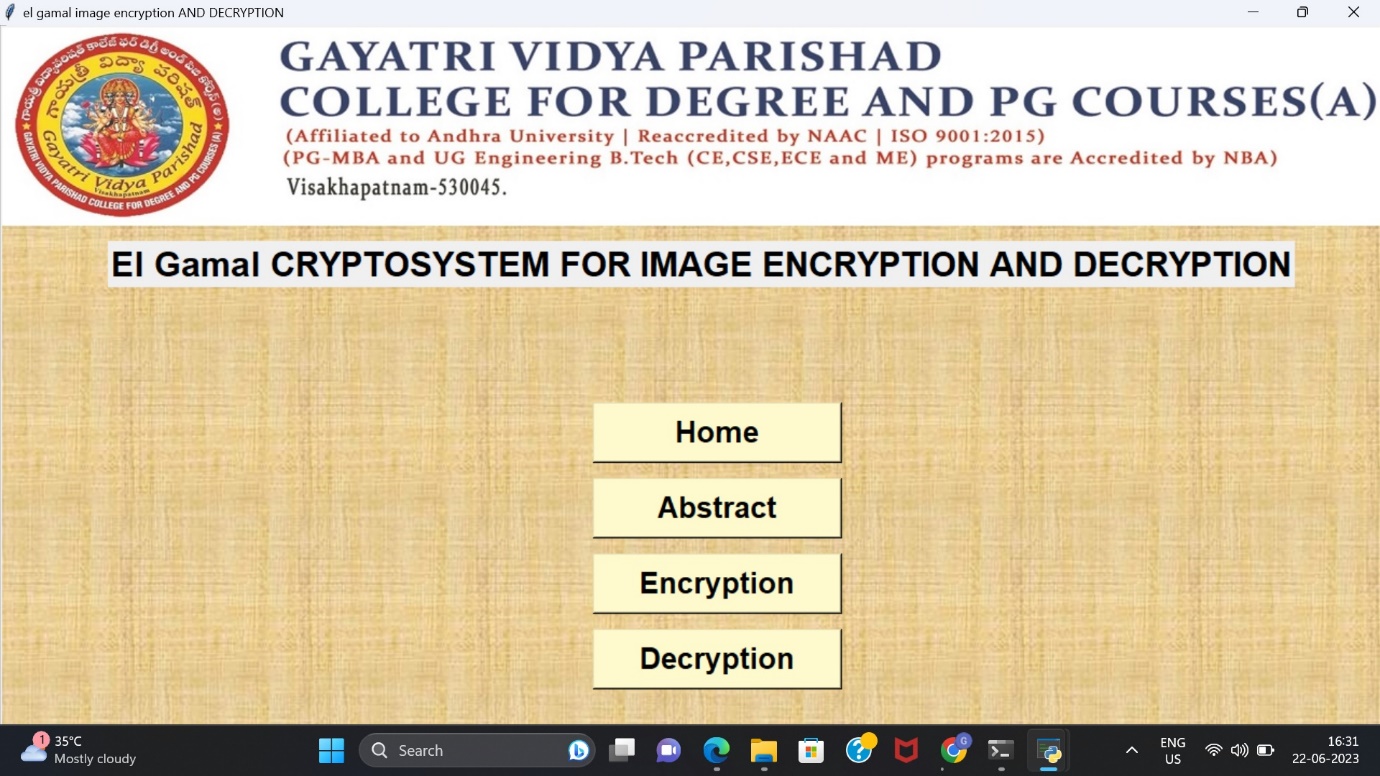
**Test Case Name:** Decryption Test

**Test Case Type:** Black Box Testing

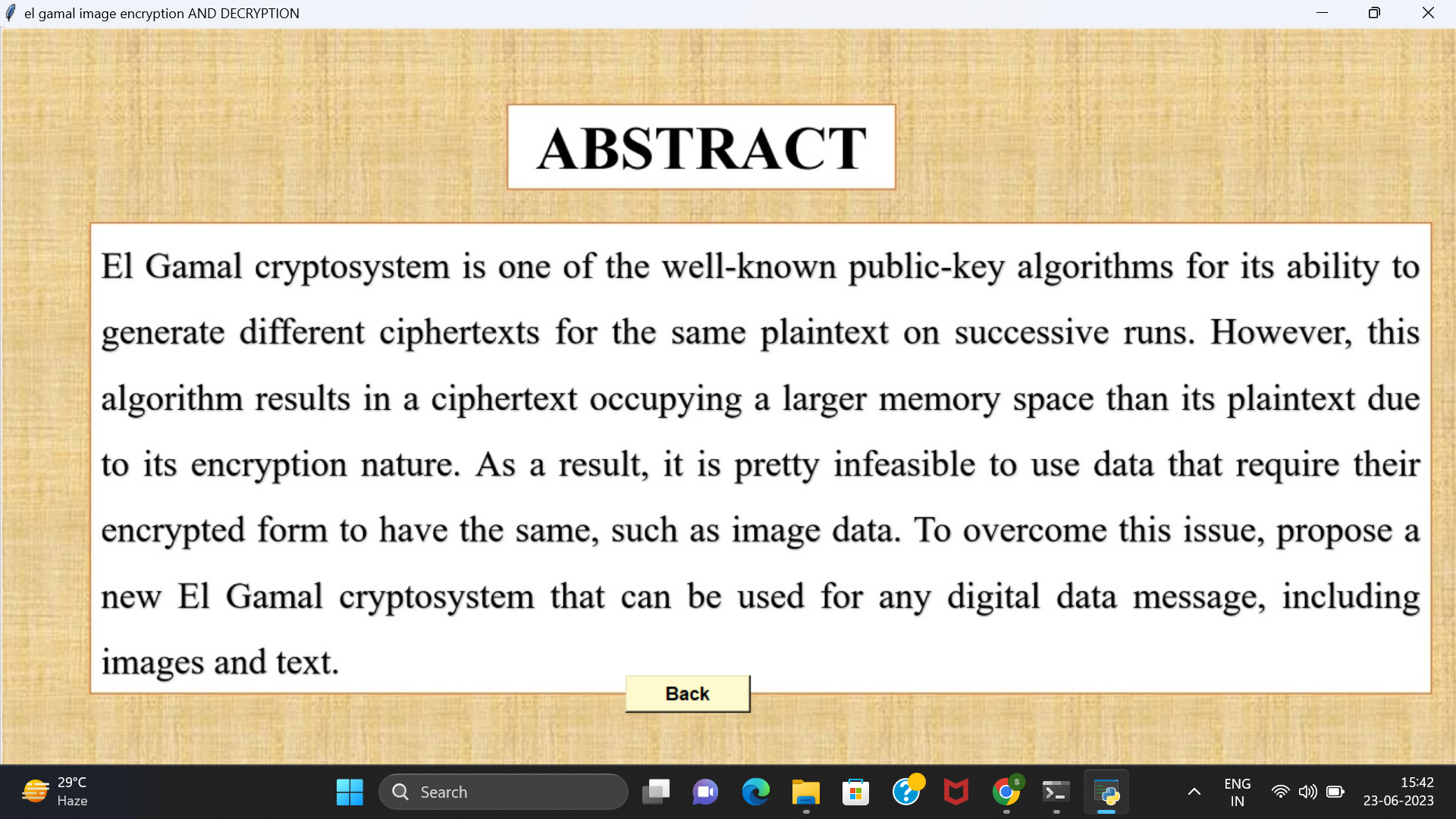
|  |  |  |  |
| --- | --- | --- | --- |
| **Decryption** | **Expected Value** | **Observed Value** | **Result** |
| User select the Cipher text. | Retrieve cipher text from the destination. | Will get the Cipher text. | Cipher text. |
| User clicks on decryption. | Decryption | Will get the Plain text. | Plain text. |

**Table 6.5.2 : Test Case-2: Decryption Process**

**SCREENS**

**Figure 7.1 Home Screen**

**Home Screen:** It is home screen,we can processed next stage here.

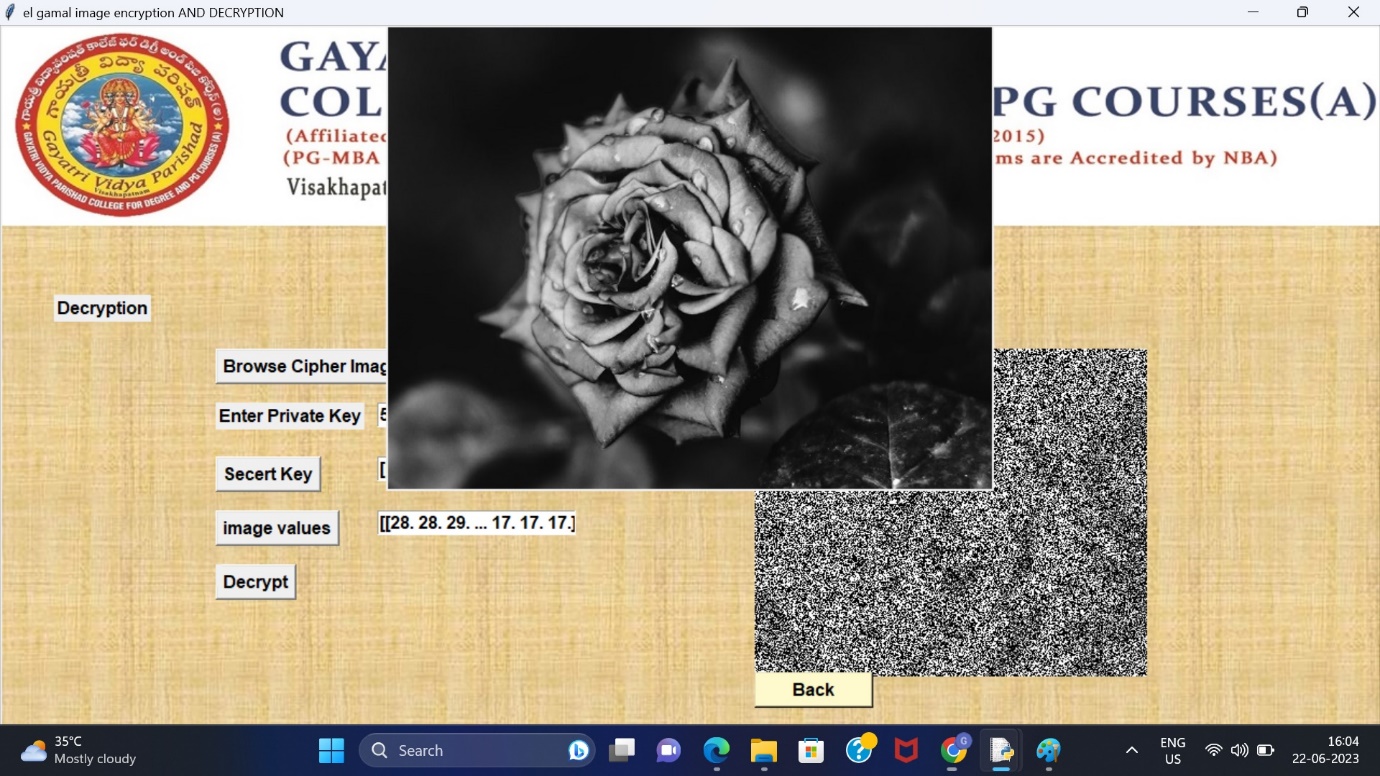
Figure 7.2 Abstract Screen

**Abstract Screen**: In this page we can see whole information about project **.**



**Figure 7.3 Encryption**

**Encrption Page :** The sender will enter the plain image then the encryption process will be performed by using public and private keys .It generates secret key to perform Xor operation then we get the Cipher Image.



**Figure 7.4 Decryption**

**Decryption:** The receiver will browse the ciphertext and private key used to generate secret key perform the Xor operation using we get the plain Image.

**CONCLUSION**

**8 CONCLUSION**

This proposes an efficient image cryptography method called new El Gamal, based on the old El Gamal. The proposed approach has several advantages over the old El Gamal. First, it results in an encrypted image having the same as the original image. Second, the time required for its encryption and decryption processes is much faster since it uses an XOR operation between each pixel and each byte generated randomly rather than using a multiplication operation with a large number used by the old El Gamal. Third, it can be applied to any digital data, such as text and images.

**REFERENCES**

**9 REFERENCES**

[1] Y. Luo, L. Cao, S. Qiu, H. Lin, J. Harkin, and J. Liu, “A chaotic mapcontrol-based and the plain image-related cryptosystem,” Nonlinear Dynamics, vol. 83, pp. 2293–2310, 2016, doi: 10.1007/s11071-015- 2481-7.

[2] L. Huang, S. Cai, X. Xiong, and M. Xiao, “On symmetric color image encryption system with permutation-diffusion simultaneous operation,” Optics and Lasers in Engineering, vol. 115, pp. 7–20, 2019, doi: 10.1016/j.optlaseng.2018.11.015.

[3] Y. Luo, R. Zhou, J. Liu, Y. Cao, and X. Ding, “A parallel image encryption algorithm based on the piecewise linear chaotic map and hyper-chaotic map,” Nonlinear Dynamics, vol. 93, pp. 1165–1181, 2018, doi: 10.1007/s11071-018-4251-9.

[4] Z. Hua, Y. Zhou, and H. Huang, “Cosine-transform-based chaotic system for image encryption,” Information Sciences, vol. 480, pp. 403– 419, 2019, doi: 10.1016/j.ins.2018.12.048.

[5] Y. Luo, S. Tang, X. Qin, L. Cao, F. Jiang, and J. Liu, “A double-image encryption scheme based on amplitude-phase encoding and discrete complex random transformation,” IEEE Access, vol. 6, pp. 77740– 77753, 2018, doi: 10.1109/ACCESS.2018.2884013.

**APPENDEX**

**10 APPENDX**

**10.1 List of Figures:**

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Figure Name** | **Page No** |
| Figure 3.4.1 | Use case diagram | 18 |
| Figure 3.4.2.1 | Sender sequence diagram | 22 |
| Figure 3.4.2.2 | Receiver sequence diagram | 22 |
| Figure 3.4.3.1 | State chart diagram for sender | 23 |
| Figure 3.4.3.2 | State chart diagram for receiver | 24 |
| Figure 7.1 | Home Screen | 69 |
| Figure 7.2 | Abstract Screen | 69 |
| Figure 7.3 | Encryption Screen | 70 |
| Figure 7.4 | Decryption Screen | 70 |

**10.2. List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Name** | **Page No.** |
| Table 3.1 | Sender Use Case | 19 |
| Table 3.2 | Key Generation Use Case | 19 |
| Table 3.3 | Encryption Use Case | 20 |
| Table 3.4 | Decryption Use Case | 20 |