A PROJECT REPORT ON

DESIGN AND ANALYSIS OF HYBRID CRYPTOGRAPHY SYSTEM FOR SECURE COMMUNICATION

Submitted in partial fulfillment of requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by

K. VAMSI MOHAN REDDY	**(19091A05H3)**
A. SAI THANU SREE	**(19091A05C7)**
L. VASAVI	**(19091A05H4)**

Under the Esteemed Guidance of

Dr. N. MADHUSUDHANA REDDY M. Tech, Ph.D.

Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)
AFFILIATED TO J.N.T UNIVERSITY ANANTAPUR. ACCREDITED BY NBA (TIER-1) & NAAC OF UGC. NEW DELHI, WITH A+ GRADE
RECOGNIZED UGC-DDU KAUSHAL KENDRA
NANDYAL-518501, (Estd-1995)

YEAR: 2022-2023

A PROJECT REPORT ON

DESIGN AND ANALYSIS OF HYBRID CRYPTOGRAPHY SYSTEM FOR SECURE COMMUNICATION

Submitted in partial fulfillment of requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by

K. VAMSI MOHAN REDDY	**(19091A05H3)**
A. SAI THANU SREE	**(19091A05C7)**
L. VASAVI	**(19091A05H4)**

Under the Esteemed Guidance of

Dr. N. MADHUSUDHANA REDDY M. Tech, Ph.D.

Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)
AFFILIATED TO J.N.T UNIVERSITY ANANTAPUR. ACCREDITED BY NBA (TIER-1) & NAAC OF UGC. NEW DELHI, WITH A+ GRADE
RECOGNIZED UGC-DDU KAUSHAL KENDRA
NANDYAL-518501, (Estd-1995)

YEAR: 2022-2023

Rajeev Gandhi Memorial College of Engineering &Technology

(AUTONOMOUS)

AFFILIATED TO J.N.T UNIVERSITY ANANTAPUR. ACCREDITED BY NBA (TIER-1) & NAAC OF UGC. NEW DELHI, WITH A+ GRADE
RECOGNIZED UGC-DDU KAUSHAL KENDRA
NANDYAL-518501, (Estd-1995)

(ESTD – 1995)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that **K. VAMSI MOHAN REDDY** (*19091A05H3*), **A. SAI THANU SREE** (*19091A05C7*) and **L. VASAVI** (*19091A05H4*) of IV- B. Tech II- semester, have carried out the major-project work entitled "**DESIGN AND ANALYSIS OF HYBRID CRYPTOGRAPHY SYSTEM FOR SECURE COMMUNICATION**" under the supervision and guidance of **Dr. N. MADHUSUDHANA REDDY,** Professor, CSE Department, in partial fulfillment of the requirements for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering** from **Rajeev Gandhi Memorial College of Engineering & Technology (Autonomous)**, Nandyal is a bonafied record of the work done by them during 2022-2023.

Project Guide

Dr. N. Madhusudhana Reddy M.Tech, Ph.D.

Professor, Dept. of CSE

Head of the Department

Dr. K. Subba Reddy M.Tech, Ph.D.

Professor, Dept. of CSE

Place: Nandyal
Date:

External Examiner

Candidate's Declaration

We hereby declare that the work done in this project entitled **"DESIGN AND ANALYSIS OF HYBRID CRYPTOGRAPHY SYSTEM FOR SECURE COMMUNICATION"** submitted towards completion of major project in IV *Year II Semester of B. Tech (CSE)* at the **Rajeev Gandhi Memorial College of Engineering & Technology**, Nandyal. It is an authentic record our original work done under the esteemed guidance of

Dr. N. Madhusudhana Reddy, Professor, department of **COMPUTER SCIENCE AND ENGINEERING**, RGMCET, Nandyal.

We have not submitted the matter embodied in this report for the award of any other Degree in any other institutions.

By

K.Vamsi mohan reddy (19091A05H3)

A. Sai Thanu sree (19091A05C7)

L. Vasavi (19091A05H4)

Dept. of CSE,RGMCET.

Place: Nandyal
Date:

ACKNOWLEDGEMENT

We manifest our heartier thankfulness pertaining to your contentment over our project guide **Dr. N. Madhusudhana Reddy** garu, Professor of Computer Science Engineering Department, with whose adroit concomitance the excellence has been exemplified in bringing out this project to work with artistry.

We express our gratitude to **Dr. K. Subba Reddy** garu, Head of the Department of Computer Science Engineering department, all teaching and non-teaching staff of the Computer Science Engineering Department of Rajeev Gandhi memorial College of Engineering and Technology for providing continuous encouragement and cooperation at various steps of our project.

Involuntarily, we are perspicuous to divulge our sincere gratefulness to our Principal, **Dr. T. Jaya Chandra Prasad** garu, who has been observed posing valiance in abundance towards our individuality to acknowledge our project work tangentially.

At the outset we thank our honourable Chairman **Dr. M. Santhi Ramudu** garu, for providing us with exceptional faculty and moral support throughout the course.

Finally, we extend our sincere thanks to all the **Staff Members** of CSE Department who have cooperated and encouraged us in making our project successful.

Whatever one does, whatever one achieves, the first credit goes to the **Parents** be it not for their love and affection, nothing would have been responsible. We see in every good that happens to us their love and blessings.

By

K.Vamsi mohan reddy (19091A05H3)

A. Sai Thanu sree (19091A05C7)

L. Vasavi (19091A05H4)

ABSTRACT

The Cryptography is the study of secure communication techniques which implies the craft of ensuring data by changing it into an unreadable format. The dynamite growth of the Internet has made an expanded familiarity with intrigue uncertainty issues. Even though security is the major constraint over the internet, numerous applications have been created and structured without considering fundamental destinations of data security that is confidentiality, authentication and protection. To forestall some undesirable clients or individuals to gain admittance to the data, cryptography is required.

The existing strategy utilizes a combination of Vigenere cipher and Polybius Square Cipher in its encryption process. The cipher text will initially be working on utilizing vigenere cipher. A picked key out of arbitrary will start the process. Toward the finish of the process, the subsequent cipher text then turns into key for the Polybius Square Cipher process. The key is used to work on the message which is plain text to create last cipher text. This process will wind up making the last ciphertext progressively hard to be broken.

However, the Polybius square has an identified drawback in the existing strategy. The Polybius square does not have a key for data encryption and decryption process making it vulnerable for cracks. Our proposed work deals with advancement of Polybius cipher in existing strategy. This modification allows extensive coverage for encrypting messages with characters using varying arrangements. This strategy also introduces a keyword to variate the arrangement of characters in the matrix. The keyword is plotted from top to bottom and left to right without repetitions. Any remaining letters not used in the keyword are then filled in the remaining cells in alphabetical order. This improved encryption results in ciphertext that is hard to be cracked utilizing existing cryptanalysis processes.

***Key Words**: Cryptanalysis, Cryptography, Encryption, Polybius Cipher, Vigenere Cipher*

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER - 1

INTRODUCTION

1.1 Introduction:

There are many aspects to security and many applications, ranging from secure ecommerce and payments to private communications and protecting passwords. One essential aspect for secure communications is that of cryptography. The Cryptography is the study of secure communication techniques which implies the craft of ensuring data by changing it into an unreadable format. The dynamite growth of the Internet has made an expanded familiarity with intrigue uncertainty issues. Even though security is the major constraint over the internet, numerous applications have been created and structured without considering fundamental destinations of data security that is confidentiality, authentication, and protection. To forestall some undesirable clients or individuals to gain admittance to the data, cryptography is required.

Encryption is defined as a systematic procedure of changing over plain message text into ciphertext. Encryption process needs any programmed encryption algorithm and a key to change over the plain message text into cipher. In the cryptography system encryption execute at the message sender side. Encryption executes the message at sender's side before sending it to the receiver.

Decryption is an opposite systematic procedure of encryption. It transforms the encrypted ciphertext into a message plaintext. In cryptography system decryption procedure execute at the receiver side. The process of decryption algorithm requires a couple of steps such as - a Decryption algorithm and a key.

Cryptography is extensively isolated into two classes relying on the Key. They are Asymmetric Key Encryption and Symmetric Key Encryption. In symmetric encryption techniques we use the same key for both encryption and decryption purpose. Asymmetric-key encryption using public and private keys, the public key is announced to all members while the private key is kept secure by the user. The sender uses the public key of the receiver to encrypt the message. The receiver uses his own private key to decrypt the message.

The NIST Computer Security Handbook [NIST95] defines the term computer security as follows: The protection afforded to an automated information system to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications).

Cryptography is one of the areas of science that studies about information security / data to avoid adverse effects due to misuse of information by irresponsible parties. Cryptography has an important role in maintaining the confidentiality of information both in the computer and at the time of transaction data.

1.2 Characteristics

Cryptography has several characteristics that make it an essential tool for securing communication and data. Here are some of the key characteristics of cryptography:

- **Confidentiality:** Cryptography provides a way to ensure the confidentiality of data by encrypting it so that it cannot be read by unauthorized parties.
- **Integrity:** Cryptography can ensure the integrity of data by providing methods to detect and prevent tampering or modification of the data.
- **Authentication:** Cryptography can provide authentication mechanisms that verify the identity of users or devices, preventing unauthorized access.
- **Non-repudiation:** Cryptography can provide non-repudiation, which ensures that a user cannot deny having performed an action or sent a message.
- **Key management:** Cryptography requires careful management of cryptographic keys used for encryption and decryption, ensuring that they are secure, unique, and properly stored.
- **Complexity:** Cryptography employs complex mathematical algorithms to ensure the security of data and communication channels, making it difficult for unauthorized parties to decrypt messages or gain access to sensitive data.
- **Adaptability:** Cryptography is an evolving field, and new cryptographic techniques and algorithms are continually being developed to address new threats and vulnerabilities.

Overall, cryptography provides a way to ensure the confidentiality, integrity, authentication, and non-repudiation of data and communication channels. It requires careful management of cryptographic keys and employs complex algorithms to provide security, and it is continually evolving to address new threats and vulnerabilities.

1.3 Features

Cryptography is the practice of secure communication in the presence of third parties or adversaries. It involves techniques for secure communication, data confidentiality, data integrity, and data authentication. Here are some of the key features of cryptography:

- **Confidentiality:** Cryptography provides a way to protect the confidentiality of data by encrypting it so that only authorized parties can access it. This is achieved by using a secret key or a public key, depending on the encryption algorithm.
- **Integrity:** Cryptography can ensure the integrity of data by detecting any unauthorized changes to the data. This is achieved by using techniques such as message authentication codes (MACs) or digital signatures.
- **Authentication:** Cryptography provides a way to authenticate the identity of communicating parties, ensuring that the message is sent by the intended sender and not a third party. This is achieved by using techniques such as digital certificates or public key infrastructure (PKI).
- **Non-repudiation**: Cryptography provides a way to ensure that a sender cannot deny having sent a message. This is achieved by using techniques such as digital signatures, which provide a non-repudiable proof of the sender's identity.
- **Key management**: Cryptography requires secure key management to protect the keys used for encryption and decryption. This includes key generation, distribution, and storage, as well as key revocation and rotation.
- **Steganography:** Cryptography can be used in conjunction with steganography, which is the practice of hiding data within other data, such as an image or a sound file. This provides an additional layer of security by making it more difficult for attackers to detect the presence of the hidden data.

1.4 Applications

Cryptography has a wide range of application areas in today's digital world. Here are some of the most common areas where cryptography is used:

- **Information Security**: Cryptography is used to secure data and communication channels to protect against unauthorized access, interception, or modification. It is used to encrypt data at rest and in transit, secure email, and messaging, and protect online transactions.
- **Cybersecurity:** Cryptography is a fundamental component of cybersecurity, providing methods for secure authentication, access control, and threat detection. It is used to secure networks, cloud services, and mobile devices, and to detect and prevent cyber-attacks.
- **Banking and Finance:** Cryptography is used to secure financial transactions and protect sensitive information such as account numbers, credit card details, and personal identification information. It is used to encrypt online banking transactions, secure ATM transactions, and authenticate users.
- **E-commerce:** Cryptography is used to secure e-commerce transactions, including online purchases, payments, and transfers. It is used to encrypt credit card and other payment information, provide secure authentication, and prevent fraud.
- **Military and Defense**: Cryptography is used extensively by the military and defense organizations to secure classified information, communication channels, and weapon systems. It is used to protect military intelligence, secure command, and control systems, and prevent unauthorized access.
- **Healthcare:** Cryptography is used to secure electronic medical records, protect patient privacy, and ensure the confidentiality of sensitive medical information. It is used to encrypt medical data, authenticate users, and prevent unauthorized access.
- **Internet of Things (IoT):** Cryptography is used in IoT devices to secure communication between devices, protect data privacy, and prevent unauthorized access. It is used to encrypt sensor data, authenticate devices, and prevent cyber-attacks.

Why Cryptography?

There are several reasons why cryptography is a crucial component to consider:

- **Security:** Cryptography provides a way to secure data and communication channels, protecting against unauthorized access, interception, or modification. Incorporating cryptography into a project can help ensure that sensitive data remains confidential, and communication is secure, protecting both the project and its users.
- **Compliance:** Depending on the industry or regulatory requirements, implementing cryptography may be mandatory to comply with regulations, laws, and standards such as GDPR, HIPAA, or PCI-DSS.
- **Trust:** By incorporating cryptography into a project, you can provide users with confidence in the security and privacy of their data. This can build trust and increase user engagement, as they feel secure knowing their information is protected.
- **Competitive advantage**: By implementing cryptography, you can differentiate your project from competitors by providing additional security and privacy measures to your users. This can lead to increased customer loyalty and a competitive advantage in the marketplace.
- **Futureproofing:** Cryptography is an evolving field, and incorporating the latest cryptographic techniques into a project can help ensure that it remains secure and relevant in the future. This can reduce the risk of security breaches and ensure the project's long-term viability.

1.5 Advantages

Cryptography provides numerous advantages, making it an essential tool for securing data and communication channels. Here are some of the key advantages of cryptography:

- **Confidentiality:** Cryptography provides a way to ensure the confidentiality of data by encrypting it, preventing unauthorized access to sensitive information.
- **Integrity:** Cryptography provides a way to ensure the integrity of data by detecting and preventing tampering or modification of the data.

- **Authentication:** Cryptography provides authentication mechanisms that verify the identity of users or devices, preventing unauthorized access.
- **Non-repudiation:** Cryptography provides non-repudiation, which ensures that a user cannot deny having performed an action or sent a message.
- **Security:** Cryptography provides a way to secure data and communication channels, protecting against unauthorized access, interception, or modification.
- **Compliance:** Cryptography may be required to comply with regulatory requirements, laws, and standards such as GDPR, HIPAA, or PCI-DSS.
- **Trust:** Cryptography can build trust by providing users with confidence in the security and privacy of their data, increasing user engagement and loyalty.
- **Competitive Advantage:** Cryptography can provide a competitive advantage by differentiating a product or service from competitors and increasing customer loyalty.
- **Futureproofing:** Cryptography is an evolving field, and incorporating the latest cryptographic techniques into a product or service can help ensure its long-term viability and security.

1.6 Summary

Cryptography is a fundamental tool for ensuring the security and privacy of digital communications and transactions, and its features play a crucial role in achieving these objectives. Cryptography has a broad range of applications in various industries and plays a critical role in securing digital communication, protecting sensitive information, and preventing cyber-attacks.

Incorporating cryptography into a project is essential to ensure the security, compliance, and trust of users' data and communication channels. It can provide a competitive advantage, future-proofing the project, and ensuring its long-term success. Overall, cryptography provides a way to secure data and communication channels, ensure confidentiality, integrity, authentication, and non-repudiation, and comply with regulatory requirements. It can build trust, provide a competitive advantage, and future-proof a product or service.

CHAPTER - 2

LITERATURE REVIEW

2.1 Introduction to Literature Survey

A literature survey is a comprehensive review of published literature in a specific field or area of study. It involves critically analysing and synthesising the existing research to identify the gaps, limitations, and opportunities for further research.

Literature surveys are essential for several reasons: Identify gaps and opportunities for research: A literature survey helps identify the gaps and limitations in the existing research, which can inform the development of new research questions and hypotheses. Additionally, a literature survey can identify emerging research trends and opportunities for future research.

Avoid duplication of research: Conducting a literature survey can help researchers avoid duplicating existing research by identifying the research that has already been conducted in the area. This can help save time and resources and prevent unnecessary repetition of research.

Evaluate and critique existing research: A literature survey involves critically evaluating and synthesising the existing research, which can help identify the strengths and weaknesses of the research. This can help researchers identify areas where further research is needed to address the limitations of the existing research.

Inform research methodology: Literature surveys can help researchers identify the most appropriate research methods and techniques for their research questions. By evaluating the existing research methods and techniques used in the area, researchers can identify the most appropriate methods and techniques for their research.

Build on existing research: By synthesising the existing research, researchers can identify the most promising areas for further research and build on the existing research to advance the field.

2.2 Related works

[1] F. M. S. Ali and F. H. Sarhan, "Enhancing security of vigenere cipher ` by stream cipher," International Journal of Computer Applications, vol. 100, no. 1, pp. 1–4, 2014.

They proposed a method which combines the substitution cipher(Vigenere cipher) and stream cipher. This method makes the encryption and decryption processes very difficult in absence of a secret binary random key which improve the security of data. But the drawback of this approach is, the use of random binary numbers of key with different letters of plain text does not completely hides the correlation between plain text and cipher text and results in high complexity due to binary calculations.

The advantages of this work are described below:

- The paper addresses a well-known weakness of the Vigenere cipher, which is its vulnerability to frequency analysis attacks. The proposed method makes it more difficult to perform such attacks by introducing the randomness of a stream cipher.
- The paper is well-organized and easy to follow, with clear explanations of the Vigenere cipher and the proposed enhancement.
- The experimental results presented in the paper show that the proposed method improves the security of the Vigenere cipher.

The Disadvantages of this work are described below:

- The paper does not provide a detailed analysis of the security of the proposed method. While it is clear that the method makes it harder to perform frequency analysis attacks, it is unclear how strong the resulting cipher is against other types of attacks.

- The paper does not compare the proposed method with other existing methods for enhancing the security of the Vigenere cipher, which makes it hard to assess its novelty and effectiveness.
- The paper does not provide a real-world use case or example of the proposed method in action.

[2] Mendrofa, Elwinus & Purba, Elwin & Siahaan, Boy & Sembiring, Rahmat Widia. (2017). Collaborative Encryption Algorithm Between Vigenere Cipher, Rotation of Matrix (ROM), and One Time Pad (OTP) Algoritma. Advances in Science, Technology and Engineering Systems Journal. 2. 13-21. 10.25046/aj020503.

They proposed a collaborative method in their paper which combines three ciphers namely Vigenere cipher, Rotation Of Matrix(ROM) and One-Time Pad(OTP). The encryption on this method obtain ciphertext to be sent to the recipient where there are two keys that are sent, among others, the key to decrypt the plaintext and the key to decrypt the key. The drawback of this approach is, Encryption and decryption process have to done through a long route, it increases time complexity of the method and still the method is modest and simple.

The advantages of this work are described below:

- The paper presents a novel approach to encryption by combining three different ciphers, which may provide stronger security compared to using a single cipher alone.
- The proposed collaborative encryption algorithm is described in detail and the authors provide a step-by-step explanation of the encryption and decryption processes.
- The experimental results presented in the paper show that the proposed method has a high level of security, with a very low probability of decryption success.

The disadvantages of this work are described below:

- The paper does not provide a thorough analysis of the security of the proposed method. While the experimental results are promising, it is unclear how strong the cipher is against other types of attacks, such as known-plaintext attacks or ciphertext-only attacks.
- The paper does not compare the proposed method with other existing methods for encryption, which makes it hard to assess its novelty and effectiveness.
- The paper does not provide a real-world use case or example of the proposed method in action.

[3] M. Maity, "A modified version of polybius cipher using magic square and western music notes," International Journal for Technological Research In Engineering, ISSN, pp. 2347– 4718, 2014

They proposed a new method in their paper which describes a modified version of polybius cipher using magic square and western music notes. In this method, a unique 6x6 magic square and heptatonic increasing C major scale C–D–E–F–G–A–B is used for encryption. First any unique 6x6 normal magic square is taken which is arranged with integers from 1 to 36. Next all the alphanumeric characters are placed like this: as A is first letter, it is placed in cell with digit 1, second letter B in the cell with digit 2 and so on. The huge combinations of keys make guessing or cracking of the keys very much impossible. This method is suitable for minimal secret text based data transfer.

The advantages of this work are described below:

- The paper presents a unique and creative approach to encryption by combining a magic square and Western music notes with the Polybius cipher.
- The proposed modified Polybius cipher is described in detail, with clear explanations of the encryption and decryption processes.
- The paper includes experimental results that demonstrate the effectiveness of the proposed method in terms of its security.

The disadvantages of this work are described below:

- The paper does not provide a thorough analysis of the security of the proposed method. While the experimental results are promising, it is unclear how strong the cipher is against other types of attacks, such as brute-force attacks or chosen-plaintext attacks.
- The paper does not compare the proposed method with other existing methods for encryption, which makes it hard to assess its novelty and effectiveness.
- The paper does not provide a real-world use case or example of the proposed method in action.

[4] C. Bhardwaj, "Modification of vigenere cipher by random numbers, ` punctuations & mathematical symbols," Journal of Computer Engineering (IOSRJCE) ISSN, pp. 2278–0661, 2012.

In this paper he proposed a modified method in their paper which describes modification of vigenere cipher by random numbers, `punctuations & mathematical symbols. This method induces that the encryption key must be any type of characters like mathematical symbols, punctuations, numbers etc. and an arbitrary number is introduced for the key to spread the spectrum and also strengthens the key used. But this modified method leads to high computation and ambiguity as different symbols used in key needs to be used in different way for encryption and decryption.

The advantages of this work are described below:

- The paper addresses a well-known weakness of the Vigenere cipher, which is its vulnerability to frequency analysis attacks. The proposed modification makes it more difficult to perform such attacks by introducing randomness and symbols.
- The proposed modification is described in detail, with clear explanations of the encryption and decryption processes.
- The paper includes experimental results that demonstrate the effectiveness of the proposed method in terms of its security.

The disadvantages of this work are described below:

- The paper does not provide a thorough analysis of the security of the proposed method. While the experimental results are promising, it is unclear how strong the cipher is against other types of attacks, such as known-plaintext attacks or chosen-plaintext attacks.
- The paper does not compare the proposed method with other existing methods for encryption, which makes it hard to assess its novelty and effectiveness.
- The paper does not provide a real-world use case or example of the proposed method in action.

[5] Arroyo, Jan Carlo & Delima, Allemar Jhone. (2020). A Modified Polybius Cipher with a New Element-in-Grid Sequencer. 9. 3249-3255. 10.30534/ijatcse/2020/119932020

They proposed a new method in their paper which describes a Modified Polybius Cipher with a New Element-in-Grid Sequencer. This method enhances Polybius square based on ASCII code values with the introduction of key. With the key, the elements inside the matrix are reorganized by shifting cells based on its ASCII decimal values. A cell shifting is done for every plaintext character encoded; thus, a new matrix is generated for each iteration[5]. This method offers enhanced security with the use of secret key and dynamically generated matrices. This scheme generates a new matrix for every character to be encrypted which in turn increases time complexity.

The advantages of this work are described below:

- The paper presents a new and creative approach to the Polybius cipher by introducing a new element-in-grid sequencer that enhances the security of the cipher.
- The proposed modification is described in detail, with clear explanations of the encryption and decryption processes.
- The paper includes experimental results that demonstrate the effectiveness of the proposed method in terms of its security.

The disadvantages of this work are described below:

- The paper does not provide a thorough analysis of the security of the proposed method. While the experimental results are promising, it is unclear how strong the cipher is against other types of attacks, such as brute-force attacks or chosen-plaintext attacks.
- The paper does not compare the proposed method with other existing methods for encryption, which makes it hard to assess its novelty and effectiveness.
- The paper does not provide a real-world use case or example of the proposed method in action.

2.3 Challenges

Some of the common challenges observed are described below:

- Lack of thorough analysis: Many of the papers do not provide a thorough analysis of the security of the proposed cipher or algorithm. This makes it difficult to assess how strong the cipher is against various types of attacks, such as brute-force attacks or chosen-plaintext attacks.
- Limited scope: Some papers focus on a specific type of cipher and propose a modification to enhance its security, but they do not compare the proposed method with other existing methods or provide a real-world use case. This limits the scope and applicability of the proposed method.
- Lack of novelty: In some cases, the proposed modifications do not introduce a significant level of novelty or improvement over existing ciphers or algorithms. This may result in a lack of interest from the academic community and industry.
- Incomplete description: Some papers do not provide a complete description of the proposed modification, making it difficult for readers to understand the encryption and decryption processes. This can hinder the reproducibility and evaluation of the proposed method.

2.4 Summary

The literature review highlights different approaches to enhance the security of the Vigenere and Polybius ciphers by incorporating additional security measures, such as stream ciphers, magic squares, random numbers, and new element-in-grid sequencers. The combination of these ciphers in a hybrid cryptography system can further improve the security of data transmission but in addition to it those approaches arises some practical imbalances like increased computation and time complexity.

CHAPTER - 3

SYSTEM DESIGN

3.1 Problem Definition

The problem addressed in the design of the Hybrid Cryptography System based on Vigenere Cipher and Polybius Cipher is the need for a secure and efficient encryption method to protect sensitive data during transmission or storage. This project aims to design and implement a hybrid cryptography system that uses the Vigenere Cipher and keyed Polybius Cipher together to encrypt and decrypt data. The system should be easy to use, efficient, and resistant to known cryptographic attacks, requiring a good understanding of the Vigenere Cipher and Polybius Cipher algorithms and familiarity with Python programming and cryptographic concepts.

3.2 Proposed Methodology

The proposed methodology for the design of the Hybrid Cryptography System based on Vigenere Cipher and Enhanced Polybius Cipher involves the following steps:

- Research and understand the Vigenere Cipher and Polybius Cipher algorithms, as well as cryptographic concepts and Python programming.
- Design the system architecture, including the user interface and the encryption/decryption algorithms.
- Implement the encryption and decryption algorithms using Python programming and the required libraries/modules.
- Test the system to ensure it is secure, efficient, and resistant to known cryptographic attacks.
- Refine the system based on testing results and feedback, and ensure the system meets the project objectives.
- Document the system design, implementation, and testing in a clear and organized manner.
- Present the system and findings to relevant stakeholders, including peers and project sponsors.

By following this methodology, we can design and implement a Hybrid Cryptography System based on Vigenere Cipher and Polybius Cipher that is secure, efficient, and meets the project objectives.

3.2.1 Algorithms

In this project we are using two algorithms namely:

a) Vigenere Cipher
b) Polybius Cipher

Vigenere Cipher:

Vigenere Cipher is a method of encrypting alphabetic text. It uses a form of polyalphabetic substitution. The encryption of the original text is done using the Vigenère square or Vigenère table. The table consists of the alphabets written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar Ciphers. At different points in the encryption process, the cipher uses a different alphabet from one of the rows. The alphabet used at each point depends on a repeating keyword.

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

Figure.3.1 Vigenere Square

Here are the steps to encrypt a message using the Vigenere cipher:

- Choose a keyword or phrase.
- Repeat the keyword or phrase until it is the same length as the plaintext message.
- Write the keyword above the plaintext message.
- For each letter in the plaintext message, find the corresponding letter in the keyword above it.
- Determine the shift amount by finding the position of the key letter in the alphabet.
- Shift the plaintext letter by the shift amount to obtain the ciphertext letter.
- Repeat steps 4-6 for each letter in the plaintext message.

To decrypt a message encrypted with the Vigenere cipher, you simply reverse the process. Instead of shifting each letter by a certain amount, you shift it back by the same amount to recover the original plaintext message.

Advantages of Vigenere Cipher:

Stronger than simple substitution ciphers: The Vigenere cipher is stronger than simple substitution ciphers because it uses a repeating key, making it harder to break.

Customizable key: The key used in the Vigenere cipher can be any length and can consist of any combination of characters, making it highly customizable.

Easy to use: The Vigenère cipher is relatively easy to use and does not require any special equipment or software.

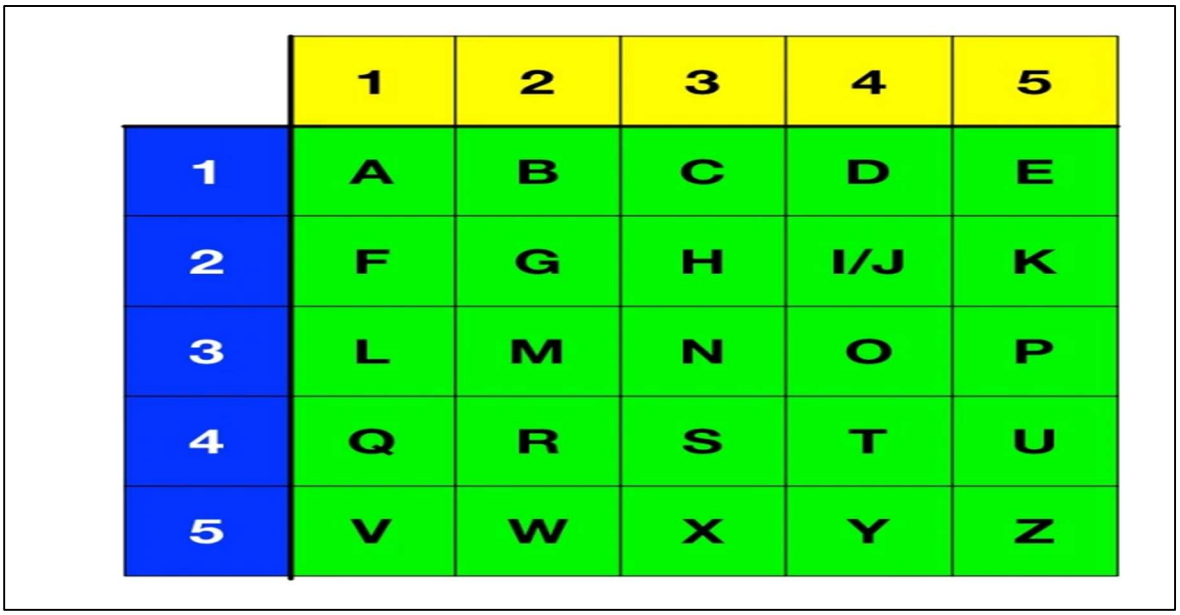
Resistance to frequency analysis: The Vigenere cipher is resistant to frequency analysis because each letter in the plaintext can be encrypted by any of the letters in the key, making it difficult to detect patterns in the ciphertext.

However, it is worth noting that the Vigenere cipher is still vulnerable to certain attacks, such as the Kasiski examination and the Friedman test. Additionally, the security of the cipher can be compromised if the key is not kept secret or if the key is reused.

Polybius Cipher:

The Polybius Cipher is a substitution cipher that encodes pairs of letters rather than individual letters. It uses a 5x5 grid of letters, with the letters I and J combined into one cell.

The letters are usually arranged in a Polybius square with the alphabet written out in a grid format.



	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Figure.3.2 Polybius Square

Here are the steps to encode a message using the Polybius Cipher:

- Write out the plaintext message.
- Divide the message into pairs of letters. If there is an odd number of letters, add a dummy letter (such as X) to the end to make an even number of pairs.
- Find each pair of letters in the Polybius square and write down the row and column number for each letter.
- Write out the row and column numbers for each pair of letters in the message.
- The resulting string of numbers is the ciphertext.

To decrypt a message using the Polybius Cipher, simply reverse the process. Divide the ciphertext into pairs of numbers, find the corresponding letters in the Polybius square, and write out the plaintext message.

Advantages of Polybius Cipher

The Polybius cipher has the following advantages:

- It is a simple and easy-to-understand cipher, making it ideal for beginners in cryptography.
- It is more secure than some of the other classical ciphers, such as the Caesar cipher or the Atbash cipher.
- The cipher is resistant to frequency analysis attacks, which are common in many other classical ciphers.
- It can be extended to support a larger range of characters and alphabets, making it suitable for use in different languages and writing systems.

3.2.2 System Architecture

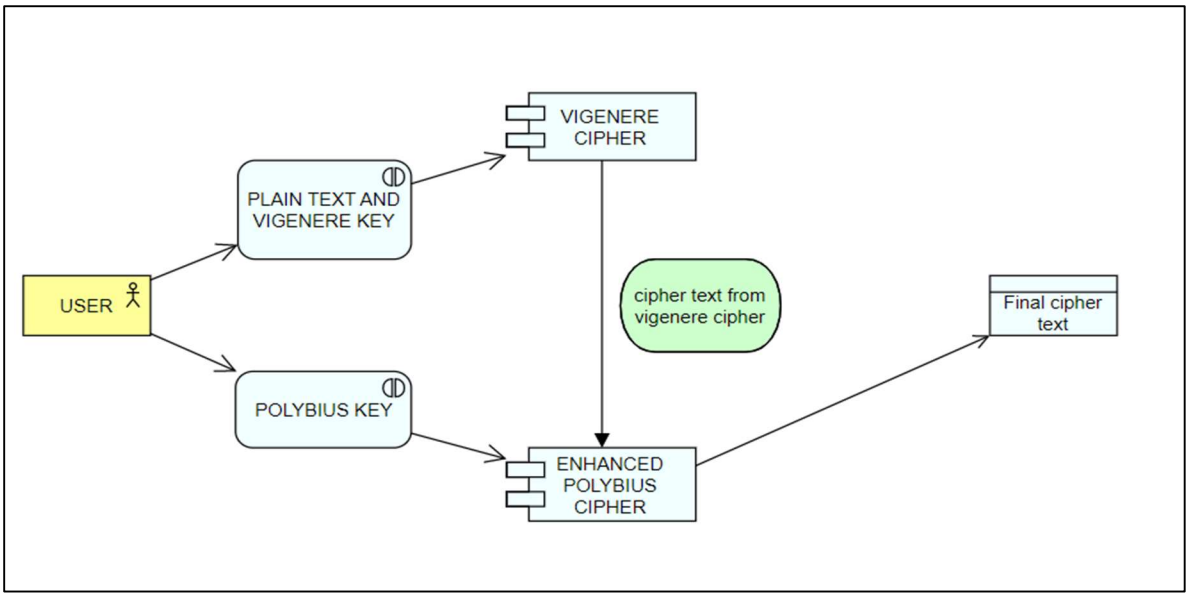


Figure.3.3 System Architecture

The system architecture of a hybrid cryptography system that uses Vigenere cipher and enhanced Polybius cipher typically consists of the following components:

Input Module: The input module is responsible for accepting the plaintext message that needs to be encrypted. The plaintext message can be in the form of a file or text input.

Vigenere Cipher Module: The Vigenere Cipher module takes the plaintext message and a key as input and generates a Vigenere ciphertext. The Vigenere Cipher uses a polyalphabetic

substitution method, where each letter in the plaintext message is shifted by a different number of places based on the key.

Keyed Polybius Cipher Module: The Keyed Polybius Cipher module takes the Vigenere ciphertext and a key as input and generates a Polybius ciphertext. The Keyed Polybius Cipher uses a substitution method, where each letter in the Vigenere ciphertext is replaced with a two-digit number that corresponds to its row and column in a Polybius square. The key is used to generate a unique Polybius square.

Output Module: The output module is responsible for displaying the Polybius ciphertext to the user. The ciphertext can be in the form of a file or text output.

The security of the hybrid cryptography system depends on the strength of the Vigenere and Polybius ciphers and the key used to encrypt the message. By combining the two ciphers, the hybrid cryptography system can provide an additional layer of security to protect the plaintext message from unauthorized access.

3.3 Working Principles

The working principles of a hybrid cryptography system that uses Vigenere cipher and keyed Polybius cipher can be explained as follows:

Encryption Process:

- The plaintext message is first input into the Vigenere Cipher module along with a Vigenere key.
- The Vigenere Cipher module generates a Vigenere ciphertext by shifting each letter of the plaintext message by a different number of places based on the key.
- The Vigenere ciphertext is then input into the Keyed Polybius Cipher module along with a Polybius key.
- The Keyed Polybius Cipher module generates a Polybius ciphertext by replacing each letter of the Vigenere ciphertext with a two-digit number that corresponds to its row and column in a Polybius square. The Polybius square is generated using the Polybius key.

Decryption Process:

- The Polybius ciphertext is first input into the Keyed Polybius Cipher module along with the same Polybius key used for encryption.
- The Keyed Polybius Cipher module generates a Vigenere ciphertext by replacing each two-digit number in the Polybius ciphertext with the corresponding letter in a Vigenere square. The Vigenere square is generated using the Vigenere key.
- The Vigenere ciphertext is then input into the Vigenere Cipher module along with the same Vigenere key used for encryption.
- The Vigenere Cipher module generates the original plaintext message by reversing the shifting process used during encryption.

The working principles of the hybrid cryptography system depend on the strength of the Vigenere and Polybius ciphers and the keys used for encryption and decryption. The Vigenere Cipher provides a polyalphabetic substitution method that makes it difficult for an attacker to decipher the ciphertext without the key. The Keyed Polybius Cipher provides a substitution method that replaces each letter in the Vigenere ciphertext with a two-digit number that makes it difficult for an attacker to identify the original message. By combining these two ciphers, the hybrid cryptography system can provide a stronger level of security to protect the plaintext message from unauthorized access.

3.3.1 Modules

The modules we proposed for a hybrid cryptography system that uses Vigenere cipher and keyed Polybius cipher are given below, generally these are not actual modules these acts as internal source for implementation.

Vigenere Cipher Module: This module is responsible for encrypting the plaintext message using the Vigenere cipher. It takes the plaintext message and a Vigenere key as inputs, and generates a Vigenere ciphertext as output.

Keyed Polybius Cipher Module: This module is responsible for encrypting the Vigenere ciphertext generated by the Vigenere Cipher module using the keyed Polybius cipher. It

takes the Vigenere ciphertext and a Polybius key as inputs, and generates a Polybius ciphertext as output.

Polybius Square Generator Module: This module is responsible for generating the Polybius square used by the Keyed Polybius Cipher module. It takes the Polybius key as input, and generates a Polybius square as output.

Decryption Module: This module is responsible for decrypting the Polybius ciphertext to obtain the original plaintext message. It takes the Polybius ciphertext, the Polybius key, and the Vigenere key as inputs, and generates the original plaintext message as output.

These modules work together to provide a hybrid cryptography system that combines the strengths of the Vigenere and Polybius ciphers to provide a higher level of security. The Vigenere Cipher module and the Keyed Polybius Cipher module are used for encryption, while the Polybius Square Generator module is used for generating Polybius square using given keyword. The Decryption module is used to reverse the encryption process and obtain the original plaintext message.

3.4 Introduction to UML:

A UML diagram is a diagram based on the UML (Unified Modelling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

Each UML diagram is designed to let developers and customers view a software system from a different perspective and in varying degrees of abstraction. UML diagrams commonly created in visual modelling tools include.

3.4.1 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. It describes the attributes and operations of a class and also the constraints imposed on the system

3.4.2 Use case Diagram

A use case diagram at its simplest is a representation of user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other of diagrams as well. The use cases are represented by either circles or ellipses.

3.4.3 Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join etc.

3.4.4 Sequence Diagram

Sequence diagrams in UML shows how object interact with each other and the order those interactions occur. It's important to note that show the interactions for a particular scenario. The processes are represented vertically, and interactions are show as arrows.

3.5 UML Diagrams:

The figure 3.4 shows the use case diagram of the system. The system consists of mainly two actors or users interacts with it, they are sender and receiver. The use cases they involved and their interaction with the system are depicted in the below figure.

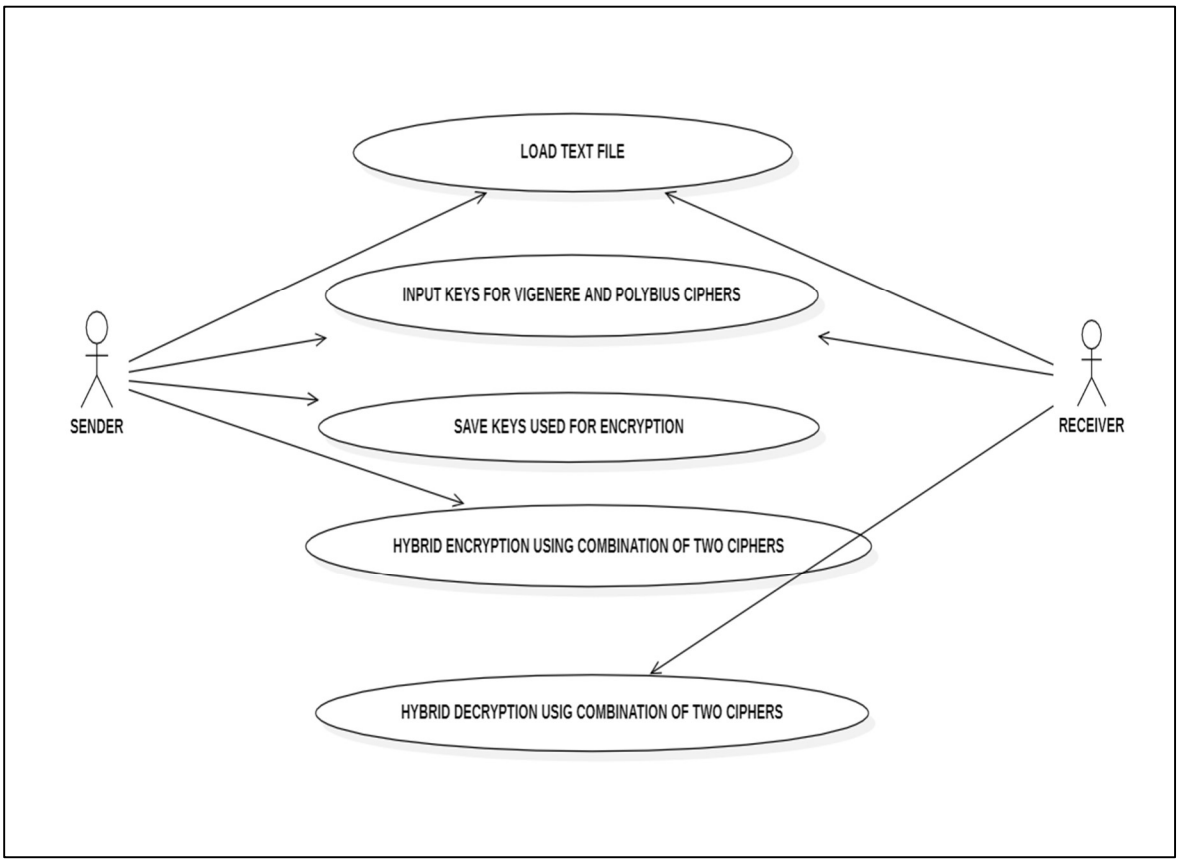


Figure.3.4 Usecase Diagram

The figure 3.5 shows the class diagram of the system. It represents static view of application. The below figure describes attributes and operations of classes and the constraints imposed on the system.

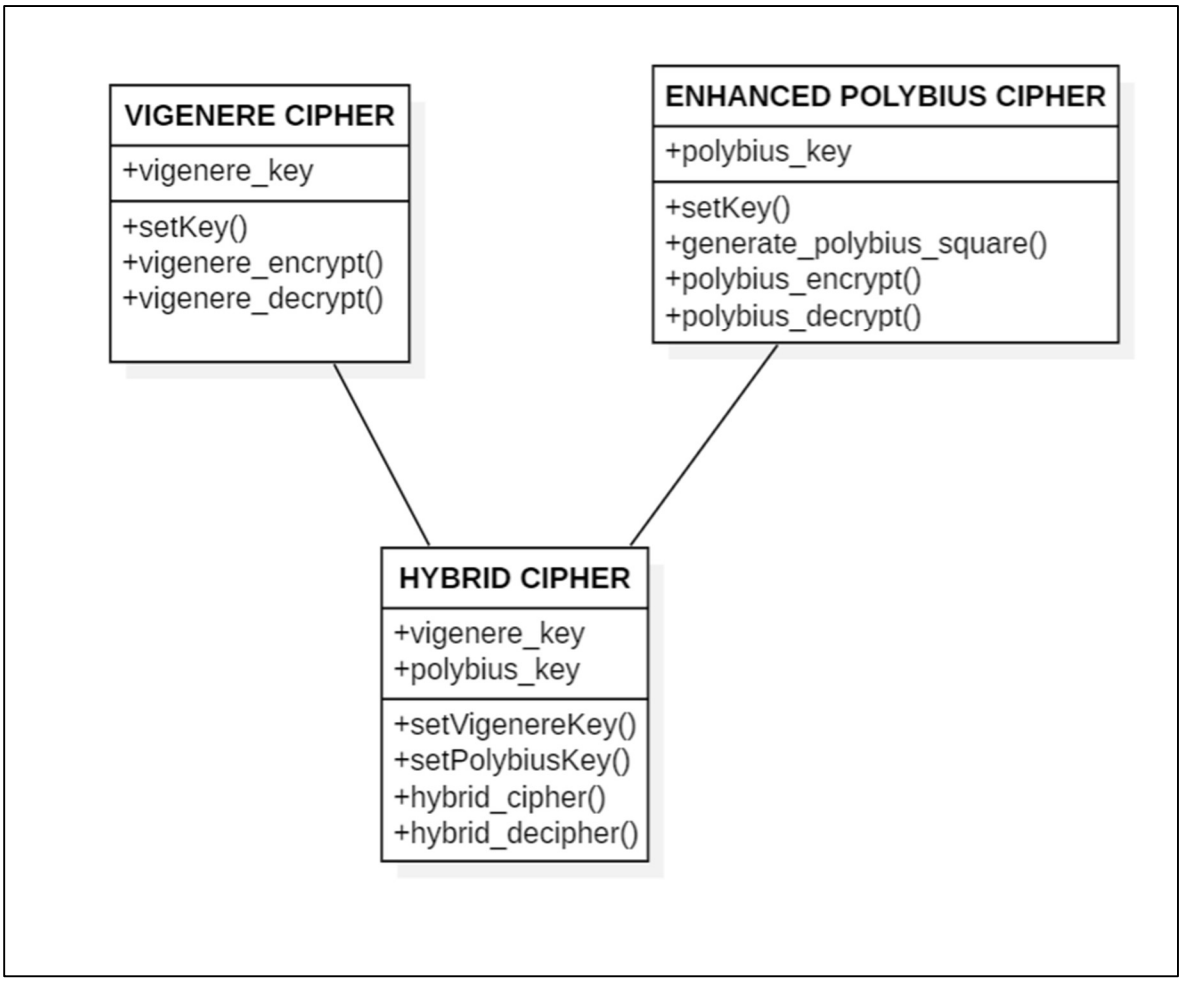


Figure.3.5 Class Diagram

The figure 3.6 shows the sequence diagram of the system. The below figure depicts the processes involved and the sequences of actions undergone during implementing the system.

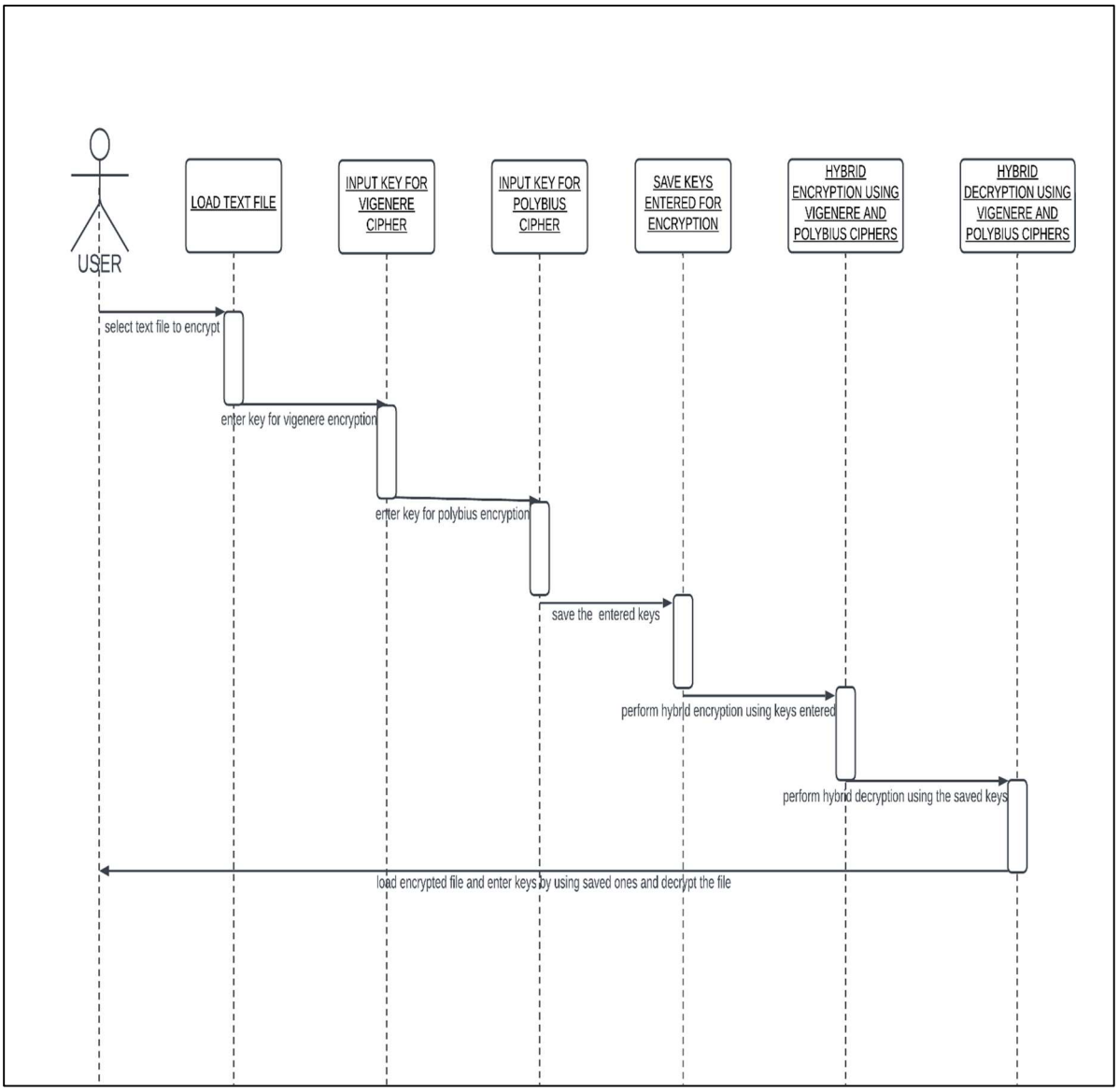


Figure.3.6 Sequence Diagram

The figure 3.7 shows the activity diagram of the system. The below figure basically represents the flow from one activity to another activity in the working of system. In the below figure each activity can be described as an operation of the system.

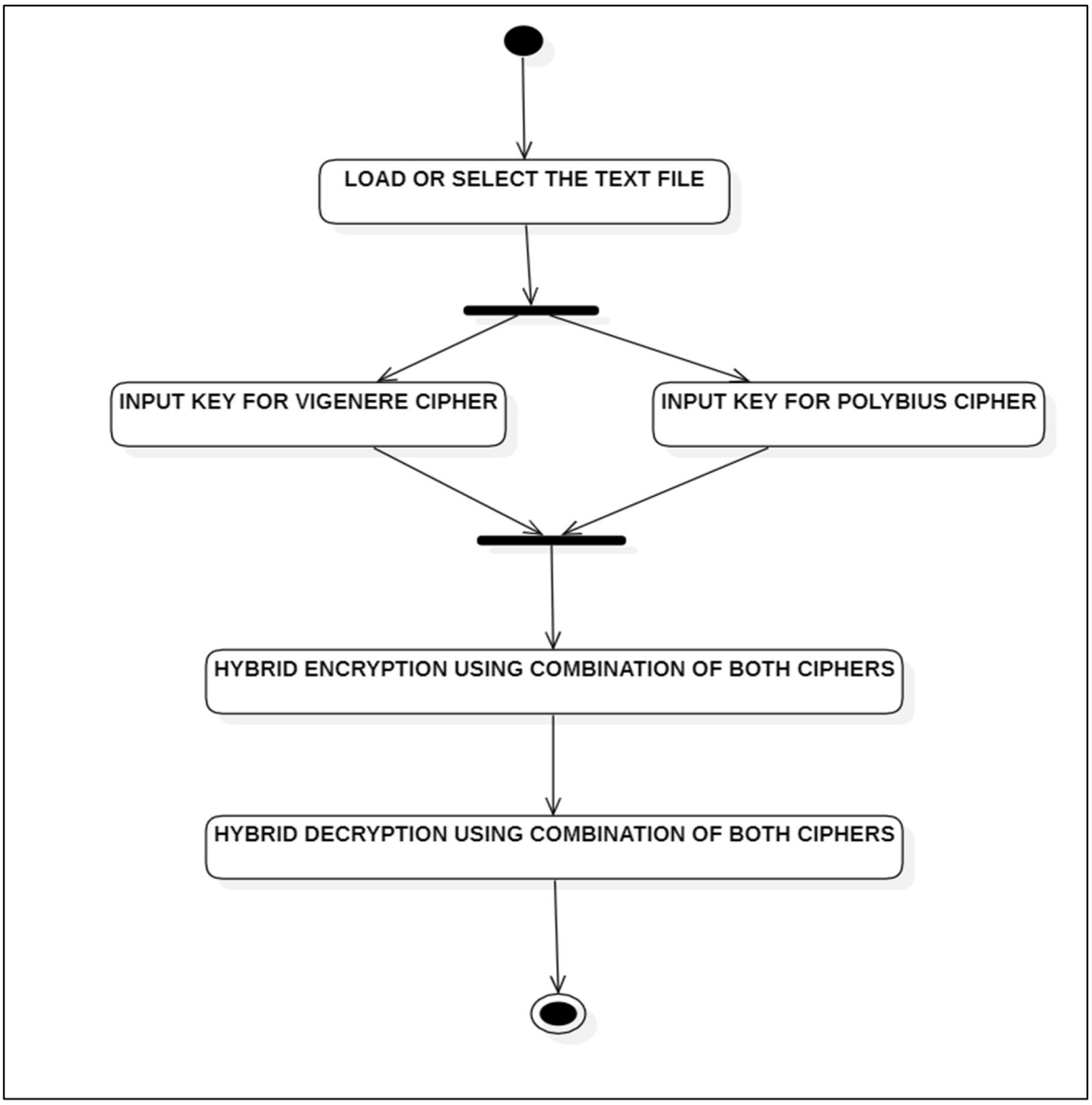


Figure.3.7 Activity Diagram

3.6 System Requirement Specification(SRS):

A System Requirements Specification (SRS) (also known as a Software Requirements Specification) is a document or set of documentation that describes the features and behavior of a system or software.

3.6.1 Requirement analysis

Requirements Analysis is the process of defining the expectations of the users for an application that is to be built or modified. It involves all the tasks that are conducted to identify the need of different stakeholders.

Requirement Analysis is a software engineering task that bridges the gap between system level software allocation and software design. Requirement Analysis is to specify software function and performance indicates software interface with other system elements.

Requirement Analysis of the system starts with the specification given by the user. This user-required specification could be formal or informal. In formal method, the user clearly states the purpose of the software. This acts as the good basis for the software engineers for the requirement analysis. In informal method purpose and the outputs are not clearly specified by the user. The responsibility is on the software engineer to get the purpose and outputs by interacting more the user.

3.6.2 Software Requirement Specification

A software requirements specification, a requirements specification for a software system is complete description of the behavior of a system to be developed and may include a set of use cases that describe interactions the user will have with the software. In addition, it also contains non-functional requirements. Non-functional requirements impose constraints on the design or implementation.

The software requirements specification document enlists all necessary requirements that are required for the project development. To derive the requirements, we need to have clear and through understanding of the products to be developed. This is prepared after detailed communications with the project team and customer.

A Software Requirements Specification minimizes the time and effort required by developers to achieve desired goals and minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs, and human users in a wide variety of real-world situations.

3.6.3 Functional Requirements

In software engineering a functional requirement defines as a function of system or its components where a function is described as a specification of behavior between output and input. It provides the following actions.

Technical Issues:

Many software projects have failed due to an incomplete or inaccurate analysis process, especially technical issues. It is a very key step in process.

Risk Analysis:

Project Risk Analysis is for the estimates of known accuracy and risk on capital investment projects. The main challenge is to implement an effective working model of hybrid encryption, define and monitor the risk impacts, analyse the probability of risk occurrence, mitigate the negative impact of risks, and monitor the course of the project with risks and uncertainties.

Performance Requirements:

The project has the following performance requirements :

- The prime requirement is that no error condition causes a project to exit abruptly.
- Any error occurred in any process should return an understandable error message.
- The response should be fast and accurate, the analysis should not be confused at any point of time about action that is happening. The system performance should be adequate.

3.6.4 Non-Functional Requirements

Reliability: The project is guaranteed to provide reliable results for every test data. The system shall operate with expected potency. The accuracy of the hybrid model should be adequate so that when it it is deployed no downtime should be occurred due to its working.

Usability: This type of hybrid models can be deployed in websites, applications, Servers, Embedded systems for secure communication. The usage of this systems are mainly in army and raw agencies for transmission of secret codes and messages.

Scalability: The need for scalability has been a driver for much of the technology innovations of the past few years. The system should perform with same time complexity even when performed on large test data. This adaption makes system scalable towards any kind of data used in encryption and decryption.

Maintainability: Maintainability is the ability to make changes to the product over time. The system should be maintainable to be consistent in performance. For example, it should be able to cope with another better ciphers in future to meet required performance and accuracy.

3.6.5 Software Requirements

- Programming language : Python
- Tool : PyCharm
- Operating System : Windows 7 or above (using windows 11)

3.6.6 Hardware Requirements

- Hard Disk Drive : 256 GB(using 512 GB)
- Processor : intel i3 or above(using core i5)
- RAM : 4 gb(min)

CHAPTER – 4

SYSTEM IMPLEMENTATION

Systems implementation is the process of defining how the information system should be built (i.e., physical system design), ensuring that the information system is operational and used, ensuring that the information system meets quality standard (i.e., quality assurance).

4.1 Overview of the Tools and Technologies used for project:

The Main tools are used in this project are:

1. Python
    - Tkinter (Python Library)
    - String (Python Library)
    - Functools (Python Library)
2. PyCharm (Software)

4.1.1 Python:

Python is a high-level programming language that is designed to be simple to read and use. It's free to use, even for commercial purposes, because it's open-source. Python is available for Mac, Windows, and Unix systems, as well as Java and .NET virtual machines.

Python, like Ruby or Perl, is a scripting language that is frequently used to create Web applications and dynamic Web content. Python is also supported by a variety of 2D and 3D imaging programs, allowing users to write custom plug-ins and extensions. GIMP, Inkscape, Blender, and Autodesk Maya are examples of applications that support a Python API. Python scripts (.PY files) can be parsed and executed right away. They can also be saved as compiled programs (.PYC files), which are commonly used as programming modules that other Python programs can reference.

4.1.2 History

The programming language Python was conceived in the late 1980s, and its implementation was started in December 1989 by Guido van Rossum at CWI in the Netherlands as a successor to ABC capable of exception handling and interfacing with the Amoeba operating system. Van

Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community.

Python 2.0 was released on October 16, 2000, with many major new features, including a cycle-detecting garbage collector (in addition to reference counting) for memory management and support for Unicode. However, the most important change was to the development process itself, with a shift to a more transparent and community-backed process. Python 3.0, a major, backwards- incompatible release, was released on December 3, 2008, after a long period of testing. Many of its major features have also been backported to the backwards-compatible, though now-unsupported, Python 2.6 and 2.7.

4.1.3 Python Features

Python has a variety of useful features that distinguish it from other programming languages. It supports object-oriented programming, procedural programming, and memory allocation that is dynamic. A few essential features are listed below:

Easy to Learn and Use:

Python is a simple programming language to learn when compared to other programming languages. Its syntax is simple and like that of the English language. The semicolon and curly brackets are not used; instead, the indentation defines the code block. For beginners, it is the recommended programming language.

Expressive Language:

Python can perform complex tasks with just a few lines of code. For example, to run the hello world program, simply type print ("Hello World"). It will only require one line of code to run, whereas Java or C will require multiple lines.

Interpreted Language:

Python is an interpreted language, which means that each line of a Python program is executed separately. The benefit of being an interpreted language is that debugging is simple and portable.

Cross-platform Language:

Python can run on a variety of platforms, including Windows, Linux, UNIX, and Macintosh. As a result, we can say that Python is a portable programming language. It allows programmers to create software for multiple competing platforms by writing only one program.

Free and Open Source:

Python is a free programming language that anyone can use. On its official website, www.python.org, it is freely available. It has a large community all over the world working hard to create new Python modules and functions. The Python community welcomes contributions from anyone. "Anyone can download its source code without paying a penny," says open-source.

Object-Oriented Language:

Python supports object-oriented programming, which introduces the concepts of classes and objects. It allows for an inheritance, polymorphism, and encapsulation, among other things. The object-oriented method aids programmers in writing reusable code and developing applications with less code.

Extensible:

It means that other languages, such as C/C++, can be used to compile the code, allowing us to use it in our Python code. It converts the program to byte code, which can be run on any platform.

Large Standard Library:

It offers a diverse set of libraries for a variety of fields, including machine learning, web development, and scripting. Tensor flow, Pandas, NumPy, Keras, and Pytorch are just a few examples of machine learning libraries. Python web development frameworks include Django, Flask, and Pyramids.

GUI Programming Support:

For the development of a desktop application, a graphical user interface is used. The libraries used to develop the web application are String, Tkinter, and Function Tools.

Integrated:

It's simple to integrate with languages like C, C++, and JAVA, among others. Python, like C, C++, and Java, executes code line by line. It makes it easy to debug the code.

Embeddable:

Other programming languages' code can be used in the Python source code. Python source code can also be used in other programming languages. It can embed other languages into our code.

Dynamic Memory Allocation:

We don't need to specify the variable's data type in Python. When we assign a value to a variable, the variable's memory is automatically allocated at run time. If the integer value 15 is assigned to x, we don't need to write int x = 15. Simply write x = 15 on a piece of paper.

4.1.4 Python Modules:

Tkinter:

The Tkinter module is a built-in Python module that provides a simple way to create graphical user interfaces (GUIs) for Python applications. It provides a variety of widgets and functions that can be used to create windows, menus, buttons, labels, text boxes, and other elements of a GUI. In the context of the hybrid cryptography system based on the Vigenere Cipher and Polybius Cipher, we can use the Tkinter module to create a simple GUI that allows users to input plaintext, keys, and perform encryption and decryption operations. The GUI can also display the ciphertext and decrypted plaintext.

The Tkinter module can be used to create a GUI with a layout that consists of frames and widgets. The frames can be used to group related widgets together, while the widgets can be used to display text, input fields, and buttons.

For example, we can use the Tkinter module to create a GUI that includes the following widgets:

- A text box for entering the plaintext.
- A text box for entering the key.
- Radio buttons for selecting between encryption and decryption operations.
- A button for performing the selected operation.
- A text box for displaying the ciphertext or decrypted plaintext.

By using the Tkinter module, we can create a user-friendly interface for the hybrid cryptography system that allows users to interact with the system without needing to know how to program or use the command line. The GUI can also provide visual feedback on the status of the system and any errors that may occur during operation.

string:

The string module in Python provides a set of constants that are useful when working with strings. The module defines various string constants such as ASCII lowercase, ASCII uppercase, digits, hex digits, oct digits, printable, and punctuation. These constants can be used to represent different sets of characters that may be needed in string processing.

In the implementation of the hybrid cryptography system based on the Vigenere Cipher and Polybius Cipher, we can use the string module to manipulate strings, such as converting strings to uppercase or lowercase, stripping leading and trailing whitespace, and replacing certain substrings. This module can be particularly useful in the implementation of the Vigenere Cipher, where we need to manipulate the plaintext and key strings to generate the ciphertext.

functools:

The functools module in Python provides a set of higher-order functions (functions that take other functions as arguments or return functions as results) that are useful when working with functions. The module defines various function manipulation functions such as partial, reduce, wraps, and cmp_to_key.

In the implementation of the hybrid cryptography system based on the Vigenere Cipher and Polybius Cipher, we may not necessarily need to use the functools module directly. However, some of the higher-order functions provided by this module can be useful when working with functions in general. For example, the partial () function can be useful if we want to create a new function that is a modified version of an existing function with some arguments fixed. The wraps () function can be useful if we want to create a new function that has the same attributes as an existing function.

4.1.5 PyCharm:

PyCharm is an integrated development environment (IDE) for Python programming language. It is developed by JetBrains and provides code analysis, debugging, testing, version control integration, and other features that help developers to write and manage their Python code more efficiently.

PyCharm provides several features that are useful for developing Python applications, including:

- Code editor with syntax highlighting, code completion, and code refactoring tools.
- Integrated debugging tools, including breakpoints and stack traces.
- Version control integration with popular version control systems like Git, SVN, and Mercurial.
- Support for running and debugging Python scripts in various environments, including local machines, remote servers, and Docker containers.
- Interactive Python console for testing code snippets and experimenting with Python commands.
- Integration with popular testing frameworks like pytest, unit test, and nose.
- Integration with virtual environments for managing dependencies and isolated Python environments.

In the implementation of the hybrid cryptography system based on the Vigenere Cipher and Polybius Cipher, we can use PyCharm as an IDE to write, test, and debug our Python code. PyCharm can provide us with a comfortable and feature-rich environment for developing

Python applications, and its integrated debugging tools can help us to identify and fix issues in our code more efficiently. Additionally, PyCharm's code completion and refactoring tools can help us to write high-quality and maintainable code.

Advantages of PyCharm:

- **Code Analysis:** PyCharm has powerful code analysis tools that can help identify issues and provide suggestions for improvements, including syntax errors, code style violations, and possible bugs.
- **Debugging:** PyCharm provides a powerful debugging environment that supports various debugging workflows, including debugging remote code, testing individual functions, and inspecting code during runtime.
- **Integration with version control:** PyCharm integrates with popular version control systems like Git, SVN, and Mercurial, making it easy to manage code changes and collaborate with others.
- **Code refactoring**: PyCharm provides code refactoring tools that make it easy to improve code quality and maintainability. For example, it can rename variables, extract methods, and optimize imports.
- **Productivity features:** PyCharm provides several productivity features like code completion, code templates, and live templates that can help developers write code more efficiently.
- **Cross-platform support**: PyCharm is available for Windows, macOS, and Linux, making it a cross-platform IDE that can be used on different operating systems.

PyCharm is a powerful and feature-rich IDE that can help developers write high-quality, maintainable code more efficiently. Its integration with version control systems, debugging tools, and code analysis features make it an essential tool for Python developers.

4.2 Sample code:

Vigenere.py:

```
import string

def vigenere_encrypt(plaintext, key):
    # Convert the plaintext and key to lowercase
    plaintext = plaintext.lower()
    key = key.lower()

# Create a list of characters that should not be encrypted
    preserve_chars = string.digits + string.punctuation + " "
    # Encrypt the plaintext message
    ciphertext = ""
    key_index = 0
    for char in plaintext:
    if char in preserve_chars:
        ciphertext += char
    else:
        # Shift the character by the corresponding amount in the key
        shift = ord(key[key_index]) - ord('a')
        char_shifted = chr((ord(char) - ord('a') + shift) % 26 +
ord('a'))
        ciphertext += char_shifted
        key_index = (key_index + 1) % len(key)
    return ciphertext

def vigenere_decrypt(ciphertext, key):
    # Convert the ciphertext and key to lowercase
    ciphertext = ciphertext.lower()
    key = key.lower()
    # Create a list of characters that should not be decrypted
    preserve_chars = string.digits + string.punctuation + " "
    # Decrypt the ciphertext message
    plaintext = ""
    key_index = 0
    for char in ciphertext:
    if char in preserve_chars:
        plaintext += char
    else:
    # Shift the character back by the corresponding amount in the key
        shift = ord(key[key_index]) - ord('a')
        char_shifted = chr((ord(char) - ord('a') - shift) % 26 +
```

```
ord('a'))
                    plaintext += char_shifted
                    key_index = (key_index + 1) % len(key)
        return plaintext
```

Polybius.py:

```
    import string

def generate_polybius_key(key):
        alphabet = string.ascii_lowercase
        # Remove any duplicate characters from the keyword
        key = "".join(sorted(set(key.lower())))
        # Remove the keyword letters from the alphabet
        alphabet = alphabet.translate(str.maketrans("", "", key))
        # Construct the Polybius square key
        polybius_key = key + alphabet
        return polybius_key

def polybius_encrypt(plaintext, key):
        # Generate the Polybius square key
        polybius_key = generate_polybius_key(key)

# Create a dictionary mapping characters to their Polybius square coordinates
        polybius_dict = {}
        for i in range(len(polybius_key)):
        row = i // 5 + 1
        col = i % 5 + 1
        polybius_dict[polybius_key[i]] = (row, col)
        # Encrypt the plaintext message
        ciphertext = ""
        for char in plaintext.lower():
        if char == " ":
                ciphertext += " "
        elif char in polybius_key:
                row, col = polybius_dict[char]
                ciphertext += str(row) + str(col)
        return ciphertext

def polybius_decrypt(ciphertext, key):
        # Generate the Polybius square key
        polybius_key = generate_polybius_key(key)
```

```
# Create a dictionary mapping Polybius square coordinates to characters
                polybius_dict = {}
                for i in range(len(polybius_key)):
                row = i // 5 + 1
                col = i % 5 + 1
                polybius_dict[(row, col)] = polybius_key[i]
                # Decrypt the ciphertext message
                plaintext = ""
                i = 0
                while i < len(ciphertext):
                if ciphertext[i] == " ":
                        plaintext += " "
                        i += 1
                else:
                        row = int(ciphertext[i])
                        col = int(ciphertext[i+1])
                        char = polybius_dict[(row, col)]
                        plaintext += char
                        i += 2
                return plaintext
```

hybridcipher.py:

```
from vigenere3 import *
from polybius3 import *

def hybrid_cipher(plaintext,key1,key2):

fciphertext=polybius_encrypt(vigenere_encrypt(plaintext,key1),
key2)
        return fciphertext
```

hybriddecipher.py:

```
from vigenere3 import *
from polybius3 import *

def hybrid_decipher(ciphertext,key1,key2):

plaintext=vigenere_decrypt(polybius_decrypt(ciphertext,key2),
key1)
        return plaintext
```

application.py:

```
from tkinter import *
from tkinter import filedialog
from functools import partial
from new import hybrid_cipher
from new1 import hybrid_decipher

global filename
button_height = 2
button_width = 25

def browseFiles():

browseFiles.filename = filedialog.askopenfilename(initialdir="/",
title="Select a Text File")
                        label_file_explorer.configure(text="File Opened: " +
browseFiles.filename)

                        pass_label.pack()
                        temp_label.pack()
                        button_save.pack()
                        button_encrypt.pack()
                        button_decrypt.pack()

def save():
                with open("keys.txt","w") as keys:
        keys.write('Vigenere Key: '+str(key_entry1.get())+'\n')
        keys.write('Polybius Key: '+str(key_entry2.get()))
    def encrypt_file():
                vig_key=str(key_entry1.get())
                pol_key=str(key_entry2.get())
                with open('key1.txt','w') as key1:key1.write(vig_key)
                with open('key2.txt','w') as key2:key2.write(pol_key)

                with open(browseFiles.filename, 'r') as file:  original =
    file.read()
                encrypted = hybrid_cipher(original,vig_key,pol_key)

                with open(browseFiles.filename, 'w') as encrypted_file:
    encrypted_file.write(encrypted)

                status_label.configure(text="Your file is Encrypted")
```

```
                    status_label.pack()

        def decrypt_file():
                    vig_key = str(key_entry1.get())
                    pol_key = str(key_entry2.get())
                    with open('key1.txt','r') as key1:k1=key1.read()
                    with open('key2.txt','r') as key2:k2=key2.read()

                    if(vig_key==k1 and pol_key==k2):
                    with open(browseFiles.filename, 'r') as enc_file:
        encrypted = enc_file.read()
                    decrypted = hybrid_decipher(encrypted,vig_key,pol_key)

                    with open(browseFiles.filename, 'w') as dec_file:
        dec_file.write(decrypted)

                    status_label.configure(text="Your file is Decrypted")
                    status_label.pack()
                    else:
                    status_label.configure(text="Invalid keys")
                    status_label.pack()


        window = Tk()

window.title('Hybrid Cryptography System Using Vigenere Cipher and Polybius
Cipher')
        window.geometry("940x740")
        window.config(background="skyblue")

main_title = Label(window, text="HYBRID CRYPTOGRAPHY SYSTEM", width=100, h
eight=2, fg="white", bg="skyblue",font =("",30))

        key_entry1=StringVar()
        key_entry2=StringVar()

        submit_para_en = partial(encrypt_file)
        submit_para_de = partial(decrypt_file)

credit = Label(window,text = "", bg="skyblue",height=2,  fg = "white", font =("",15))

        label_file_explorer = Label(window, text="File Name : ", width=100, height=2,
        fg="white", bg="skyblue",font =("",20))
```

```
pass_label = Label(window, text="Keys for encryption/decryption : ", width=100, height=
2, fg="black", bg="skyblue",font =("",20))

        temp_label = Label(window, text="", height=3, bg="skyblue")

button_explore = Button(window, text="Browse File", command=browseFiles,
width=button_width, height=button_height, font =("",15))

        key_label1=Label(window,text="Vigenere Key : ",fg="black").place(x=200,y=350
)
        entry1 = Entry(window,textvariable=key_entry1,show="*").place(x=300,y=350)

        key_label2=Label(window,text="Polybius Key : ",fg="black").place(x=500,y=350)
        entry2=Entry(window,textvariable=key_entry2,show='*').place(x=600,y=350)



        button_save = Button(window, text="Save Keys", command=save, width=
        button_width, height=button_height, font =("",15))

button_encrypt = Button(window, text="Encrypt", command=submit_para_en, width=butt
on_width, height=button_height, font =("",15))

button_decrypt = Button(window, text="Decrypt", command=submit_para_de, width=butt
on_width, height=button_height, font =("",15))

status_label = Label(window, text="", width=100, height=4, fg="black",
bg="skyblue",font =("",17))



        credit.pack()
        main_title.pack()
        label_file_explorer.pack()
        button_explore.pack()
        window.mainloop()
```

CHAPTER – 5

SYSTEM TESTING

Testing forms an integral part of any software development project. Testing helps in ensuring that the final product is by and large, free of defects and it meets the desired requirements. Proper testing in the development phase helps in identifying the critical errors in the design and implementation of various functionalities thereby ensuring product reliability. Even though it is a bit time-consuming and a costly process at first, it helps in the long run of software development.

5.1 Need of Testing

Testing is an essential part of the development process for any software system, including the hybrid cryptography system based on the Vigenère Cipher and Polybius Cipher. Testing is necessary to ensure that the system is working as intended and to identify any potential security vulnerabilities that may exist. Without testing, it is impossible to ensure that the system is secure and free from errors or bugs. Testing can help to identify issues early in the development process, which can ultimately save time and resources in the long run. Testing the hybrid cryptography system can involve several types of testing, including unit testing, integration testing, functional testing, security testing, and performance testing. Each type of testing focuses on different aspects of the system and can help to identify potential issues or vulnerabilities.

Testing is crucial to the success of the hybrid cryptography system based on the Vigenère Cipher and Polybius Cipher. By testing the system thoroughly, we can ensure that it is secure, reliable, and works as intended.

5.2 Types of Testing

There are several types of testing that could be performed on the hybrid cryptography system, including:

- **Unit Testing**: This type of testing focuses on testing individual components of the system, such as the Vigenere Cipher and Polybius Cipher algorithms, to ensure that they are working correctly.
- **Integration Testing**: This type of testing focuses on testing how the different components of the system work together, such as how the Vigenere Cipher and Polybius Cipher are combined, to ensure that they are working correctly.
- **Functional Testing**: This type of testing focuses on testing the system's functionality, such as encrypting and decrypting messages using different keys, to ensure that the system is working as intended.
- **Security Testing**: This type of testing focuses on identifying potential security vulnerabilities in the system, such as using brute force attacks to try to crack the encryption, to ensure that the system is secure.
- **Performance Testing**: This type of testing focuses on testing how the system performs under different conditions, such as encrypting and decrypting large amounts of data, to ensure that the system is performing optimally.

By performing these different types of testing, we can ensure that the hybrid cryptography system based on the Vigenere Cipher and Polybius Cipher is working correctly and is secure from potential attacks.

5.3 Test Cases

Table.5.1 Passed TestCases

Test No	Test Cases	Expected Output	Actual Output	Pass/ Fail
1	Importing the pre-defined and user-defined modules	Imported	Imported without errors	Pass

2	Executing application page to view main window of application	Executed	Executed successfully and main window is opened	Pass
3	Loading text file which you want to encrypt	Loaded	Loaded successfully and file path is displayed	Pass
4	Taking Keys for encryption as input from users	Input taken	Input is taken successfully	Pass
5	Saving keys into a text file	Keys saved in to a file	Keys are saved into a file named keys.txt for verification during decryption.	Pass
6	Encrypting the contents of loaded file	Data encrypted successfully	Data is encrypted successfully and encrypted content is written back to the loaded file	Pass

			for transmission	
7	Decrypting the contents of a loaded file	Decrypted successfully	The file contents are decrypted successfully by verifying keys that are saved in other text files, and decrypted contents are written back to file.	Pass
8	Displaying status description	Successfully displayed based on actions performed by users.	Status is successfully displayed based on activity i.e.,after encryption it shows 'your file is successfully encrypted' and so on.	Pass

Table.5.2 Failed TestCases

Test No	Test Cases	Expected Output	Actual Output	Pass/ Fail
1	Selecting file with contents other than alphabets and performing encryption.	Successfully selected and encrypted	Encryption ignores other characters which results in loss of information.	Failed
2	Selecting an unencrypted file during decryption.	File will be decrypted	An error occurs due to invalid type.	Failed
3	Performing decryption of a file after encrypting another file.	File will be decrypted and original plaintext is retrieved.	Plain text will be variated due to change in value of keys during second encryption.	Failed

5.4 Results / Screenshots

The figure 5.1 depicts the main window of the application which is displayed through executing the application file.

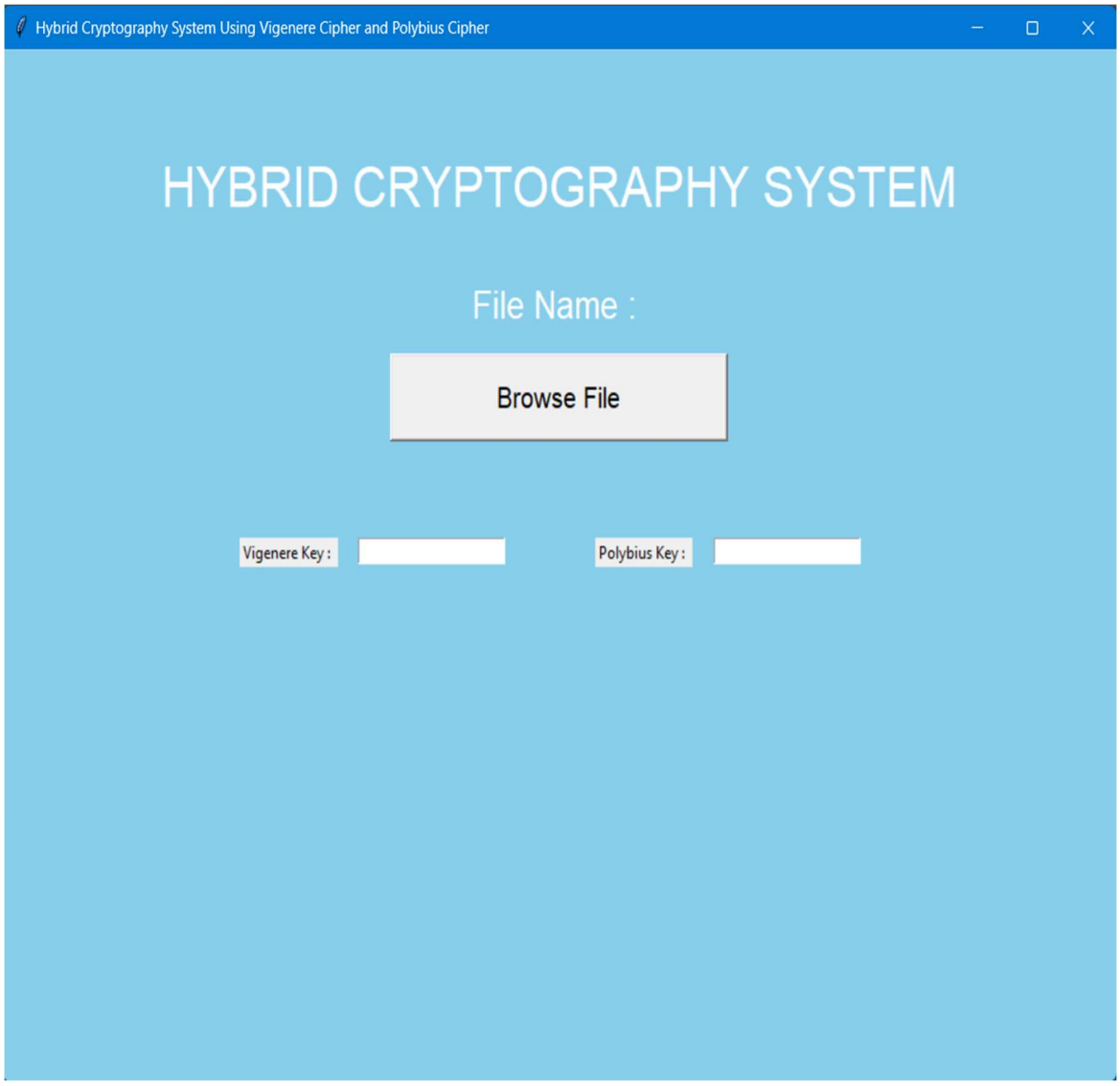


Figure.5.1 Main window of application

The figure 5.2 depicts the activity of file selection. In this activity user is allowed to select a text file for encryption and the path of selected file is displayed as shown in figure.

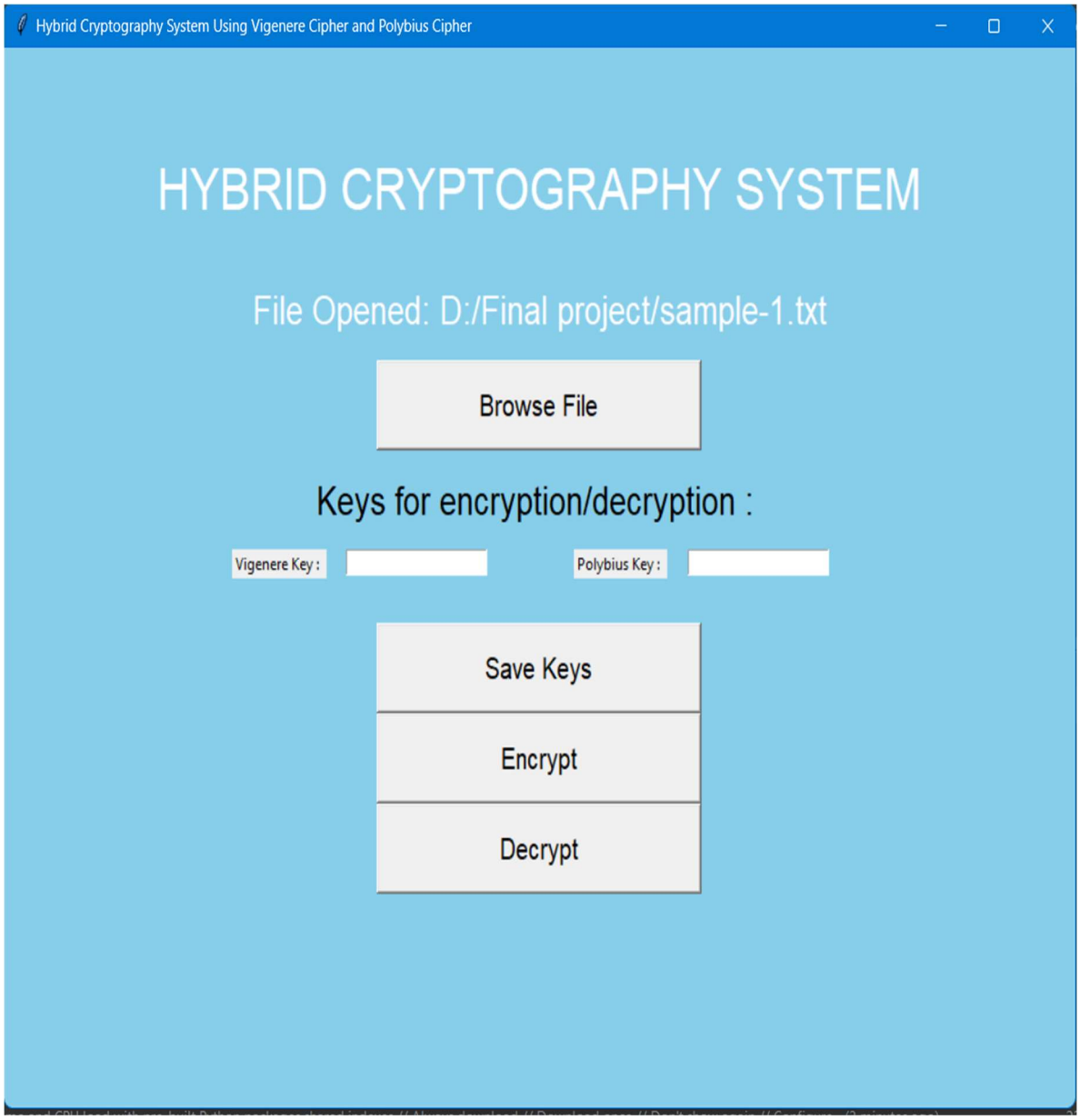


Figure.5.2 File selection activity

The figure 5.3 depicts the activity of taking keys for encryption as input from user. The keys entered would be hidden as shown in figure.

Figure.5.3 Input keys for encryption/decryption

The figure 5.4 depicts the encryption activity in which the contents of selected file is encrypted using hybrid algorithm proposed. The status of activity is displayed using status label as shown in figure.

Figure.5.4 File encryption

The figure 5.5 shows the original contents of the file before encryption which includes some sample text to test the system for effectiveness in its encryption process.

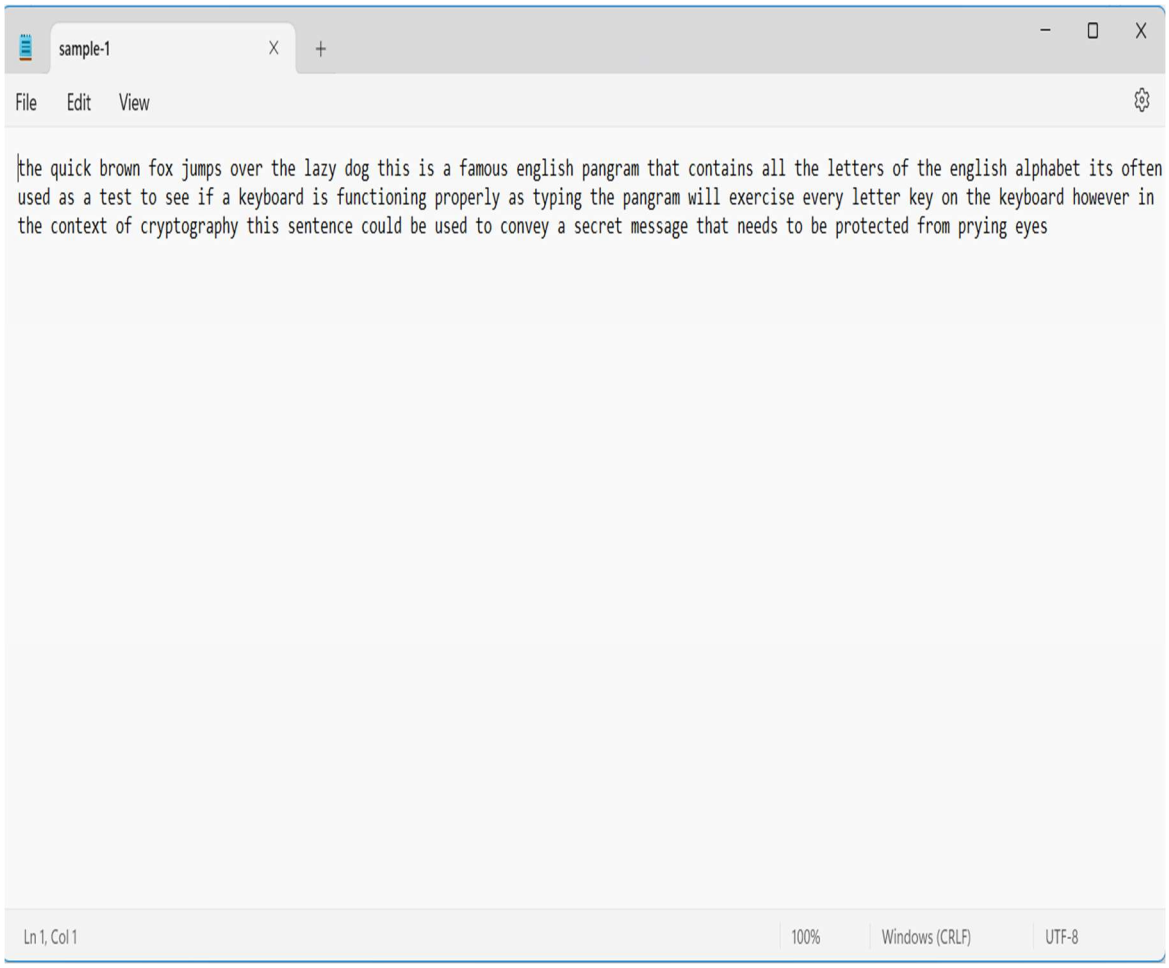


Figure.5.5 Original contents of file

The figure 5.6 shows the encrypted contents of the file after encryption using hybrid approach proposed the project.

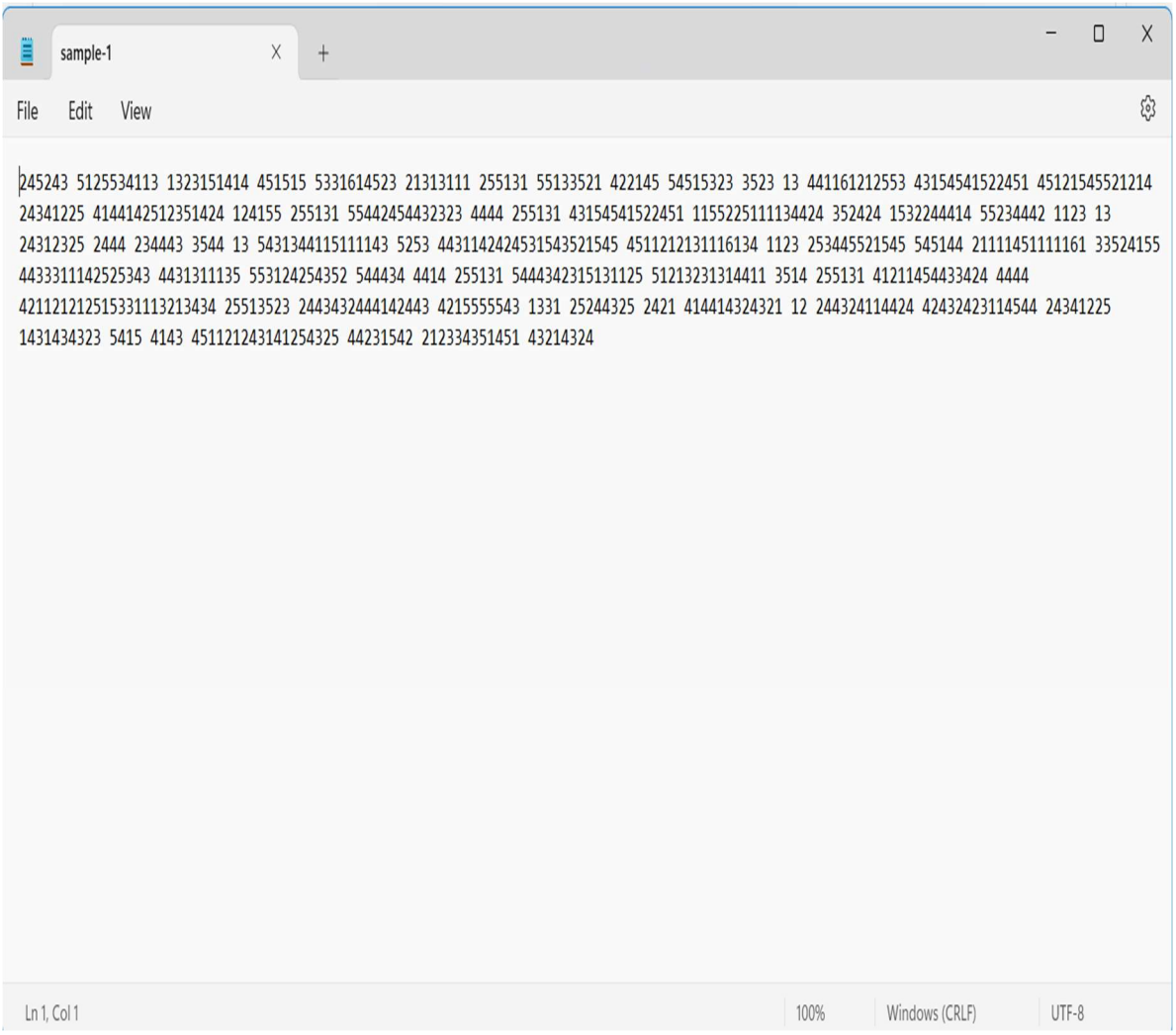


Figure.5.6 Encrypted contents of file

The figure 5.7 depicts the decryption activity in which an encypted file is selected and its contents are decrypted using the same keys saved during encryption.

Figure.5.7 File decryption

The figure 5.8 depicts the decryption activity in which when incorrect keys are provied for decryption the status will be displayed as the invalid keys as shown in figure.

Figure.5.8 File decryption with invalid keys

CHAPTER –6

CONCLUSION

The hybrid cryptography system based on the Vigenere Cipher and Enhanced Polybius Cipher provides an effective way to enhance the security of encrypted data. By combining the strengths of both encryption algorithms, we can create a more complex and difficult-to-crack encryption scheme. The Vigenere Cipher provides polyalphabetic substitution, while the Polybius Cipher offers letter-to-number substitution, which increases the complexity of the ciphertext. However, it is important to note that this system is not unbreakable, and it may still be vulnerable to certain attacks. To ensure the security of the encrypted data, it is essential to choose a strong, random, and secret key. Future enhancements could further increase the system's security, such as improved key management, adding additional encryption algorithms, developing a more secure Polybius Cipher, or integrating modern cryptography techniques.

Finally, The hybrid cryptography system based on the Vigenere Cipher and Enhanced Polybius Cipher provides a good level of security and is relatively easy to implement.

CHAPTER – 7

FUTURE ENHANCEMENTS

The Vigenere cipher and the Keyed Polybius cipher are both classical encryption techniques, However, they are susceptible to various attacks, such as frequency analysis and known plaintext attacks. To enhance the security of a hybrid cryptography system based on these ciphers, some possible future enhancements are:

- **Key management:** One of the critical factors that affect the security of a hybrid cryptography system is key management. Therefore, an essential enhancement is to develop a robust key management system that ensures the security and integrity of the keys used in the encryption and decryption processes.
- **Add a third layer of encryption:** To further enhance the security of the system, a third layer of encryption can be added. The third layer could be a modern cryptographic algorithm such as AES or RSA.
- **Use a larger key space:** The Vigenere cipher and the Keyed Polybius cipher have a limited key space, making them vulnerable to brute-force attacks. Therefore, increasing the key space by using longer keys can make the system more secure.
- **Randomizing the keys:** To make it harder for an attacker to guess the key, the keys used in the encryption process can be randomized. This can be achieved by using a random number generator to generate the key or by using a key derivation function.
- **Adding authentication:** To ensure the integrity of the message, an authentication mechanism such as a digital signature or a message authentication code (MAC) can be added.
- **Using a different cipher:** While the Vigenere and Keyed Polybius ciphers are good at hiding the frequency distribution of the plaintext, they are vulnerable to other attacks.

BIBLIOGRAPHY

[1] Ali, F. M. S., & Sarhan, F. H. (2014). Enhancing security of vigenere cipher by stream cipher. *International Journal of Computer Applications*, *100*(1), 1-4.

[2] Mendrofa, E. H., Purba, E. Y., Siahaan, B. Y., & Sembiring, R. W. (2017). Collaborative encryption algorithm between vigenere cipher, rotation of matrix (ROM), and one time pad (OTP) algoritma. *Adv. Sci. Technol. Eng. Syst. J*, *2*(5), 13-21.

[3] Maity, M. (2014). A modified version of polybius cipher using magic square and western music notes. *International Journal For Technological Research In Engineering, ISSN*, *2347*(4718), 1.

[4] Bhardwaj, C. (2012). Modification of vigenere cipher by random numbers, punctuations & mathematical symbols. *Journal of Computer Engineering (IOSRJCE) ISSN*, (2278-0661).

[5] Arroyo, J. C. T., & Delima, A. J. P. (2020). A Modified Polybius Cipher with a New Element-in-Grid Sequencer. *International Journal*, *9*(3).

[6] Vatshayan, S., Haidri, R. A., & Verma, J. K. (2020, July). Design of hybrid cryptography system based on vigenère cipher and polybius cipher. In *2020 International Conference on Computational Performance Evaluation (ComPE)* (pp. 848-852). IEEE.

[7] Biswas, M. H., Ali, M. A., Rahman, M., & Sohel, M. M. K. (2019). A systematic study on classical cryptographic cypher in order to design a smallest cipher. *Int. J. Sci. Res. Publ*, *9*(12), 507-11.

[8] Coggins III, P. E., & Glatzer, T. (2020). An Algorithm for a Matrix-Based Enigma Encoder from a Variation of the Hill Cipher as an Application of 2× 2 Matrices. *Primus*, *30*(1), 1-18.

[9] Thakor, V. A., Razzaque, M. A., & Khandaker, M. R. (2021). Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities. *IEEE Access*, *9*, 28177-28193.

[10] Ahmad, S. A., & Garko, A. B. (2019, December). Hybrid cryptography algorithms in cloud computing: A review. In *2019 15th International Conference on Electronics, Computer and Computation (ICECCO)* (pp. 1-6). IEEE.