A Project Report

on

"CRYPTOTOOL"

Submitted for partial fulfillment of requirement for the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

By

MR. PRATHAMESH L. KHANDELWAL **MISS. PRANOTI P. POHOKAR**

MISS. KHUSHI R. KAWANPURE **MISS. PUJA R. JADHAV**

MR. TANAY A. NANGLIA

Under the guidance of

Prof. P. H. DHOLE

Department of Computer Science and Engineering

(Accredited by NBA)

Sipna College of Engineering & Technology, Amravati.

(An ISO 9001 : 2015 Certified)

NAAC Accredited with Grade"A+"

Sant Gadge Baba Amravati University, Amravati

2023-2024

Project Report
on
"CRYPTOTOOL"

SUBMITTED TO SANT GADGE BABA AMRAVATI UNIVERSITY
IN THE PARTIAL FULFILLMENT OF THE DEGREE OF

BACHELORS OF ENGINEERING
In
COMPUTER SCIENCE AND ENGINEERING

By

MR.PRATHAMESH L. KHANDELWAL MISS. PRANOTI P. POHOKAR

MISS.KHUSHI R. KAWANPURE MISS.PUJA R. JADHAV

MR.TANAY A. NANGLIA

Under the guidance of

Prof. P.H.DHOLE

Department of Computer Science and Engineering
(Accredited by NBA)
Sipna College of Engineering & Technology, Amravati.
(An ISO 9001 : 2015 Certified)

NAAC Accredited with Grade"A+"
Sant Gadge Baba Amravati University, Amravati

2023-2024

Sipna College of Engineering & Technology, Amravati

Computer Science & Engineering

CERTIFICATE

This Is to certify that **Mr. Prathamesh L. Khandelwal, Miss. Khushi R. Kawanpure, Miss. Pranoti P. Pohokar, Miss. Puja R. Jadhav, Mr. Tanay A. Nanglia** has satisfactorily completed the project work towards the **Bachelor of Engineering** Degree of Sant Gadge Baba Amravati University, Amravati in **Computer Science & Engineering** discipline on the topic entitled **"CRYPTOTOOL",** during the academic year 2023-2024 under my supervision and guidance.

Date:

Prof. P. H. Dhole
Guide

Dr. V. K. Shandilya
Head of Department

Dr. S.M. Kherde
Principal

PROJECT APPROVAL SHEET

Project Entitled

"CRYPTOTOOL"

By

Mr. Prathamesh L. Khandelwal **Miss. Pranoti P. Pohokar**

Miss. Khushi R. Kawanpure **Miss. Puja R. Jadhav**

Mr. Tanay A. Nanglia

Is approved for the degree of

Bachelor of Engineering

in

Computer Science & Engineering

of

Sant Gadge Baba Amravati University, Amravati

Internal Examiner	**External Examiner**
Name:	**Name:**
Date:	**Date:**

ACKNOWLEDGEMENT

A moment of pause, to express a deep gratitude to several individuals, without whom this project could not have been completed. We feel immense pleasure to express deep sense of gratitude and indebtedness to our guide **Prof**. **P. H. Dhole** and **Dr. V. K. Shandilya,** Head, Department of Computer Science & Engineering for constant encouragement and noble guidance.

We express our sincere thanks to the other staff members of the department for their kind co-operation.

We express our sincere thanks to **Dr. S. M. Kherde**, Principal, Sipna College of Engineering & Technology for his valuable guidance.

We also express our sincere thanks to the library staff members of the college for their Valuable support.

Last but not the least we are thankful to our friends and our parents whose best wishes are always with us.

Name of Student

Mr. Prathamesh L. Khandelwal

Miss. Khushi R. Kawanpure

Miss. Pranoti P. Pohokar

Miss. Puja R. Jadhav

Mr. Tanay A. Nanglia

ABSTRACT

The study presents a comprehensive exploration of cryptographic techniques, tracing their evolution from ancient historical periods to contemporary technological advancements. Motivated by the need for a versatile platform encompassing both symmetric and asymmetric encryption algorithms, our research introduces a standalone application designed to serve as a unified hub for cryptographic operations. This platform integrates major encryption modes, including Caesar Cipher, AES Algorithm, Modified Caesar Cipher, and RSA, alongside two proprietary algorithms, namely the Arithmetic No Key Algorithm and Bit Manipulation Algorithm. Leveraging modular development principles and the Java Cryptographic Extension Framework, the application exhibits enhanced flexibility and usability, catering to users ranging from cryptography enthusiasts to professionals. By using Dynamic Loading, the application seamlessly ties together diverse cryptographic tools, promoting worldwide awareness of cryptography and its applications. Through a holistic approach, our research endeavors to bridge classic and modern cryptographic methods, thereby fostering a deeper understanding and appreciation of the intricacies of encryption and decryption techniques in today's digital landscape.

Table of Content

List of Figures

List of Tables

List of Screenshots

INTRODUCTION

As Technique of Encryption and decryption is not very hard to understand because we already using different encryption methods to transfer our message since from Ancienthistorical Period to today's technological era.

Here we are providing platform for both the symmetric as well as Asymmetric key cryptographic algorithms specifically keeping its eye on all-in-one encryption standards via incorporating major four encryption modes Caesar Cipher, AES Algorithm, Modified Caesar cipher, RSA. Six Algorithms we have inserted here out of which two are indigenously made named as Arithmetic No Key Algorithm and Bit Manipulation Algorithm.

Motivation for building this project comes from software Engineering Project where we come up with lots of ideas and finally, we finalize this topic. As there are lots of cryptographictools and website are available in the world of Cryptography but we are trying make one such mechanism where we can tie both the symmetric and asymmetric key algorithms with each other by using Dynamic Loading.

It is a Stand-Alone Application which can perform multiple Encryption via using Java (JDK 21 and above). This Stand-Alone Application system that aims enhancing worldwide awareness of cryptography and its uses; JAVA Cryptographic extension Framework is thebase of these proposed system mechanism. In this application we have used modular development hence flexibility of application is increased. Combined efforts have led to the development of a Stand- alone application that shows several classic and modern methods of cryptography, amongst which: Caesar ciphers and various methods of cryptanalysis.

CRYPTOGRAPHY

Cryptography is a dynamic and a mandatory component of digital business. Organizations need visibility into their cryptographic instances as well as guidance from not only standards groups such as NIST and ISO (International Organization for Standardization), but also the web browsers who control the user interfaces that connect businesses with consumers via secure online communications.

Crypto agility is the key to keeping pace with the latest cryptographic compliance requirements, standards, and recommendations that sustain and secure digital business.

Cryptography is the practice and study of techniques for securing communication and data from adversaries. It's a fundamental aspect of modern information security, used in various contexts such as online banking, secure communication, digital signatures, and more.

At its core, cryptography involves encoding plaintext (readable information) into ciphertext (encoded information) using algorithms and keys. The cipher text can then be safely transmitted or stored, and only authorized parties possessing the correct key can decode it back into plaintext.

Some Basic Terminologies in Cryptography :

- **Plaintext:** This is the original, readable message or data before any encryption is applied.
- **Ciphertext:** After encryption, the plaintext is transformed into ciphertext, which is an unreadable form of the message.
- **Encryption:** The process of converting plaintext into ciphertext using an algorithm and a key.
- **Decryption:** The reverse process of encryption, transforming ciphertext back into plaintext using the appropriate decryption key.
- **Key:** A parameter used in conjunction with an algorithm to encrypt and decrypt data. In symmetric cryptography, the same key is used for both encryption and decryption, while in asymmetric cryptography, there are separate public and private keys.
- **Hash Function:** A one-way function that transforms input data into a fixed-size string of characters, known as a hash value or digest. It's commonly used for data integrity verification and password storage. Examples include SHA-256 (Secure Hash Algorithm 256-bit) and MD5(Message Digest Algorithm 5).
- **Digital Signature:** A cryptographic technique used to verify the authenticity and integrity of a message or document. It involves using a private key to create a unique signature for the message, which can be verified by anyone with access to the corresponding public key.

There are two main types of cryptography:

Symmetric Cryptography:

In symmetric cryptography, the same key is used for both encryption and decryption. This means that both the sender and the receiver must possess the same secret key. Algorithms like AES (Advanced Encryption Standard) and DES (Data Encryption Standard) are examples of symmetric cryptography.

Asymmetric Cryptography:

Also known as public-key cryptography, asymmetric cryptography uses a pair of keys: a public key and a private key. The public key is freely distributed and used for encryption, while the private key is kept secret and used for decryption. RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography) are common asymmetric encryption algorithms.

Cryptography serves several critical purposes:

- **Confidentiality:** It ensures that unauthorized parties cannot read sensitive information.
- **Integrity:** It verifies that the data has not been altered or tampered with during transmission.
- **Authentication:** It confirms the identity of communicating parties.
- **Non-repudiation:** It prevents individuals from denying their actions or transactions.

Overall, cryptography plays a vital role in maintaining the security and privacy of digital information in today's interconnected world.

Cryptography and steganography are both techniques used to secure information, but theydiffer in their approaches and objectives:

Cryptography:

- Involves the transformation of plaintext into ciphertext using algorithms and keys.
- Focuses on hiding the content of a message by encoding it in a way that makes it unreadable to unauthorized parties.
- The goal of cryptography is to ensure confidentiality, integrity, authenticity, and non-repudiation of data.
- Examples include encrypting a message with a secret key or using digital signatures to verifythe authenticity of a message.

Steganography:

- Involves hiding the existence of a message within another medium, such as an image, audio file or text.
- Focuses on concealing the presence of the message itself rather than its content.
- The goal of steganography is to ensure that the existence of the message remains undetected by unauthorized parties.
- Examples include embedding text within an image file by subtly modifying pixel values orconcealing a message within the least significant bits of an audio file.
- In summary, while cryptography focuses on securing the content of a message through encryption and authentication, steganography focuses on concealing the existence of themessage within another medium to achieve secrecy. They can be used in combination to enhance the security and privacy of communication and data.

Literature Review

- “THE HANDBOOK ON APPLIED CRYPTOGRAPHY” has given a brief account on cryptographic tool working and uses [1].

- Bhuman Vyas in "Security Challenges and Solutions in Java Application Development" addresses the security landscape specific to Java applications. Vyas explores vulnerabilities and offers solutions tailored to Java environments, likely encompassing topics such as secure coding practices, access control, and encryption techniques [2].

- Zhengyi Lu conducts an in-depth analysis of the Advanced Encryption Standard (AES) in "Analysis on AES encryption standard and safety". This analysis, presented at the Third International Symposium on Computer Engineering and Intelligent Communications, provides insights into the safety and efficacy of AES, a fundamental encryption standard used in Java application security [3].

- Yashar Salami, Vahid Khajehvand, and Esmaeil Zeinali's work titled "Cryptographic Algorithms: A Review of the Literature, Weaknesses and Open Challenges" offers a comprehensive review of cryptographic algorithms. Their paper, likely covering both symmetric and asymmetric cryptography, highlights weaknesses and challenges, providing valuable insights for implementing secure cryptographic solutions in Java applications [4].

- Arwa Zabian, Shakir Mrayyen, Abram Magdy Jonan, Tareq Al-Shaikh, and Mohammed Tthazi Al-Khaiyat propose a "Multi-layer encryption algorithm for data integrity in cloud computing". Although not directly focused on Java, this paper introduces encryption techniques relevant to securing data in distributed environments, which are often encountered in Java based cloud applications [5].

- M. A. Al-Shabi's "A Survey on Symmetric and Asymmetric Cryptography Algorithms in Information Security" provides a survey of cryptographic algorithms. This survey, likely encompassing various encryption techniques, offers insights into their applicability and effectiveness in information security, pertinent for securing Java applications [6].

- Prof. (Dr.) Amit Verma and Anjali Gakhar analyze "Tools and Techniques in Cryptography", shedding light on available cryptographic tools and methodologies. While not Java-specific, this analysis offers valuable insights into integrating cryptographic techniques into Java application security frameworks [7].

- Punita Meelu and Sitender Malik explore "AES: A Symmetric Key Cryptographic System". Although an older publication, this paper remains relevant as it provides a detailed examination of AES, a cornerstone encryption algorithm utilized in Java application security for securing sensitive data [8].

- Tanusree Saha's "Complement-Based Modified Approach to Secure Small Text Message Combining Triangulation Method" proposes an enhanced Symmetric key encryption algorithm which is the combination of substitution technique, 2's compliment technique and Triangulation Method[9].

- Deepak & Parveen work "Modern Encryption and Decryption Algorithm based on ASCII Value and Binary Operations" proposed an algorithm based on ASCII values and binary operations for both encryption and decryption to enhance the security[10].

- Shashi Gautam, Shubha Mishra, & Dr. Manish Shrivastava work titled "An Enhanced Encryption Technique using BCD and Bit Complementation". They proposed a cryptographic algorithm based on binary operations and with basic CPU computation. This algorithm uses BCD (binary coded decimal), 1's complement for data encryption and 2's complement for key encryption [11].

System Analysis

An in-depth analysis of the limitations of the current Cryptographic Tool reveals its vulnerability in the face of modern cyber threats. The inability to utilize asymmetric key cryptography is a significant drawback, and this project seeks to address this limitation by introducing a more comprehensive approach to cryptographic key management.

As cyber threats continue to evolve, the need for robust cryptographic tools becomes increasingly paramount. The original Cryptographic Tool, initially designed with a focus on simplicity through symmetric key cryptography, is ripe for enhancement. This project addresses this need by introducing advanced cryptographic methods to strengthen data protection.

Proposed work and Objectives:

In the dynamic landscape of cyber security, the demand for advanced cryptographic tools has never been more pressing. As data breaches and cyber threats escalate in sophistication, the Cryptographic Tool Enhancement project emerges as a timely response to fortify digital defenses. The project centers on the evolution of a Cryptographic Tool, initiallydesigned with simplicity in mind, primarily utilizing symmetric key cryptography.

Recognizing the need for a comprehensive and adaptable solution, the updated version aims to integrate both symmetric and asymmetric key cryptographic algorithms.

The enhanced Cryptographic Tool represents a pivotal advancement in securing sensitive information and communications. With an expanded repertoire of cryptographic techniques, the tool endeavors to address the limitations of its predecessor, introducing a versatile and robust cryptographic infrastructure. Beyond the rudimentary security measures of the original tool, the updated version is poised to provide a multifaceted defense against a spectrum of cyber threats.

Proposed System

This mechanism provides both the symmetric and asymmetric key Cryptographic Algorithms.

1. To place both these algorithms at one we have proposed here common Interface called Crypto Algorithm Interface and both the symmetric and asymmetric algorithms have to compulsorily implement.

2. The interface Crypto Algorithm consists of two methods - encrypt () and decrypt (). Both of these methods have a return type and parameter of String to ensure that they accept and return Strings.

3. The AlgorithmLoader is a module designed to dynamically load and manage cryptographic algorithms used in the application. Its purpose is to provide a flexible way to integrate different encryption and decryption algorithms without tightly coupling them to the main application logic.

4. By creating the AlgorithmLoader, we avoid hardcoding specific cryptographic algorithms directly into the application code. This dynamic loading process at runtime enables ourapplication to support multiple algorithms without altering the core code. This not only saves time and effort but also enhances the flexibility and scalability of the System.

5. Initialization: The AlgorithmLoader is initialized when the application starts.

6. Scanning for Algorithm Classes: It scans the specified package (in our implementation "cryptoAlgos") for classes that implement the CryptoAlgorithm interface.

7. Loading Algorithm Classes: It loads the algorithm classes dynamically using Java

reflection. The classes are identified, and Java reflection is used to load them into the application dynamically. This is done using the Class.forName() method, which takes the fully qualified class name as a string and returns the Class object for that class.

8. Storing Algorithm Classes: The loaded algorithm classes are stored in a list or data structure for easy access

9. Algorithm Selection: When needed, the application can request a specific algorithm by its name from the AlgorithmLoader.

10.Instantiation: After loading the names into the System, the user is asked to select a specific Algorithm module class, and then the Reflection is used again to instantiate the objects of the selected class. This is typically done using the Class.newInstance() method or by calling a constructor using Constructor.newInstance().

11.Returning Algorithm Object: It returns the instantiated algorithm object to the application for encryption or decryption

The Following is the logic to use Java Reflection in the proposed mechanism.

```
// Loading algorithm classes dynamically using reflection
Class<?>loadedClass = Class.forName("cryptoAlgos."+ className);

// Instantiating algorithm class
CryptoAlgorithm algorithmInstance =
(CryptoAlgorithm)
loadedClass.getDeclaredConstructor().newInstance();
```

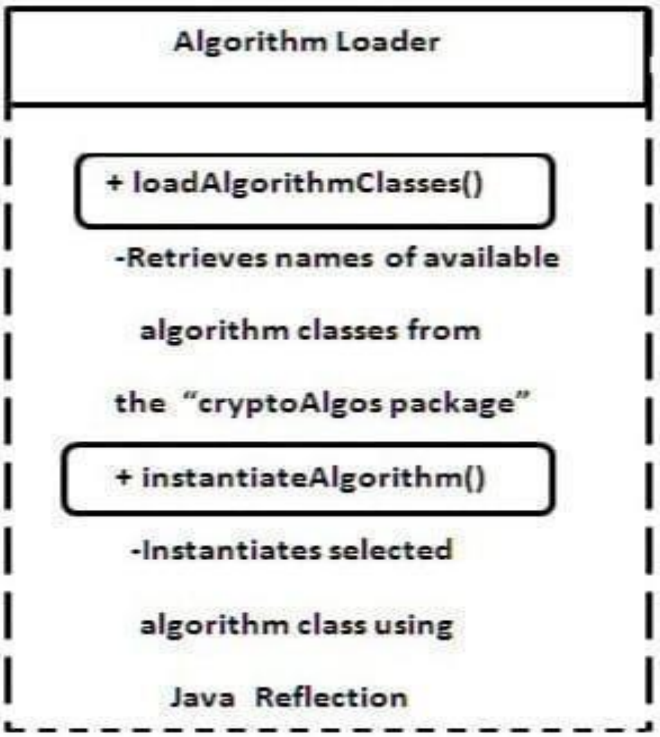


Figure: - 3.1(Working of Algorithm Loader)

Technology Used

- Primary Language Programming JAVA (jdk 21)
- Java swing (GUI Toolkit)
- Java Reflection
- Java Cryptographic Extension(JCE) Framework

Technical Toolkit

- VS Code
- Notepad++
- Java(jdk 21)

System design

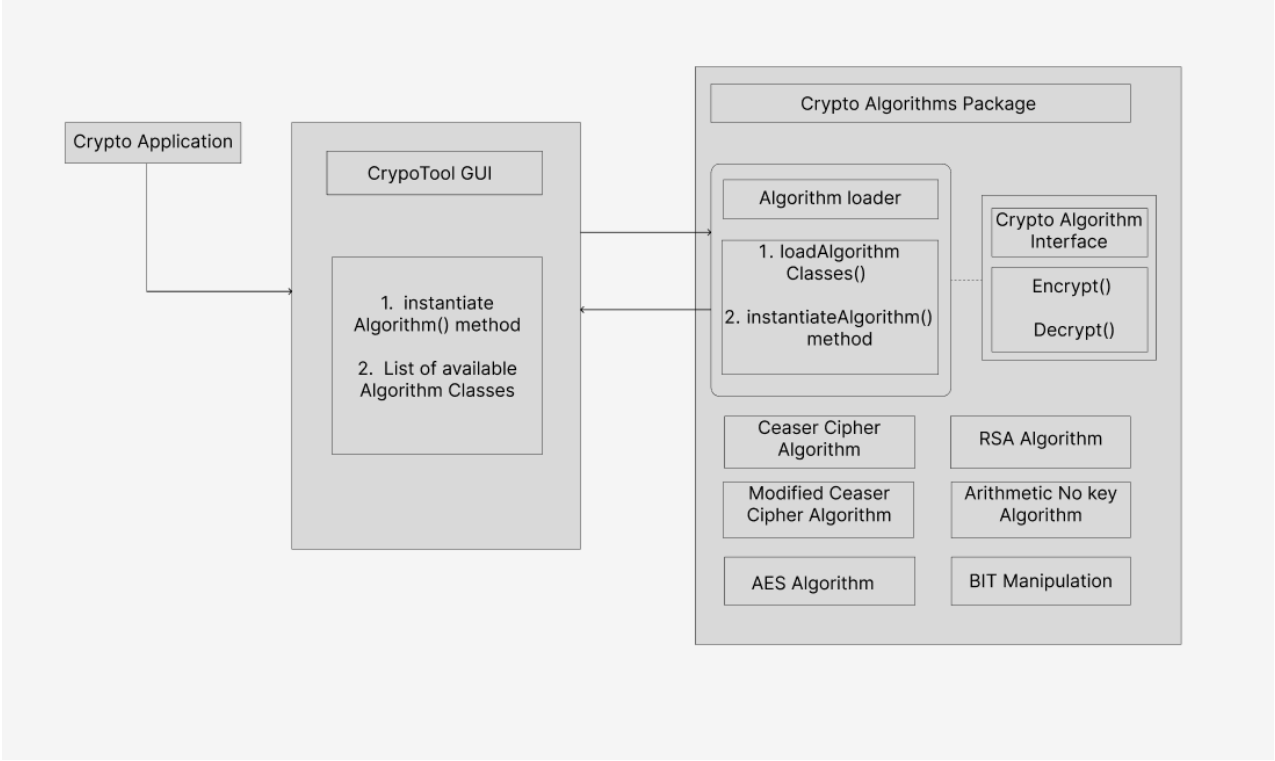


Figure:-4.1(Implementation Diagram)

Methodology

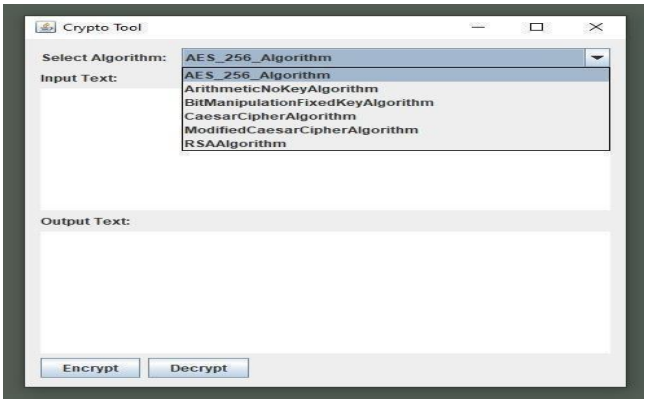
Proposed Application specially includes total 6 algorithms out of which 4are studied by usand 2of them are created by us proprietary algorithms.

Key terms

1) KEY =A key in cryptography is a piece of information, usually a string of numbers or letters that are stored in a file, which, when processed through a cryptographic algorithm, can encode or decode cryptographic data.

2) Encryption = Encryption is a way of scrambling data so that only authorized parties can understand the information.

3) Decryption = The process of encryption transforms information from its original format called plaintext into an unreadable format called ciphertext while it is being shared or transmitted.



Screenshot: - 01(Application Interface)

Sequence Diagram

Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows as parallel vertical lines (lifelines), different processes or objects that live simultaneously and as horizontal arrows, the messages exchanged between them in the order in which they occur

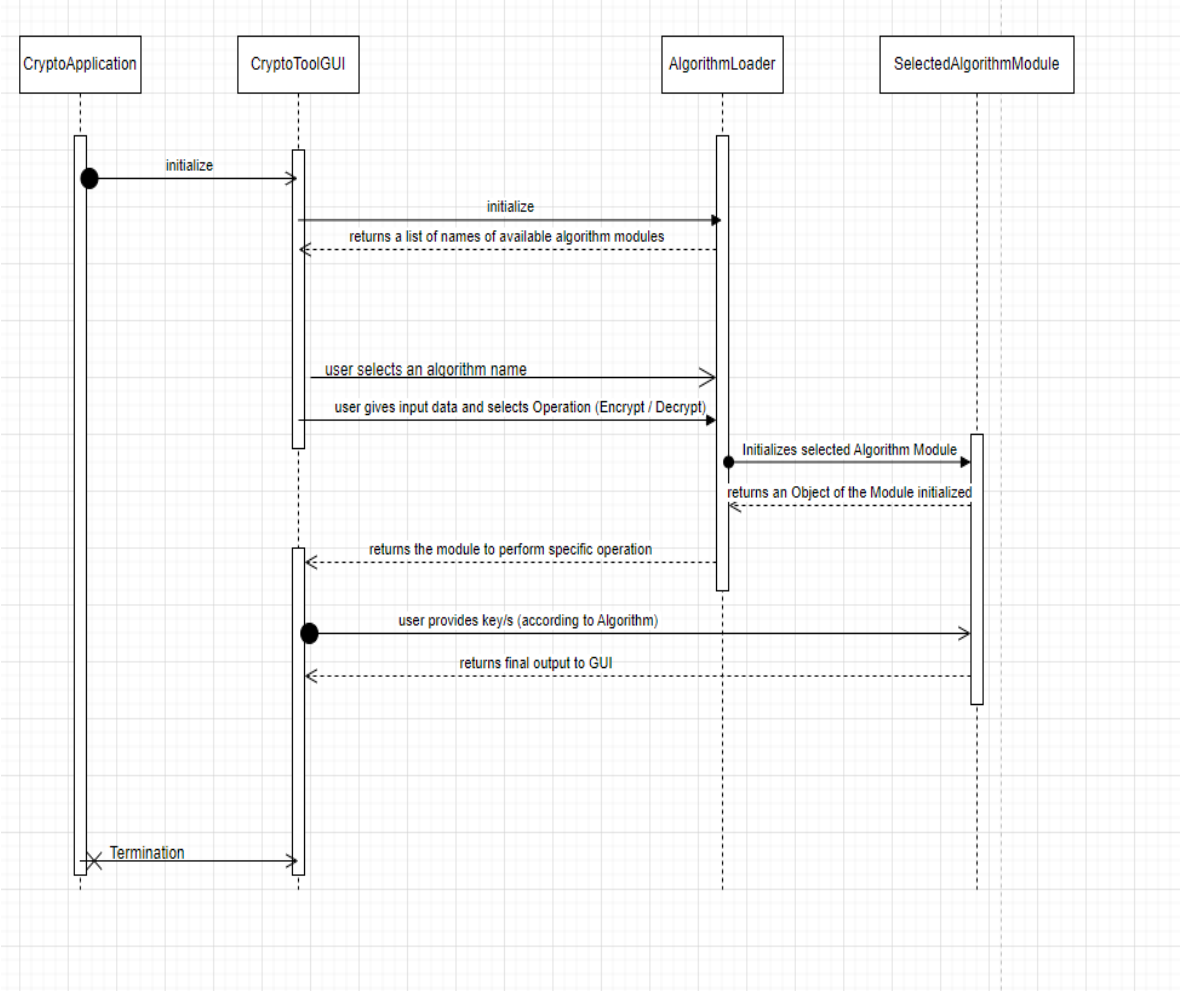


Figure:-4.2(Sequence diagram)

CryptoApplication initializes CryptoTool GUI, CryptoTool GUI simultaneously initialize AlgorithmLoader which scans cryptoAlgos package for available module classes. AlgorithmLoader provides a list of names of available algorithm modules. Now user has to select the desired algorithm name. After the user has input data that they want to encrypt or decrypt. After that Algorithmloader initializes the selected algorithm module. This object is passing to the CryptoTool GUI. The object is responsible for performing encryption and decryption operation. If the algorithm requires a key then the user has to input it via CryptoTool GUI which then transfers it to the selected algorithm module object. The Object directly transfers result to CryptoTool GUI. The result is displayed on CryptoTool GUI.

Data Flow Diagram

A data flow diagram is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated

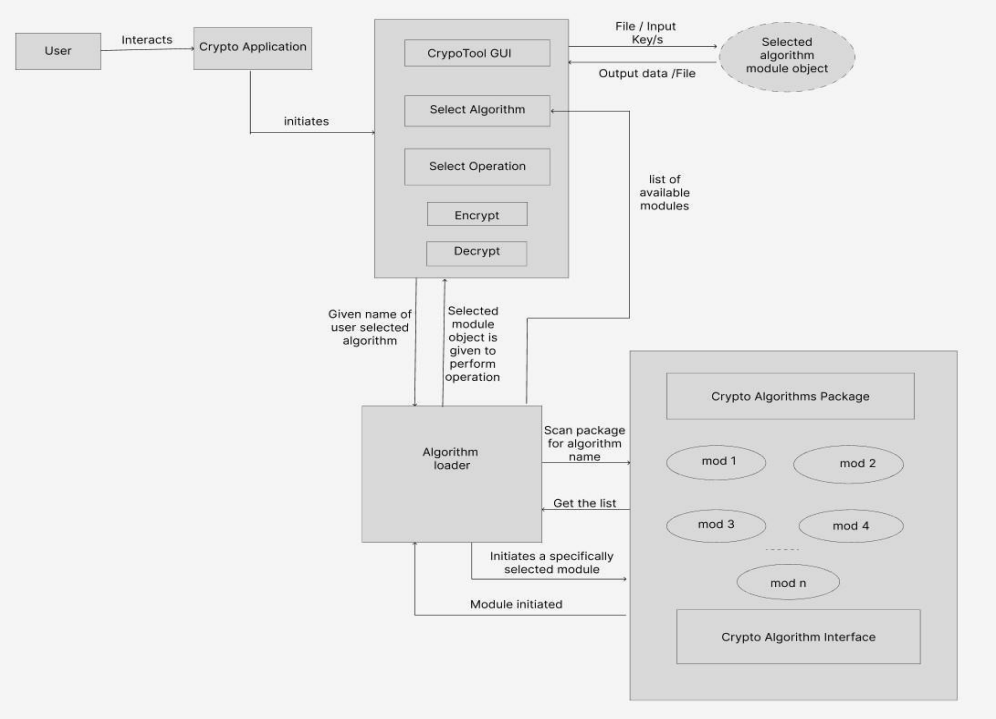


Figure:-4.3(Data Flow Diagram)

A data flow diagram (DFD) is a visual representation of the flow of data through a system or process. In the case of a CryptoTool, a DFD can be used to illustrate how data moves through the system and process occurs.

1. The user starts the Interaction with the interface and initiates the GUI

2. The next step is a selection of an algorithm from available modules

3. The algorithm name selected by the user is then sent to the algorithm loader, it initializes the object of the particular module selected and the output of this process is sent back to the GUI layer

4. A temporarily selected algorithm module object then carries out the encryption or decryption operations for the user and provides an output file.

Algorithms Used

1) Caeser Cipher

The Caesar cipher, also known as the shift cipher, is one of the simplest and earliest known encryption techniques. It involves shifting each letter in the plaintext by a fixed number of positions down or up the alphabet. It is a substitution Cipher which replaced each letter of given plaintext with 3 positions down the alphabet.

Here's a simple example using a Caesar cipher with a shift of 3:

Plaintext: HELLO

- H is shifted 3 positions to the right, becoming K
- E is shifted 3 positions to the right, becoming H
- L is shifted 3 positions to the right, becoming O
- L is shifted 3 positions to the right, becoming O
- Is shifted 3 positions to the right, becoming R So, the ciphertext would be: KHOOR

To decrypt, you would reverse the process, shifting each letter in the ciphertext 3 positions to the left, resulting in the original plaintext: HELLO.

The Caesar cipher is straightforward and easy to understand, but it's not very secure because there are only 25 possible keys (shifts), making it susceptible to brute force attacks. However, it serves as a fundamental example of encryption and is often used in introductory cryptography courses.

Encryption Logic

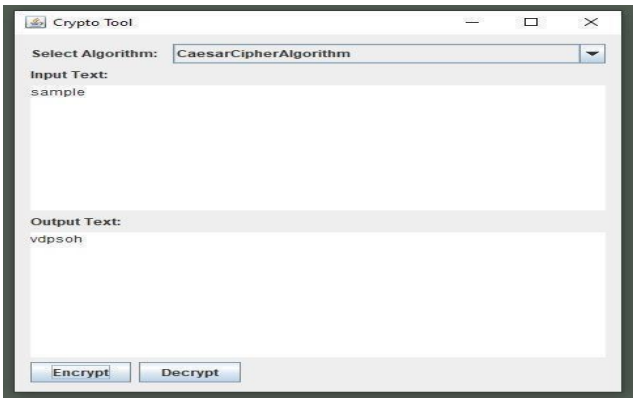
a. Choose a shift value (e.g., 3).

b. Iterate through each character in the plaintext.

c. Shift each letter in the plaintext forward in the alphabet by the chosen shift value.

d. Handle wraparound if the shift extends beyond the bounds of the alphabet.

e. Output the encrypted text.



Screenshot: - 02(Caeser Cipher Encryption)

Decryption Logic

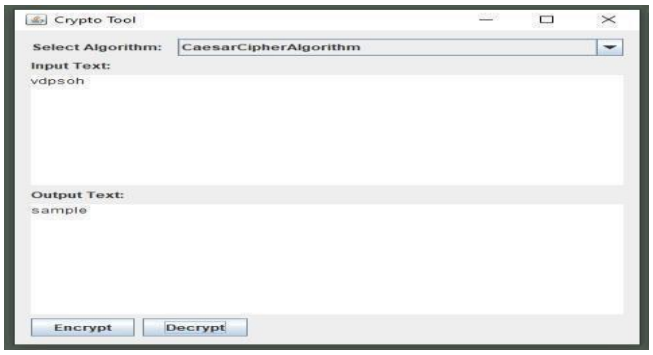
a. Given the ciphertext and the known shift value (e.g., 3).

b. Iterate through each character in the ciphertext.

c. Shift each letter in the ciphertext backward in the alphabet by the chosen shift value.

d. Handle wraparound if the shift extends beyond the bounds of the alphabet.

e. Output the decrypted text.



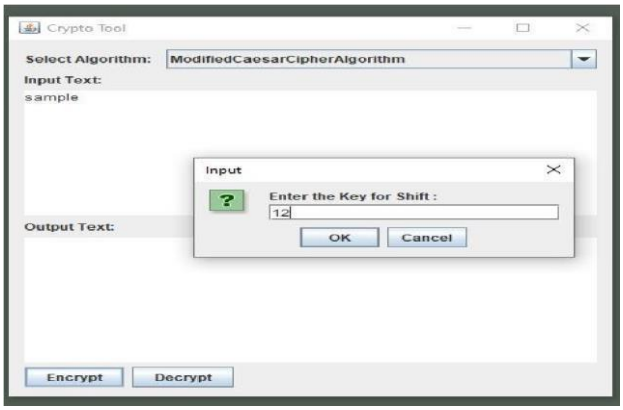
Screenshot:-03(Caeser Cipher Decryption)

2) Modified Caeser Cipher

The Modified Caesar Cipher is an extension of the traditional Caesar Cipher that allows forvariable shift values for each character in the plaintext. Instead of applying the same shift value to all characters, each character may be shifted by a different amount based on a predefined rule or key.

The Modified Caesar Cipher adds complexity compared to the traditional Caesar Cipherby allowing for different shift values for each character. However, it still shares similar vulnerabilities, such as being susceptible to frequency analysis and brute force attacks if the key space is limited. It's simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet.

For example with a shift of 1, A would be replaced by B, B would become C, and so on.



Screenshot: - 04(Enter Key for Encryption)

Encryption Logic

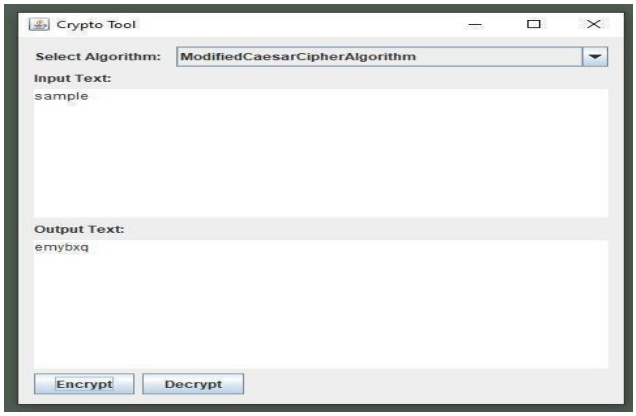
a. Iterate through each character in the plaintext.

b. Determine the shift value for the current character based on its position or any othercriteria you've defined.

c. Apply the shift to the current character. Be sure to handle wraparound (if shifting past 'z',loopback to 'a' or vice versa).

d. Append the modified character to the encrypted text.

e. Repeat this process for all characters in the plaintext.



Screenshot: - 05(Modified Caeser Cipher Encryption)

Decryption Logic

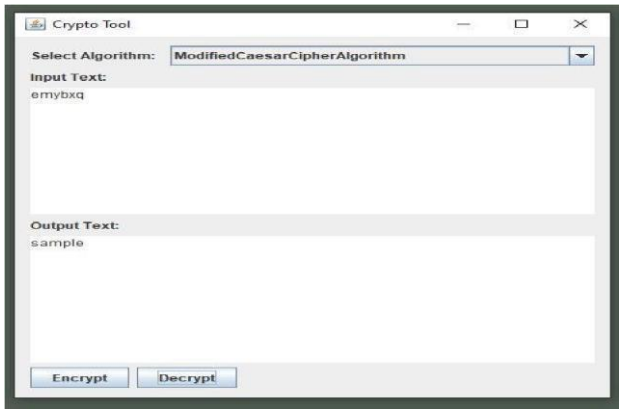
a. Perform the reverse of the encryption algorithm. Iterate through each character in the encrypted text.

b. Determine the shift value for the current character based on its position or any other criteria.

c. Apply the inverse shift (shift in the opposite direction) to the current character.

d. Append the modified character to the decrypted text.

e. Repeat this process for all characters in the encrypted text.



Screenshot: - 06(Modified Caeser Cipher Decryption)

3) AES_256_Algorithm

AES (Advanced Encryption Standard) is a symmetric encryption algorithm widely used for securing sensitive data. Here this algorithm is specifically making with the help of JAVA Cryptographic Extension Framework.

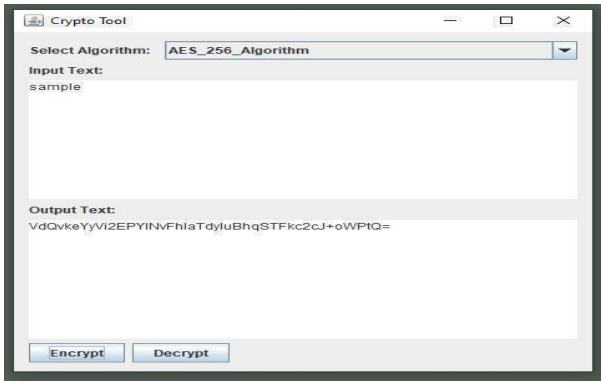
AES256, a variant of AES, employs a 256-bit key for encryption and decryption. It operates on blocks of data, typically 128 bits, and applies a series of substitution, permutation, and mixing operations (SubBytes, ShiftRows, MixColumns, and AddRoundKey) in multiple rounds (10 rounds for AES-128, 12 rounds for AES-192, and 14 rounds for AES-256) to achieve strong cryptographic security.

For example, AES-256 encrypts a 128-bit block of plaintext using a 256-bit key, producingciphertext that is computationally infeasible to decrypt without the key.

Screenshot: - 07(Enter Key for Encryption in AES-256)

Encryption Logic

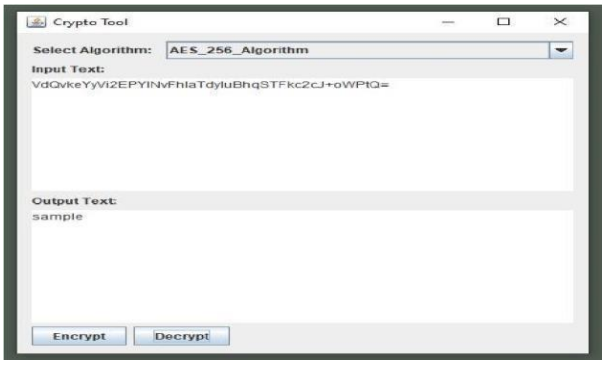
- ***Key Expansion***: Generate round keys from the original encryption key.
- ***Initial Round***: AddRoundKey step, where each byte of the state is combined with a block of the round key. ***Rounds***: Consisting of SubBytes, ShiftRows, MixColumns, and AddRoundKey transformations.
- ***Final Round***: Omit Mix Columns step. Output the final state as the ciphertext. These steps iteratively transform the plaintext block into ciphertext using a 256-bit key.



Screenshot: - 08(AES-256 Encryption)

Decryption Logic

- ***Key Expansion***: Generate round keys from the original encryption key.
- ***Initial Round***: Inverse AddRoundKey step.
- ***Rounds***: Inverse SubBytes, Inverse ShiftRows, Inverse MixColumns, and Inverse AddRoundKeytransformations.
- ***Final Round***: Omit Inverse MixColumns step. Output the final state as the decrypted plaintext.



Screenshot:- 09 (AES-256 Decryption)

4) RSA Algorithm

The RSA algorithm is a widely used asymmetric encryption algorithm named after its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman, who introduced it in 1977. RSA is primarily used for secure communication, digital signatures, and key exchange. RSA relies on the difficulty of factoring large composite numbers into their prime factors for its security. Breaking RSA encryption involves factoring the modulus n into its prime factors, which becomes computationally infeasible for sufficiently large prime numbers.

RSA is widely used in various applications, including secure communication over the internet (such as SSL/TLS), digital signatures for authentication and integrity verification, and key exchange in protocols like SSH and PGP. However, RSA encryption and decryption operations can be computationally intensive, especially for large key sizes, leading to the development of alternative algorithms such as elliptic curve cryptography (ECC) for certain applications.

1. Key Generation: Generate a public-private key pair consisting of a public key (e, n) and aprivate key (d, n).

2. Choose two large prime numbers, p and q.

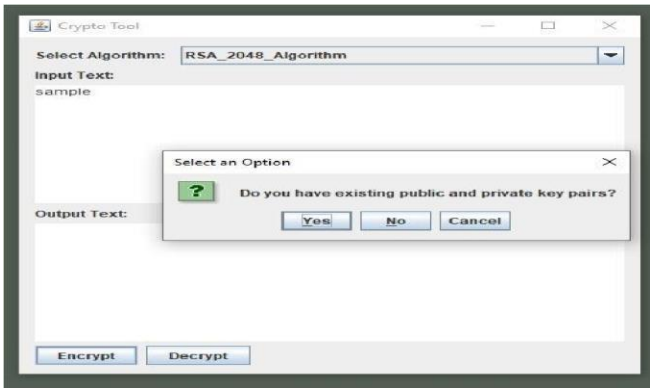
3. Compute n = p * q and φ(n) = (p-1) * (q-1).

4. Select e, relatively prime to φ(n), as the public exponent.

5. Calculated as the modular multiplicative inverse of e modulo φ(n).

6. Encryption: To encrypt a message m, compute ciphertext c = m^e mod n using the recipient's public key.

7. Decryption: To decrypt the ciphertext c, compute the plaintext m = c^d mod n using therecipient's private key.

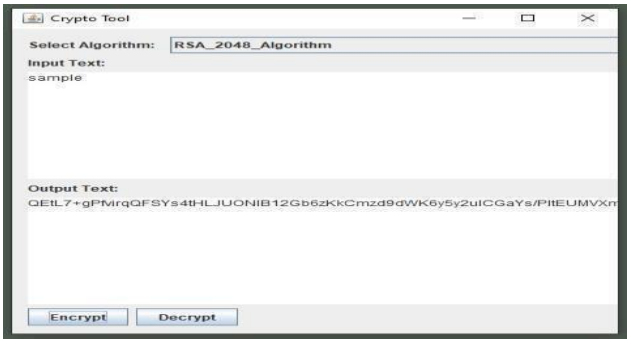


Screenshot:-10(Asking for existing Public Key & Private Key)

Encryption Logic

To encrypt a message m, compute ciphertext

c = m^e mod n using the recipient's public key (e, n).

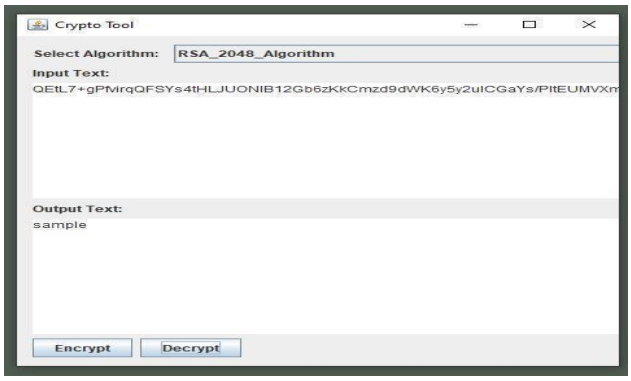


Screenshot: - 11(RSA Encryption)

Decryption Logic

To decrypt the ciphertext c, compute the plaintext

m = c^d mod n using the recipient's private key.



Screenshot: -12 (RSA Decryption)

5) Arithmetic No Key Algorithm

The arithmetic algorithm is based entirely on the integer array, which is a converted version ofa byte array. The byte array is derived from the ASCII values of the characters from the input file. For simplicity and understanding, let's consider the content of the input file as "ABCD". The tool reads this input file in the form of ASCII values, resulting in the byte array: {65,66, 67, 68}. This byte array can vary in size based on the input file; the tool creates an array of that size, with each element having a byte data type.

We then convert this byte array into an integer array as follows: {65, 66, 67, 68}. In this integer array, each element has an int data type. This preprocessing step is necessary for thealgorithm as it operates on integer arrays.

1. Initial Round:

Read the text from the file. Convert it to the byte array of Ascii value of each character present over in the input file

2. Rounds:

- Convert the type of Array as we required whole byteArray into an intArray for further process

- Difference between the terms in Arrays and store as same position starting from index-0 till length of the array whereas the index-0 element remains constant for this round.

Code:

```
//ROUND-1
// Convert byte array to int array
  for (int i = 1; i<byteArray.length; i++) { intArray[i] = byteArray[i] - byteArray[i - 1];
}
```

- In the resultant array we performed some arithmetic operations like addition and multiplication for little complexity

Code:

```
//ROUND-2
for (int i = 0; i<intArray.length; i++) { intArray[i] = ((intArray[i] + 25) * 10) ;
}
```

- Sum the terms in Resultant Arrays and store as same position starting from index-0 till length of the array whereas the index-0 element remains constant for this round.

Code:

```
//ROUND-3
for (int i = 1; i<intArray.length; i++) { intArray[i] = intArray[i] + intArray[i-1] ;
}
```

• In the resultant array we performed subtraction operations for bit complexity.

Code:

//ROUND-4

for (int i = 0; i<intArray.length; i++) {

intArray[i] = intArray[i] - 100 ;

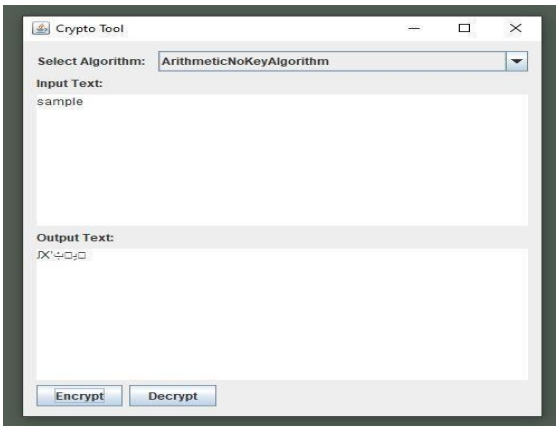
}

3) Final Round:

Concern about readability we convert back this ASCII values to characters.

Code:

// Convert back to ASCII and create encrypted
stringStringBuilder encryptedText = new
StringBuilder();
for (int value :intArray){
encryptedText.append((char) value);
}



Screenshot:-13(Arithmetic No Key Encryption)

Decryption Logic

1. Initial Round:

Read the text from the file. Convert it to the byte array of Ascii value of each character presentover in the input file

2. Rounds:

• Convert the type of Array as we required whole byteArray into an intArray for further process.

• On intArray, we performed Addition operations for bit complexity.

Code:

```
//ROUND-4
for (int i = 0; i<intArray.length; i++) {
intArray[i] = intArray[i] +100 ;
}
```

• Difference in the terms in Resultant Arrays and store as same position starting from intArray.length-1 till index0 of the array whereas the index-0 element remains constant forthis round.

Code:

```
//ROUND-3
for (int i =intArray.length-1; i>0; i--) {
intArray[i] = intArray[i] -intArray[i-1] ;
}
```

- In the resultant array we performed some arithmetic operations like addition and multiplication for little complexity

Code:

```
//ROUND-2
for (int i = 0; i<intArray.length; i++) {
intArray[i] = ((intArray[i] / 10) -25) ;
}
```

Sum the terms in Resultant Arrays and store as same position starting from index-0 tilllength of the array whereas the index-0 element remains constant for this round.

Code:

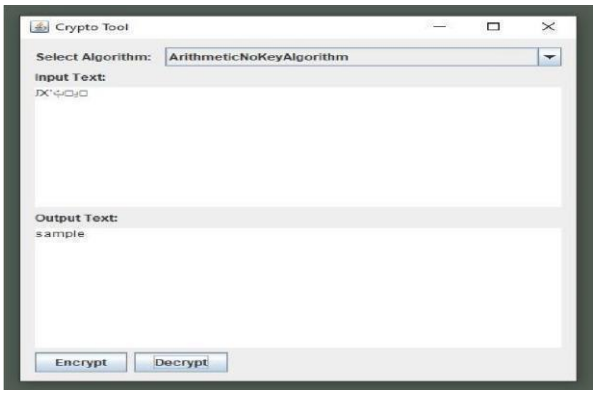
```
//ROUND-3
for (int i = 1; i<intArray.length;i++) {
intArray[i] = intArray[i] + intArray[i1] ;
}
```

3) Final Round:

Concern about readability we convert back this ASCII values to characters.

Code:

```
StringBuilder decryptedText = new
StringBuilder();for(int value : intArray){
decryptedText.append((char) value);
}
```



Screenshot:- 14(Arithmetic No Key Decryption)

Advantages:

1. Simple Implementation: The algorithm is relatively easy to understand and implement, making it accessible for developers.

2. No Key Required: As the name suggests, it operates without a key, which simplifies theencryption process.

3. Readability: It ensures readability by converting ASCII values back to characters in the finalround, which can be beneficial depending on the use case.

4. Variable Input Size: It can handle input files of varying sizes, adapting the array size accordingly.

Disadvantages:

1. Limited Security: Since it operates without a key, it might not provide robust security againstsophisticated attacks.

2. Potential for Information Loss: Depending on the operations performed, there's a risk of losinginformation during encryption and decryption.

3. Simplicity Might Lead to Vulnerabilities: The simplicity that makes it easy to implementcould also make it more vulnerable to attacks or exploitation.

6) Bit Manipulation Fixed Key Algorithm

Bit manipulation algorithms are totally based on the bits of each element of the integer array, which is a converted version of a byte array. The byte array is received from the ASCII values of the characters from the input file. For simplicity and understanding, suppose "ABCD" is the content of the input file. Then the tool reads that input file in the form of ASCII values, resulting in the byte array as follows: Byte array = {65, 66, 67, 68} Now, this byte array can have a variable size, depending on what you give in the input file. The tool will create an array of that size, and each element will have the data type byte. We then convert that byte array into an integer array as follows: Int array = {65, 66, 67, 68} In this integer array, each element has the data type int. Now, all operations are done on the bit level of each element in the integer array. So now, each operation is performed on the bits as input for further processing.

Encryption Logic

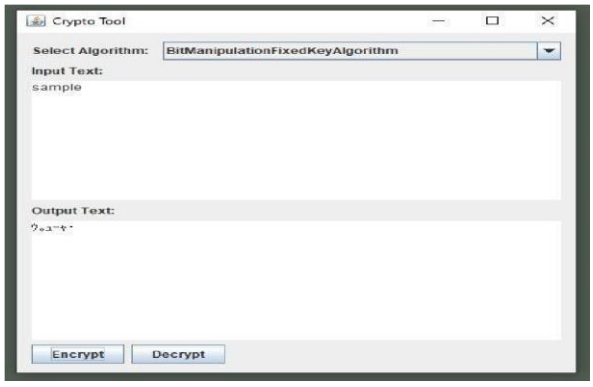
Step 1: Convert plaintext to ASCII byte array

Step 2: Convert byte array to int array(Convert byte to unsigned int)

Step 3: Bit Operation withevery single bit

Step 4: XOR each element with constant

Step 5: Display as ASCII character string.



Screenshot:- 15(Bit Manipulation Fixed Key Encryption)

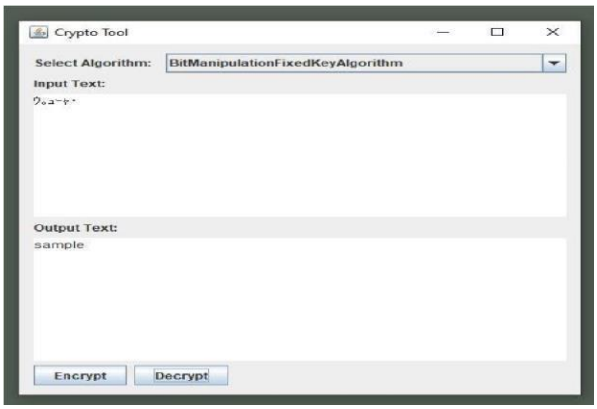
Decryption Logic

Step 1: Convert encrypted text to int array

Step 2: XOR each element with constant

Step3: Bit Operation with every single bit

Step4: Display as ASCII character string.



Screenshot:-16(Bit Manipulation Fixed Key Decryption)

Advantages:

1. Bit-Level Operations: Operating on the bit level can provide finer control over encryption anddecryption processes.

2. Fixed Key: Using a fixed key simplifies the encryption and decryption processes, reducing theoverhead of managing and exchanging keys.

3. Efficiency: Bit manipulation operations can be efficient in terms of computational resourcescompared to arithmetic operations.

4. Potential for High Security: Depending on the complexity of the bit manipulation operations and the strength of the fixed key, it can offer higher security compared to simpler algorithms.

Disadvantages:

1. Complexity: Bit manipulation operations can be complex to understand and implement,requiring a deeper understanding of bitwise operations.

2. Limited Flexibility: Using a fixed key means that the algorithm might lack flexibility inadapting to different security requirements or scenarios.

3. Risk of Key Exposure: Since the key is fixed, if it's compromised, all data encrypted usingthat key becomes vulnerable.

4. Loss of Readability: Bit-level operations might result in encrypted data that's less humanreadable compared to ASCII-based encryption.

Result and Analysis

Sr no.	Algorithm	Plain text	Time taken to encrypt (in milliseconds)	Memory used (in bytes)
1.	Caesar Cipher	1.sample	**0.1731**	**3344096**
		2.Hello World@123	**0.0173**	**5359480**
		3.Secure Data	**0.0162**	**3832456**
		4.Hi there we are testing an algorithm	**0.5168**	**6863048**
2.	Modified Caesar Cipher	1.Sample	**2185**	**4200616**
		2.Hello World@123	**678**	**5839768**
		3. Secure Data	**729**	**5179856**
		4.Hi there we are testing an algorithm	**816**	**4347840**
3.	AES-256 Algorithm	1.Sample	**123.0**	**6756344**
		2.Hello World@123	**105.0**	**8229096**
		3. Secure Data	**115.0**	**5523024**
		4.Hi there we are testing an	**416.0**	**5179472**

		algorithm		
4.	RSA Algorithm	1.Sample	**5**	**87512**
		2.Hello World@123	**84**	**1067592**
		3. Secure Data	**92**	**1167936**
		4.Hi there we are testing an algorithm	**113**	**1140224**
5.	Arithmetic No key Encryption	1.Sample	**0.0779**	**3247672**
		2.Hello World@123	**0.0468**	**3677208**
		3. Secure Data	**0.0213**	**3444192**
		4.Hi there we are testing an algorithm	**0.0402**	**3712640**
6.	Bit Manipulation Fixed key encryption	1.Sample	**0.0242**	**4381256**
		2.Hello World@123	**0.0231**	**3335872**
		3. Secure Data	**0.0233**	**3185144**
		4.Hi there we are testing an algorithm	**0.0287**	**3430104**

The table gives a comparison of different encryption algorithms based on their performance metrics, including the time taken to encrypt, memory usage, and example cipher texts.

Caesar Cipher:

1. It's a basic encryption method that shifts letters by a fixed amount.
2. It's relatively fast, taking a small fraction of a second to encrypt even long messages.
3. It uses a moderate amount of memory.
4. Example: Encrypts "Sample" in 0.1731 milliseconds.

Modified Caesar Cipher:

1. Similar to the classic Caesar Cipher but with some changes.
2. Encryption takes longer compared to the original Caesar Cipher.
3. Uses a moderate amount of memory.
4. Example: Encrypts "Hello World@123" in678 milliseconds.

AES-256 Algorithm:

1. A more advanced encryption standard known for its strong security.
2. Encryption is slower compared to the Caesar Cipher variants.
3. Requires a higher amount of memory.
4. Example: Encrypts "Secure Data" in 115milliseconds.

RSA Algorithm:

1. A public-key encryption method.
2. Encryption times are higher compared to symmetric key algorithms.
3. Uses a moderate amount of memory.
4. Example: Encrypts "Hi there we are testing an algorithm" in 113 milliseconds.

Arithmetic No Key Encryption:

1. An encryption method without using a specific key.
2. It's fast and uses relatively low memory.
3. Example: Encrypts "Sample" in 0.0779milliseconds.

Bit Manipulation Fixed Key Encryption:

1. Encrypts data using fixed bit manipulation techniques.
2. Fast encryption with moderate memory usage.
3. Example: Encrypts "Hello World@123" in0.0231 milliseconds.

In summary, different encryption methods offer varying levels of security, speed, and memoryusage. While simpler methods like the Caesar Cipher are fast and require less memory, more advanced algorithms like AES and RSA provide stronger security but at the cost of increased computational resources.

FEATURES

1. Multiple Encryption Algorithms
2. Flexibility in accessing files
3. Stand Alone Application.
4. Text based and File based Input and output both the options are available for the user.
5. Optimized memory utilization.
6. Modular Software Designing approach is used.
7. Iterative development approach used

Advantages

1. Stand-alone application:-We have created stand-alone application which provides security onlocal level basis and does not require internet
2. Simple GUI
3. Text based and File based Input and output both the options are available for the user.
4. All controls are given to the users.
5. Faster upgrades with new algorithm modules
6. Modular design

Disadvantages

1. Due to limitation on network, sharing of encrypted data is hard
2. Overhead of algorithm selection for users other than developer
3. Development language limitation

FUTURE SCOPE

- **Algorithm Optimization**: Investigate methods to optimize the performance and efficiency of the implemented algorithms, such as enhancing encryption and decryption speed or reducing memory usage, to ensure scalability and applicability in largescale systems.

- **Security Analysis and Improvements**: Conduct rigorous security analysis of the implemented algorithms to identify potential vulnerabilities and develop robust countermeasures. Additionally, explore the integration of post-quantum cryptography techniques to enhance resistance against quantum computing threats.

- **User Authentication and Access Control**: Extend the platform to include features for user authentication and access control, such as multi-factor authentication and role-based access control, to enhance security and usability in real-world applications.

- **Integration with Cloud Services**: Explore the integration of the platform with cloud services and distributed computing frameworks to facilitate secure data storage, processing, and communication in cloud-based environments.

CONCLUSION

In conclusion, the development of a comprehensive platform for symmetric and asymmetric encryption algorithms represents a significant advancement in the field of cryptography. By incorporating major encryption modes such as Caesar Cipher, AES, Modified Caesar Cipher, and RSA, alongside novel algorithms like Arithmetic No Key Algorithm and Bit Manipulation Algorithm, the platform provides users with a versatile toolkit for secure communication and data protection.

Through modular development and integration with the Java Cryptographic Extension Framework, the platform offers flexibility, usability, and educational value to users ranging from cryptography enthusiasts to professionals.

Future endeavors could focus on algorithm optimization, security analysis, user authentication, access control, and integration with cloud services, ultimately advancing the platform's capabilities and relevance in diverse real-world scenarios.

REFERENCES

1. **A Handbook of Applied Cryptography** by Alfred J. Menezes, Paul C. Van Oorschot and Scott A. Vanstone, CRC Press Series on Discrete Mathematics and Its Applications.

2. Bhuman Vyas. “Security Challenges and Solutions in Java Application Development”, ISSN: 2319-5045 Volume 12, Issue 2, July-December, 2023.

3. Zhengyi Lu. "Analysis on AES encryption standard and safety", Proc. SPIE 12462, Third International Symposium on Computer Engineering and Intelligent Communications (ISCEIC 2022), 1246215 (2 February2023).

4. Yashar Salami, Vahid Khajehvand, Esmaeil Zeinali. “Cryptographic Algorithms: A Review ofthe Literature, Weaknesses and Open Challenges”, Department of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran Received 07 April 2023; Accepted 31 May 2023 .

5. Arwa Zabian, Shakir Mrayyen, Abram magdy Jonan, Tareq Al − shaikh, Mohammed tthazi Al
   − khaiyat. “Multi-layer encryption algorithm for data integrity in cloud computing”, International Journal Artificial Intelligent and Informatics 63 Vol. 3, No. 2, December 2022.

6. M. A. Al-Shabi. “A Survey on Symmetric and Asymmetric Cryptography Algorithms in information Security” International Journal of Scientific and Research Publications, Volume9, Issue 3, March 2019.

7. Prof. (Dr.) Amit Verma, Anjali Gakhar. “Analysis of Tools and Techniques in cryptography”, FPInternational Journal of Computer Science Research (IJCSR) Volume2, Issue 1, Pages 37- 44, March 2015.

8. Punita Meelu & Sitender Malik. “AES: A SYMMETRIC KEY CRYPTOGRAPHIC SYSTEM”, International Journal of Information Technology and Knowledge Management January-June 2011.

9. Tanusree Saha. "Complement-Based Modified Approach to Secure Small Text Message Combining Triangulation Method”, International Journal of innovation in Engineering and Technology (IJIET),Volume 6, Issue 4 April 2016.

10. Deepak & Parveen “Modern Encryption and Decryption Algorithm based on ASCII Value and Binary Operations” International Journal of Computer Applications (0975 – 8887) Volume 172 – No.1, August 2017.

11. Shashi Gautam, Shubha Mishra,& Dr. Manish Shrivastava’s “An Enhanced Encryption Technique using BCD and Bit Complementation” International Research Journal of Engineering and Technology (IRJET), Volume: 04 Issue: 01 | Jan -2017.

International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 11 Issue: 04 | Apr 2024 www.irjet.net p-ISSN: 2395-0072

A Comprehensive Platform for Symmetric and Asymmetric Cryptographic Algorithms: Bridging Ancient and Modern Encryption Techniques

Prathamesh L. Khandelwal[1], Prof. Pallavi H. Dhole[2], Khushi R. Kawanpure[3] , Pranoti P. Pohokar[4] , Puja R. Jadhav[5],Tanay A. Nanglia[6]

[1]*Student, Sipna College of Engineering and Technology, Maharashtra, India*
[2]*Assistant Professor, Sipna College of Engineering and Technology, Maharashtra, India*
[3]*Student, Sipna College of Engineering and Technology, Maharashtra, India*
[4]*Student, Sipna College of Engineering and Technology, Maharashtra, India*
[5]*Student, Sipna College of Engineering and Technology, Maharashtra, India*
[6]*Student, Sipna College of Engineering and Technology, Maharashtra, India*

Abstract - The study presents a comprehensive exploration of cryptographic techniques, tracing their evolution from ancient historical periods to contemporary technological advancements. Motivated by the need for a versatile platform encompassing both symmetric and asymmetric encryption algorithms, our research introduces a standalone application designed to serve as a unified hub for cryptographic operations. This platform integrates major encryption modes, including Caesar Cipher, AES Algorithm, Modified Caesar Cipher, and RSA, alongside two proprietary algorithms, namely the Arithmetic No Key Algorithm and Bit Manipulation Algorithm. Leveraging modular development principles and the Java Cryptographic Extension Framework, the application exhibits enhanced flexibility and usability, catering to users ranging from cryptography enthusiasts to professionals. By using Dynamic Loading, the application seamlessly ties together diverse cryptographic tools, promoting worldwide awareness of cryptography and its applications. Through a holistic approach, our research endeavors to bridge classic and modern cryptographic methods, thereby fostering a deeper understanding and appreciation of the intricacies of encryption and decryption techniques in today's digital landscape.

***Key Words**: [Symmetric Key Cryptography, Modified Caesar Cipher, AES, RSA, Arithmetic No Key, Bit Manipulation Fixed Key]*

INTRODUCTION

As Technique of Encryption and decryption is not very hard to understand because we already using different encryption methods to transfer our message since from Ancient historical Period to today's technological era. Here we are providing platform for both the symmetric as well as Asymmetric key cryptographic algorithms specifically keeping its eye on all-in-one encryption standards via incorporating major four encryption modes Caesar Cipher, AES Algorithm, Modified Caesar cipher, RSA. Six Algorithms we have inserted here out of which two are indigenously made named as Arithmetic No Key Algorithm and Bit Manipulation Algorithm. Motivation for building this project comes from software Engineering Project where we come up with lots of ideas and finally, we finalize this topic.

As there are lots of cryptographic tools and website are available in the world of Cryptography but we are trying to make one such mechanism where we can tie both the symmetric and asymmetric key algorithms with each other by using Dynamic Loading. It is a Stand-Alone Application which can perform multiple Encryption via using Java (JDK 21 and above). This Stand-Alone Application system that aims enhancing worldwide awareness of cryptography and its uses; JAVA Cryptographic extension Framework is the base of these proposed system mechanism. In this application we have used modular development hence flexibility of application is increased. Combined efforts have led to the development of a Stand-alone application that shows several classic and modern methods of cryptography, amongst which: Caesar ciphers and various methods of cryptanalysis.

I. Literature Review

- "THE HANDBOOK ON APPLIED CRYPTOGRAPHY" has given a brief account on cryptographic tool working and uses [1].

- Bhuman Vyas in "Security Challenges and Solutions in Java Application Development" addresses the security landscape specific to Java applications. Vyas explores vulnerabilities and offers solutions tailored to Java environments, likely encompassing topics such as secure coding practices, access control, and encryption techniques [2].

- Zhengyi Lu conducts an in-depth analysis of the Advanced Encryption Standard (AES) in "Analysis on AES encryption standard and safety". This analysis, presented at the Third International Symposium on Computer Engineering and Intelligent Communications, provides insights into the safety and efficacy of AES, a fundamental encryption standard used in Java application security [3].

- Yashar Salami, Vahid Khajehvand, and Esmaeil Zeinali's work titled "Cryptographic Algorithms: A Review of the Literature, Weaknesses and Open Challenges" offers a comprehensive review of cryptographic algorithms. Their paper, likely covering both symmetric and asymmetric cryptography, highlights weaknesses and challenges, providing valuable insights for implementing secure cryptographic solutions in Java applications [4].

- Arwa Zabian, Shakir Mrayyen, Abram Magdy Jonan, Tareq Al-Shaikh, and Mohammed Tthazi Al-Khaiyat propose a "Multi-layer encryption algorithm for data integrity in cloud computing". Although not directly focused on Java, this paper introduces encryption techniques relevant to securing data in distributed environments, which are often encountered in Java based cloud applications [5].

- M. A. Al-Shabi's "A Survey on Symmetric and Asymmetric Cryptography Algorithms in Information Security" provides a survey of cryptographic algorithms. This survey, likely encompassing various encryption techniques, offers insights into their applicability and effectiveness in information security, pertinent for securing Java applications [6].

- Prof. (Dr.) Amit Verma and Anjali Gakhar analyze "Tools and Techniques in Cryptography", shedding light on available cryptographic tools and methodologies. While not Java-specific, this analysis offers valuable insights into integrating cryptographic techniques into Java application security frameworks [7].

- Punita Meelu and Sitender Malik explore "AES: A Symmetric Key Cryptographic System". Although an older publication, this paper remains relevant as it provides a detailed examination of AES, a cornerstone encryption algorithm utilized in Java application security for securing sensitive data [8].

- Tanusree Saha's "Complement-Based Modified Approach to Secure Small Text Message Combining Triangulation Method" proposes an enhanced Symmetric key encryption algorithm which is the combination of substitution technique, 2's compliment technique and Triangulation Method[9].

- Deepak & Parveen work "Modern Encryption and Decryption Algorithm based on ASCII Value,Integer array, Byte array and Binary Operations" proposed an algorithm based on ASCII values and binary operations for both encryption and decryption to enhance the security[10].

- Shashi Gautam, Shubha Mishra, & Dr. Manish Shrivastava work titled "An Enhanced Encryption Technique using BCD and Bit Complementation". They proposed a cryptographic algorithm based on binary operations and with basic CPU computation. This algorithm uses BCD (binary coded decimal), 1's complement for data encryption and 2's complement for key encryption [11].

II. Features

1. Multiple Encryption Algorithm
2. Flexibility in accessing file
3. Stand Alone Application.
4. Text based and File based Input and output both the options are available for the user.
5. Optimized memory utilization.
6. Modular Software Designing approach is used.
7. Iterative development approach used

III. Proposed System

This mechanism provides both the symmetric and asymmetric key Cryptographic Algorithms.

1) To place both these algorithms at one we have proposed here common Interface called Crypto Algorithm Interface and both the symmetric and asymmetric algorithms have to compulsorily implement.

2) The interface CryptoAlgorithm consists of two methods - encrypt() and decrypt(). Both of these methods have a return type and parameter of String to ensure that they accept and return Strings.

3) The AlgorithmLoader is a module designed to dynamically load and manage cryptographic algorithms used in the application. Its purpose is to provide a flexible way to integrate different encryption and decryption algorithms without tightly coupling them to the main application logic.

4) By creating the AlgorithmLoader, we avoid hardcoding specific cryptographic algorithms directly into the application code. This dynamic loading process at runtime enables our application to support multiple algorithms without altering the core code. This not only saves time and effort but also enhances the flexibility and scalability of the System.

5) Initialization: The AlgorithmLoader is initialized when the application starts.

6) Scanning for Algorithm Classes: It scans the specified package (in our implementation "cryptoAlgos") for classes that implement the CryptoAlgorithm interface. Loading Algorithm Classes: It loads the algorithm classes dynamically using Java reflection. The classes are identified, and Java reflection is used to load them into the application dynamically. This is done using the Class.forName() method, which takes the fully qualified class name as a string and returns the Class object for that class.

7) Storing Algorithm Classes: The loaded algorithm classes are stored in a list or data structure for easy access.

8) Algorithm Selection: When needed, the application can request a specific algorithm by its name from the AlgorithmLoader.

9) Instantiation: After loading the names into the System, the user is asked to select a specific Algorithm module class, and then the Reflection is used again to instantiate the objects of the selected class. This is typically done using the Class.newInstance() method or by calling a constructor using Constructor.newInstance().

10) Returning Algorithm Object: It returns the instantiated algorithm object to the application for encryption or decryption.

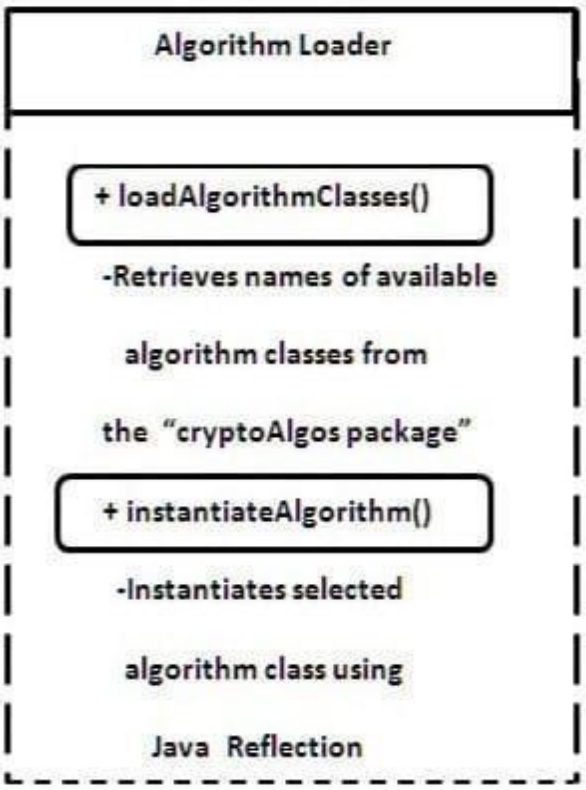


Figure: - 01(Working of Algorithm Loader)

METHODOLOGY

Proposed Application specially includes total 6 algorithms out of which 4are studied by us and 2 of them are created by us proprietary algorithms.

Key terms

1) KEY =A key in cryptography is a piece of information, usually a string of numbers or letters that are stored in a file, which, when processed

through a cryptographic algorithm, can encode or decode cryptographic data.

2) Encryption =Encryption is a way of scrambling data so that only authorized parties can understand the information.

3) Decryption=The process of encryption transforms information from its original format called plaintext into an unreadable format called ciphertext while it is being shared or transmitted.

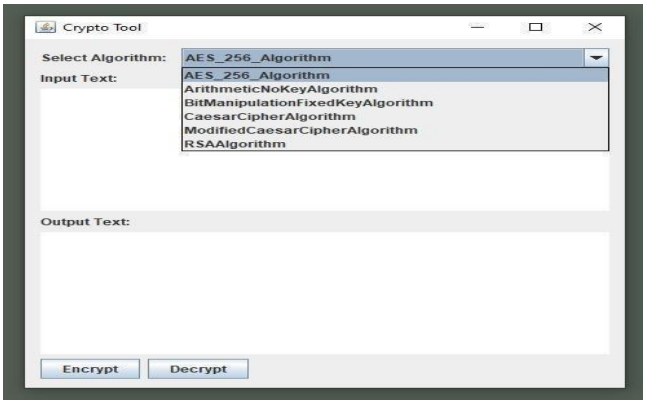


Figure: - 02(Application Interface)

Algorithms Used

1) Caeser Cipher

It is a substitution Cipher which replaced each letter of given plaintext with 3 positions down the alphabet.

Encryption Logic

a. Choose a shift value (e.g., 3).
b. Iterate through each character in the plaintext.
c. Shift each letter in the plaintext forward in the alphabet by the chosen shift value.
d. Handle wraparound if the shift extends beyond the bounds of the alphabet.
e. Output the encrypted text.

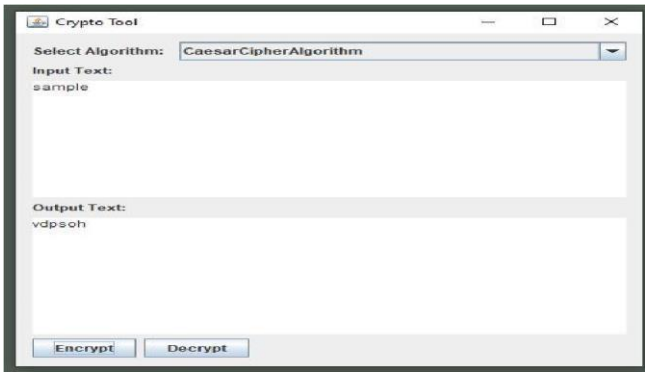


Figure: - 03(Caeser Cipher Encryption)

Decryption Logic

a. Given the ciphertext and the known shift value (e.g., 3).
b. Iterate through each character in the ciphertext.
c. Shift each letter in the ciphertext backward in the alphabet by the chosen shift value.
d. Handle wraparound if the shift extends beyond the bounds of the alphabet.
e. Output the decrypted text.

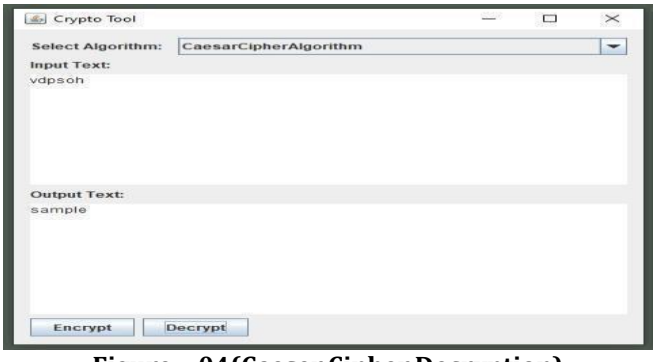


Figure :- 04(Caeser Cipher Decryption)

2) Modified Caeser Cipher

It's simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet.

For example with a shift of 2, A would be replaced by C, B would become D, and so on.

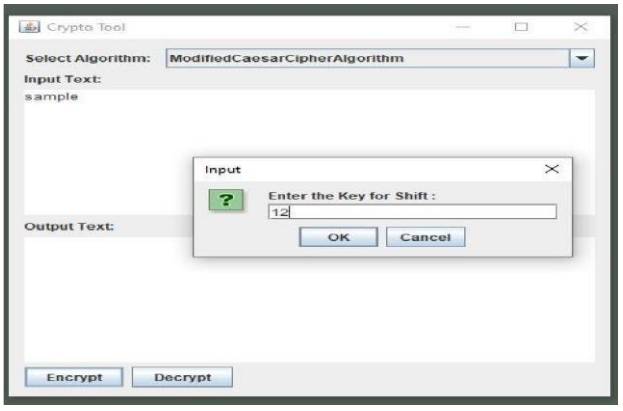


Figure: - 05(Enter Key for Encryption)

Encryption Logic

a) Iterate through each character in the plaintext.
b) Determine the shift value for the current character based on its position or any other criteria you've defined.
c) Apply the shift to the current character. Be sure to handle wraparound (if shifting past 'z', loop back to 'a' or vice versa).
d) Append the modified character to the encrypted text.
e) Repeat this process for all characters in the plaintext.

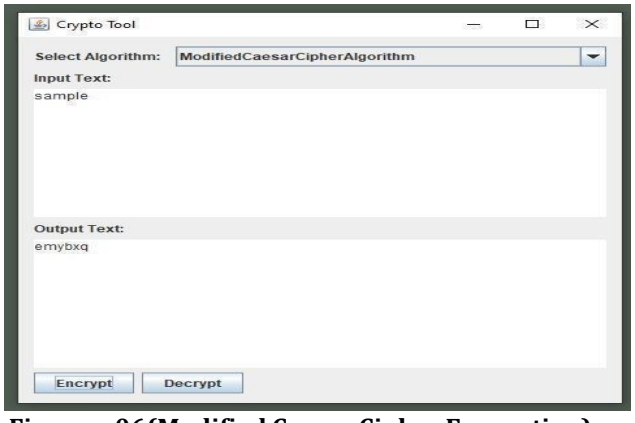


Figure: - 06(Modified Caeser Cipher Encryption)

Decryption Logic

a. Perform the reverse of the encryption algorithm. Iterate through each character in the encrypted text.
b. Determine the shift value for the current character based on its position or any other criteria.
c. Apply the inverse shift (shift in the opposite direction) to the current character.
d. Append the modified character to the decrypted text.
e. Repeat this process for all characters in the encrypted text.

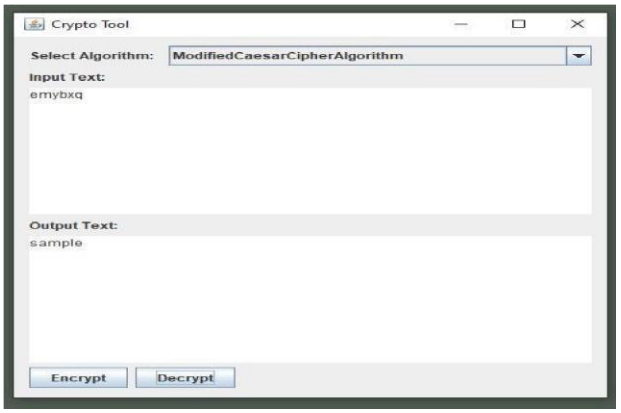


Figure: - 07(Modified Caeser Cipher Decryption)

3) AES_256_Algorithm

AES (Advanced Encryption Standard) is a symmetric encryption algorithm majorly used for securing sensitive and confidential data. Here this algorithm is specifically making with the help of JAVA Cryptographic Extension Framework. AES256, a variant of AES, employs a 256-bit key for encryption and decryption. It operates on blocks of data, typically 128 bits, and applies a series of substitution, permutation, and mixing operations (SubBytes, ShiftRows, MixColumns, and AddRoundKey) in multiple rounds (10 rounds for AES-128, 12 rounds for AES-192, and 14 rounds for AES-256) to achieve strong cryptographic security. For example, AES-256 encrypts a 128-bit block of plaintext using a 256-bit key, producing ciphertext that is computationally infeasible to decrypt without the key.

Figure: - 08(Enter Key for Encryption in AES-256)

Encryption Logic

Key Expansion: Generate round keys from the original encryption key.

Initial Round: AddRoundKey step, where each byte of the state is combined with a block of the round key. ***Rounds***: Consisting of SubBytes, ShiftRows, MixColumns, and AddRoundKey transformations.

Final Round: Omit Mix Columns step.

Output the final state as the ciphertext.

These steps iteratively transform the plaintext block into ciphertext using a 256-bit key.

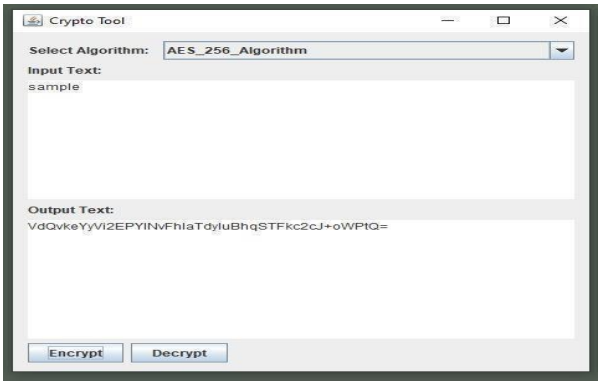


Figure: - 09 (AES-256 Encryption)

Decryption Logic

Key Expansion: Generate round keys from the original encryption key.

Initial Round: Inverse AddRoundKey step.
Rounds: Inverse SubBytes, Inverse ShiftRows, Inverse MixColumns, and Inverse AddRoundKey transformations.
Final Round: Omit Inverse MixColumns step.
Output the final state as the decrypted plaintext.

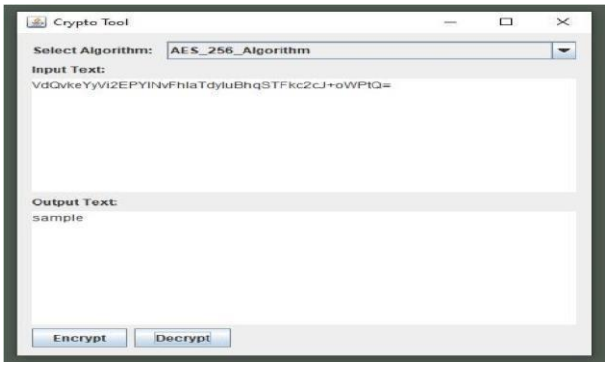


Figure: - 10 (AES-256 Decryption)

4) RSA Algorithm

RSA (Rivest-Shamir-Adleman) is a widely-used asymmetric encryption algorithm. We used here JAVA Cryptographic Extension Framework to stud and implement this algorithm.

1. Key Generation: Generate a public-private key pair consisting of a public key (e, n) and a private key (d, n).
2. Choose two large prime numbers, A and B.
3. Compute n = A* B and φ(n) = (A-1) * (B-1).
4. Select e, relatively prime to φ(n), as the public exponent.
5. Calculated as the modular multiplicative inverse of e modulo φ(n).
6. Encryption: To encrypt a message m, compute ciphertext c = m^e mod n using the recipient's public key.
7. Decryption: To decrypt the ciphertext c, compute the plaintext m = c^d mod n using the recipient's private key.

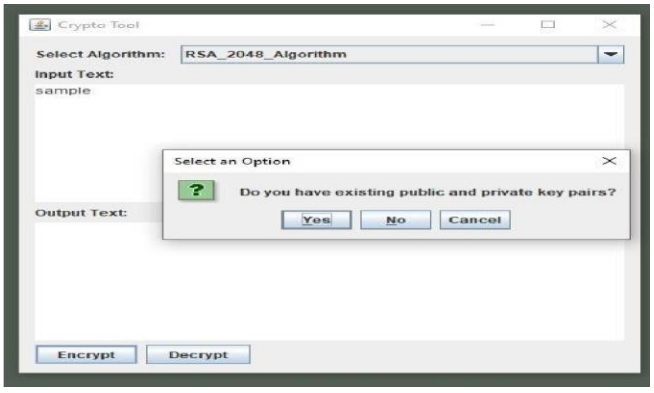


Figure: - 11(Asking for existing Public Key & Private Key)

Encryption Logic

To encrypt a message m, compute ciphertext
c = m^e mod n using the recipient's public key (e, n).

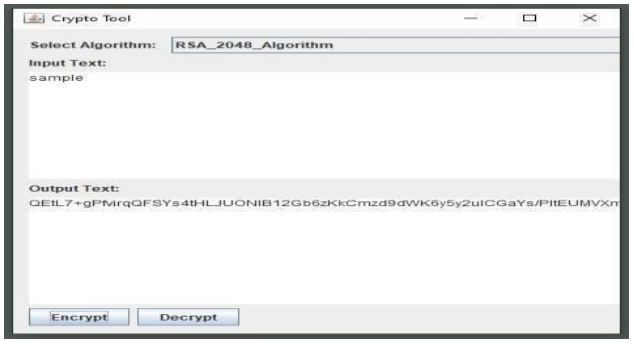


Figure: - 12(RSA Encryption)

Decryption Logic

To decrypt the ciphertext c, compute the plaintext
m = c^d mod n using the recipient's private key.

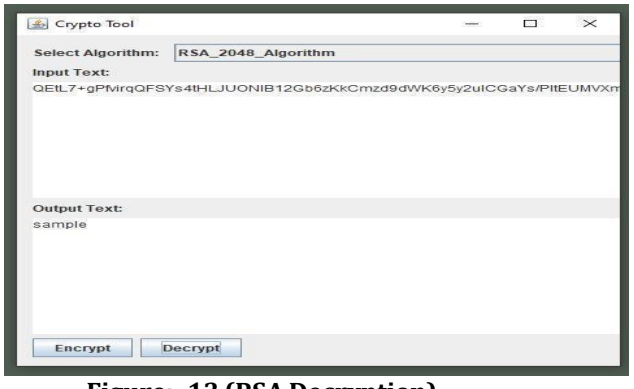


Figure: -13 (RSA Decryption)

5) Arithmetic No Key Algorithm

The arithmetic algorithm is based entirely on the integer array, which is a converted version of a byte array. The byte array is derived from the ASCII values of the characters from the input file. For simplicity and understanding, let's consider the content of the input file as "ABCD". The tool reads this input file in the form of ASCII values, resulting in the byte array: {65, 66, 67, 68}. This byte array can vary in size based on the input file; the tool creates an array of that size, with each element having a byte data type. We then convert this byte array into an integer array as follows: {65, 66, 67, 68}. In this integer array, each element has an int data type. This preprocessing step is necessary for the algorithm as it operates on integer arrays.

Working

1. Initial Round:

Read the text from the file. Convert it to the byte array of Ascii value of each character present over in the input file

2. Rounds:

- Convert the type of Array as we required whole byteArray into an intArray for further process
- Difference between the terms in Arrays and store as same position starting from index-0 till length of the array whereas the index-0 element remains constant for this round.

• In the resultant array we performed some arithmetic operations like addition and multiplication for little complexity

- Sum the terms in Resultant Arrays and store as same position starting from index-0 till length of the array whereas the index-0 element remains constant for this round.

- In the resultant array we performed subtraction operations for bit complexity.

3. Final Round:

Concern about readability we convert back this ASCII values to characters.

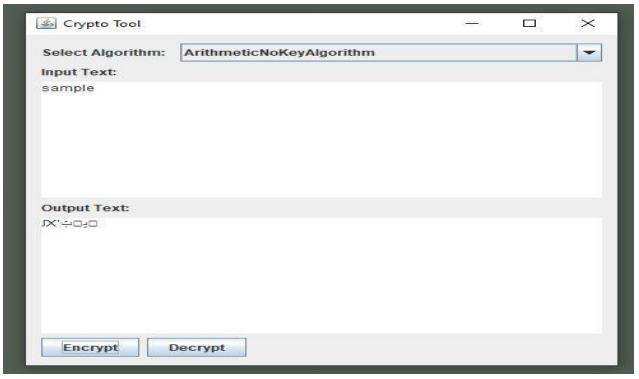


Figure: - 14(Arithmetic No Key Encryption)

Decryption Logic

1. Initial Round:
   Read the text from the file. Convert it to the byte array of Ascii value of each character present over in the input file
2. Rounds:
   - Convert the type of Array as we required whole byteArray into an intArray for further process.
   - On intArray, we performed Addition operations for bit complexity.
   - Difference in the terms in Resultant Arrays and store as same position starting from intArray.length-1 till index0 of the array whereas the index-0 element remains constant for this round.
   - In the resultant array we performed some arithmetic operations like addition and multiplication for little complexity
   - Sum the terms in Resultant Arrays and store as same position starting from index-0 till length of the array whereas the index-0 element remains constant for this round.

3. **Final Round:**
   Concern about readability we convert back this ASCII values to characters.

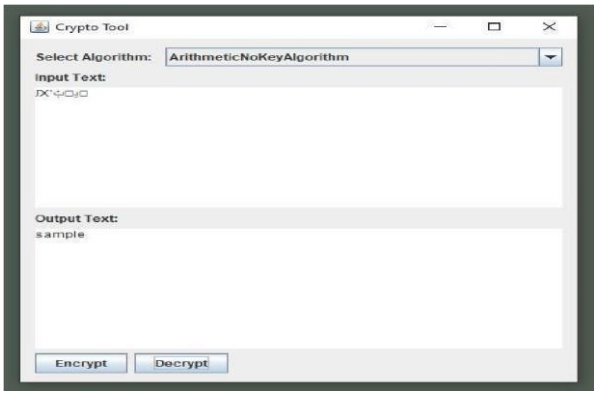


Figure :- 15(Arithmetic No Key Encryption)

Advantages:

1. Simple Implementation: The algorithm is relatively easy to understand and implement, making it accessible for developers.
2. No Key Required: As the name suggests, it operates without a key, which simplifies the encryption process.
3. Readability: It ensures readability by converting ASCII values back to characters in the final round, which can be beneficial depending on the use case.
4. Variable Input Size: It can handle input files of varying sizes, adapting the array size accordingly.

Disadvantages:

1. Limited Security: Since it operates without a key, it might not provide robust security against sophisticated attacks.
2. Potential for Information Loss: Depending on the operations performed, there's a risk of losing information during encryption and decryption.
3. Simplicity Might Lead to Vulnerabilities: The simplicity that makes it easy to implement could also make it more vulnerable to attacks or exploitation.

6) Bit Manipulation Fixed Key Algorithm

Bit manipulation algorithms are totally based on the bits of each element of the integer array, which is a converted version of a byte array. The byte array is received from the ASCII values of the characters from the input file. For simplicity and understanding, suppose "ABCD" is the content of the input file. Then the tool reads that input file in the form of ASCII values, resulting in the byte array as follows: Byte array = {65, 66, 67, 68} Now, this byte array can have a variable size, depending on what you give in the input file. The tool will create an array of that size, and each element will have the data type byte. We then convert that byte array into an integer array as follows: Int array = {65, 66, 67, 68} In this integer array, each element has the data type int. Now, all operations are done on the bit level of each element in the integer array. So now, each operation is performed on the bits as input for further processing.

Encryption Logic

Step:Convert plaintext to ASCII byte array

Step 2: Convert byte array to int array(Convert

byte to unsigned int)

Step 3: Bit Operation with every single bit

Step 4: XOR each element with constant

Step 5: Display as ASCII character string.

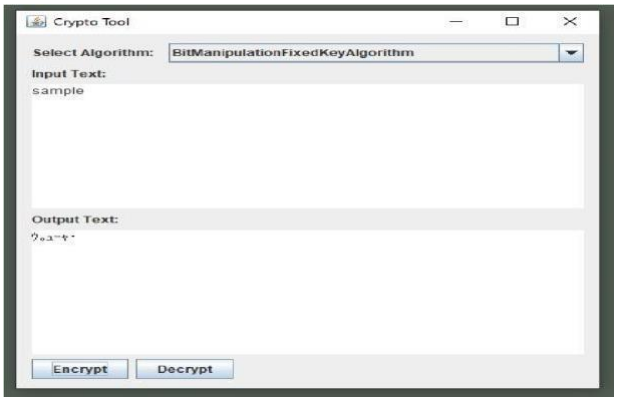


Figure :- 16(Bit Manipulation Fixed Key Encryption)

Decryption Logic

Step 1: Convert encrypted text to int array

Step 2: XOR each element with constant

Step 3: Bit Operation with every single bit

Step 4: Display as ASCII character string.

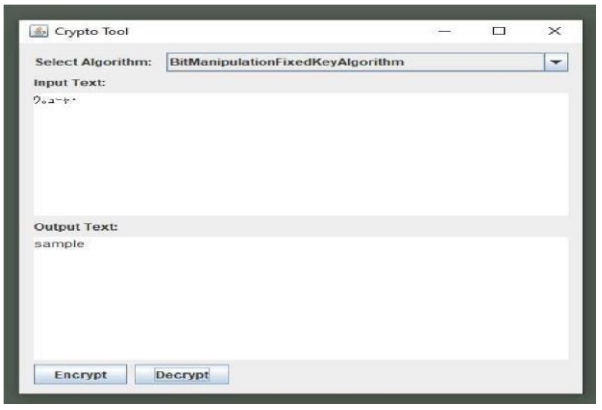


Figure:-17(Bit Manipulation Fixed Key Decryption)Advantages:

1. Bit-Level Operations: Operating on the bit level can provide finer control over encryption and decryption processes.
2. Fixed Key: Using a fixed key simplifies the encryption and decryption processes, reducing the overhead of managing and exchanging keys.
3. Efficiency: Bit manipulation operations can be efficient in terms of computational resources compared to arithmetic operations.
4. Potential for High Security: Depending on the complexity of the bit manipulation operations and the strength of the fixed key, it can offer higher security compared to simpler algorithms.

Disadvantages:

1. Complexity: Bit manipulation operations can be complex to understand and implement, requiring a deeper understanding of bitwise operations.
2. Limited Flexibility: Using a fixed key means that the algorithm might lack flexibility in adapting to different security requirements or scenarios.
3. Risk of Key Exposure: Since the key is fixed, if it's compromised, all data encrypted using that key becomes vulnerable.
4. Loss of Readability: Bit-level operations might result in encrypted data that's less human-readable compared to ASCII-based encryption.

Result and Analysis

Sr no.	Algorithm	Plain text	Time taken to encrypt (in milliseconds)	Memory used (in bytes)
1.	Ceasar Cipher	1.sample	**0.1731**	**3344096**
		2.Hello World@123	**0.0173**	**5359480**
		3.Secure Data	**0.0162**	**3832456**
		4.Hi there we are testing an algorithm	**0.5168**	**6863048**
2.	Modified Caeser Cipher	1.Sample	**2185**	**4200616**
		2.Hello World@123	**678**	**5839768**
		3. Secure Data	**729**	**5179856**
		4.Hi there we are testing an algorithm	**816**	**4347840**
3.	AES-256 Algorithm	1.Sample	**123.0**	**6756344**
		2.Hello World@123	**105.0**	**8229096**
		3. Secure Data	**115.0**	**5523024**
		4.Hi there we are testing an algorithm	**416.0**	**5179472**
4.	RSA Algorithm	1.Sample	**5**	**87512**
		2.Hello World@123	**84**	**1067592**
		3. Secure Data	**92**	**1167936**
		4.Hi there we are testing an algorithm	**113**	**1140224**
5.	Arithmetic No key Encryption	1.Sample	**0.0779**	**3247672**
		2.Hello World@123	**0.0468**	**3677208**
		3. Secure Data	**0.0213**	**3444192**
		4.Hi there we are testing an algorithm	**0.0402**	**3712640**
6.	Bit Manipulation Fixed key encryption	1.Sample	**0.0242**	**4381256**
		2.Hello World@123	**0.0231**	**3335872**
		3. Secure Data	**0.0233**	**3185144**
		4.Hi there we are testing an algorithm	**0.0287**	**3430104**

The table gives a comparison of different encryption algorithms based on their performance metrics, including the time taken to encrypt, memory usage, and example ciphertexts.

Caesar Cipher:

1) It's a basic encryption method that shifts letters by a fixed amount.
2) It's relatively fast, taking a small fraction of a second to encrypt even long messages.
3) It uses a moderate amount of memory.
4) Example: Encrypts "Sample" in 0.1731 milliseconds.

- **Modified Caesar Cipher:**

1) Similar to the classic Caesar Cipher but with some changes.
2) Encryption takes longer compared to the original Caesar Cipher.
3) Uses a moderate amount of memory.
4) Example: Encrypts "Hello World@123" in 678 milliseconds.

- **AES-256 Algorithm:**

1) A more advanced encryption standard known for its strong security.
2) Encryption is slower compared to the Caesar Cipher variants.
3) Requires a higher amount of memory.
4) Example: Encrypts "Secure Data" in 115 milliseconds.

- **RSA Algorithm:**

1) A public-key encryption method.
2) Encryption times are higher compared to symmetric key algorithms.
3) Uses a moderate amount of memory.
4) Example: Encrypts "Hi there we are testing an algorithm" in 113 milliseconds.

- **Arithmetic No Key Encryption:**

1) An encryption method without using a specific key.
2) It's fast and uses relatively low memory.
3) Example: Encrypts "Sample" in 0.0779 milliseconds.

- **Bit Manipulation Fixed Key Encryption:**

1) Encrypts data using fixed bit manipulation techniques.
2) Fast encryption with moderate memory usage.
3) Example: Encrypts "Hello World@123" in 0.0231 milliseconds.

In summary, different encryption methods offer varying levels of security, speed, and memory usage. While simpler methods like the Caesar Cipher are fast and require less memory, more advanced algorithms like AES and RSA provide stronger security but at the cost of increased computational resources.

FUTURE SCOPE

- ***Algorithm Optimization***: Investigate methods to optimize the performance and efficiency of the implemented algorithms, such as enhancing encryption and decryption speed or reducing memory usage, to ensure scalability and applicability in largescale systems.

- ***Security Analysis and Improvements***: Conduct rigorous security analysis of the implemented algorithms to identify potential vulnerabilities and develop robust countermeasures. Additionally, explore the integration of post-quantum cryptography techniques to enhance resistance against quantum computing threats.

- ***User Authentication and Access Control***: Extend the platform to include features for user authentication and access control, such as multi-factor authentication and role-based access control, to enhance security and usability in real-world applications.

- ***Integration with Cloud Services***: Explore the integration of the platform with cloud services and distributed computing frameworks to facilitate secure data storage, processing, and communication in cloud-based environments.

CONCLUSION

In conclusion, the development of a comprehensive platform for symmetric and asymmetric encryption algorithms represents a significant advancement in the field of cryptography. By incorporating major encryption modes such as Caesar Cipher, AES, Modified Caesar Cipher, and RSA, alongside novel algorithms like Arithmetic No Key Algorithm and Bit Manipulation Algorithm, the platform provides users with a versatile toolkit for secure communication and data protection.

Motivated by the need for a unified encryption solution, the project aimed to bridge ancient and modern Encryption techniques while promoting awareness of cryptography's importance in today's technological era. Through modular development and integration with the Java Cryptographic Extension Framework, the platform offers flexibility, usability, and educational value to users ranging from cryptography enthusiasts to professionals.

Looking ahead, there is ample opportunity for further research and development to enhance the platform's performance, security, and usability. Future endeavors could focus on algorithm optimization, security analysis, user authentication, access control, and integration with cloud services, ultimately advancing the platform's capabilities and relevance in diverse real-world scenarios. Overall, this research contributes to the ongoing evolution and application of cryptography in safeguarding digital information and communications.

REFERENCES

1. Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (Year). Applied Cryptography: A Handbook. CRC Press Series on Discrete Mathematics and Its Applications.

2. 2 . Vyas, B. (Year). Security Issues and Resolutions in Java Application Development. Journal Name, Volume(Issue), Pages.

3. Zhengyi Lu. "Analysis on AES encryption standard and safety", Proc. SPIE 12462, Third International Symposium on Computer Engineering and Intelligent Communications (ISCEIC 2022), 1246215 (2 February2023).

4. Salami, Y., Khajehvand, V., & Zeinali, E. (Year). A Comprehensive Review of Cryptographic Algorithms: Analysis of Existing Literature, Vulnerabilities, and Ongoing Challenges. Department of Computer and Information Technology Engineering(IT), Qazvin Branch, Islamic Azad University, Qazvin, Iran. Received on 07 April 2023; Accepted on 31 May 2023.

5. Arwa Zabian, Shakir Mrayyen, Abram magdy Jonan, Tareq Al – shaikh, Mohammed tthazi Al – khaiyat . "Multi-layer encryption algorithm for data integrity in cloud computing", International Journal Artificial Intelligent and Informatics 63 Vol. 3, No. 2, December 2022.

6. Al-Shabi, M. A. (Year). An Overview of Symmetric and Asymmetric Cryptography Algorithms in Information Security. International Journal of Scientific and Research Publications, Volume 9(Issue 3), March [2019].

7. Prof. (Dr.) Amit Verma, Anjali Gakhar. "Analysis of Tools and Techniques in cryptography", FPInternational Journal of Computer Science Research (IJCSR) Volume 2, Issue 1, Pages 37-44, March 2015.

8. Punita Meelu & Sitender Malik. "AES: A SYMMETRIC KEY CRYPTOGRAPHIC SYSTEM", International Journal of Information Technology and Knowledge Management January-June 2011.

9. Tanusree Saha. "Complement-Based Modified Approach to Secure Small Text Message Combining Triangulation Method", International Journal of innovation in Engineering and Technology (IJIET),Volume 6, Issue 4 April 2016.

10. Deepak & Parveen "Modern Encryption and Decryption Algorithm based on ASCII Value and Binary Operations" International Journal of Computer Applications (0975 – 8887) Volume 172 – No.1, August 2017.

11. Shashi Gautam, Shubha Mishra,& Dr. Manish Shrivastava's "An Enhanced Encryption Technique using BCD and Bit Complementation" International Research Journal of Engineering and Technology (IRJET), Volume: 04 Issue: 01 | Jan -2017.

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Prathamesh L. Khandelwal

In recognition the publication of the manuscript entitled

A Comprehensive Platform for Symmetric and Asymmetric Cryptographic Algorithms: Bridging Ancient and Modern Encryption Techniques

published in our Journal Volume 11 Issue 4 April 2024

Impact Factor : 8.226

www.irjet.net

Editor in Chief

E-mail : editor@irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Prof. Pallavi H. Dhole

In recognition the publication of the manuscript entitled

A Comprehensive Platform for Symmetric and Asymmetric Cryptographic Algorithms: Bridging Ancient and Modern Encryption Techniques

published in our Journal Volume 11 Issue 4 April 2024

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 8.226

www.irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Khushi R. Kawanpure

In recognition the publication of the manuscript entitled

A Comprehensive Platform for Symmetric and Asymmetric Cryptographic Algorithms: Bridging Ancient and Modern Encryption Techniques

published in our Journal Volume 11 Issue 4 April 2024

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 8.226

www.irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Pranoti P. Pohokar

In recognition the publication of the manuscript entitled

A Comprehensive Platform for Symmetric and Asymmetric Cryptographic Algorithms: Bridging Ancient and Modern Encryption Techniques

published in our Journal Volume 11 Issue 4 April 2024

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 8.226

www.irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Puja R. Jadhav

In recognition the publication of the manuscript entitled

A Comprehensive Platform for Symmetric and Asymmetric Cryptographic Algorithms: Bridging Ancient and Modern Encryption Techniques

published in our Journal Volume 11 Issue 4 April 2024

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 8.226

www.irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Tanay A. Nanglia

In recognition the publication of the manuscript entitled

A Comprehensive Platform for Symmetric and Asymmetric Cryptographic Algorithms: Bridging Ancient and Modern Encryption Techniques

published in our Journal Volume 11 Issue 4 April 2024

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 8.226

www.irjet.net