CrAlSim: A Cryptography Algorithm Simulator

Submitted in partial fulfillment of the

requirements of the degree of

Bachelor of Engineering in Information Technology

By

Avinash Singh (19101B0029)

Swapnil Mane (19101B0033)

Mihir Bist (19101B0044)

Under the Guidance of

Prof. Samuel Jacob

Department of Information Technology

Vidyalankar Institute of Technology

Wadala(E), Mumbai-400437

University of Mumbai

2022-23

CERTIFICATE OF APPROVAL

This is to certify that the project entitled

"CrAlSim: A Cryptography

Algorithm Simulator"

is a bonafide work of

Avinash Singh (19101B0029)

Swapnil Mane (19101B0033)

Mihir Bist (19101B0044)

submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the

degree of

Undergraduate in **"INFORMATION TECHNOLOGY"**.

Guide	Head of Department	Principal
(Prof. Samuel Jacob)	(Dr. Vipul Dalal)	(Dr. S.A. Patekar)

Project Report Approval for B. E.

This project report entitled ***CrAlSim: A Cryptography Algorithm Simulator***

by

1. ***Avinash Singh (19101B0029)***
2. ***Swapnil Mane (19101B0033)***
3. ***Mihir Bist (19101B0044)***

is approved for the degree of ***Bachelor of Engineering in Information Technology.***

Examiners

1._____-

2._____-

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced theoriginal sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/sourcein our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

	Name of student	Roll No.	Signature
1.	Avinash Singh	19101B0029	
2.	Swapnil Mane	19101B0033	
3.	Mihir Bist	19101B0044	

Date:

ACKNOWLEDGEMENT

We are honored to present "**CrAlSim: A Cryptography Algorithm Simulator**" as our B.E Final Year Project. We are using this opportunity to express our profound gratitude to our principal "**Dr. Sunil Patekar**" for providing us with all proper facilities and support.

We express our deepest gratitude towards our HOD "**Dr. Vipul Dalal**" for his valuable and timely advice during the various phases in our project. We would like to thankour project guide **"Prof. Samuel Jacob"** for support, patience and faith in our capabilities and for giving us flexibility in terms of working and reporting schedules. Finally, we would like to thank everyone who has helped us directly or indirectly in our project.

We would also like to thank our staff members and lab assistant for permitting us touse computer in the lab as when required. We thank our college for providing us with excellent facilities that helped us to complete and present this project.

Avinash Singh
Swapnil Mane
Mihir Bist

ABSTRACT

In order to preserve the integrity and confidentiality of the system as well as the messages transferred between users, encryption techniques are a crucial component of information security. Students will learn better if there is a graphical representation of every event taking place in the algorithm rather than them memorizing the steps happening in the cipher in order to grasp the internal workings of complex algorithms. This paper explains how a system for simulating cryptographic algorithms operates. The foundation of CrAlSim is matrix generation, gradual color-based value changes in the cryptographic algorithms' code, underlined by each function's visual representation of the computation from the first to the last encryption step. The method demonstrates how the internal conversions in an algorithm that produced the encryptions, whether it be a straightforward monoalphabetic cipher like Affine or a block cipher like AES (Advance Encryption Standard), occurs based on user inputs for a plain message or a key wherever required. A typical user can understand the source code even for developing other algorithm visualization systems because the encryption functions adjusted with the simulations are programmed in JavaScript and only HTML and CSS are used to display the web elements and styling. This paper describes the CrAlSim method in detail and talks about the outcomes.

CONTENTS

List of Figures

LIST OF TABLES

CHAPTER 1
INTRODUCTION

INTRODUCTION

Albeit learning cryptographic algorithms is difficult, security is extremely important in the internet age. And these ciphers, or encryption techniques, aid in preserving that level of anonymity amongst data streams shared by many users. Now, it would be too difficult and time-consuming for students who are interested in cryptography to attempt to learn these techniques. Therefore, we are developing a project called "CrAlSim: A Cryptography Algorithm Simulator" to facilitate learning by allowing users to just browse our website and look up the algorithms they need. From this point on, kids can practice encryption and decryption operations independently using their own customized messages and key values.

However, understanding the workings of these algorithms and the background flow is necessary for understanding encryption systems; simply having access to the encrypted text is not sufficient. Here comes our tool Simulator, which will assist users in performing this specific task. Our simulator is essentially a graphical visualization tool for these ciphers because it allows users to delve deeply into the inner workings of a cipher, see how one alphabet can change to another while still retaining its meaning when linked to a few keys, and learn how these ciphers produce accurate and well-secured copies of our actual messages so that they can be seen by anyone but understood by no one trying to eavesdrop on a conversation.

In order to help the user, understand the relative flow of all transformations, we're using basic HTML, CSS for animations, and JavaScript to write our own code and create functions that handle data access, object and subroutine callbacks, and displaying all events in the form of visual images, tables, and color highlights. The goal of this project is to develop a simulator that can live-encrypt data streams between two nodes while under specific conditions, providing the optimum opportunity for practice and learning.

Problem Statement

Students in secondary schools should be able to understand the various online data protection techniques that are used. To pursue higher education and secure prestigious positions in the security field, students will need to study and employ a variety of hashing and encrypting algorithms. Step visualization simulations of the majority of these algorithms are made available on our website, together with picture and step visualization encoders and decoders, as extra features, to help the general public educate and understand these algorithms. Users must comprehend robust encryption primitives in order to protect our systems, and visual cryptography can assist in resolving issues like data obfuscation, access control, and assaults on simpler ciphers. By making cipher simulators visible, we want to address the problem of data security.

Scope

The application was developed on a website platform; therefore, individuals who wish to use it can just visit the website on their browsers. The system is a website application which provides step visualization simulations for cryptography algorithms. The project is designed to create a website-based guidance tool that would help students explore and make decisions about various ciphers and which algorithms to choose. It provides information about encryption, decryption, and step visualization simulator for plaintexts, images and step visualizations. The tool acting as a simulator is easy to use. Students can simulate and perform encryptions which will develop their visualization and steps based on learned skills.

Motivation

Students today frequently struggle with the decision of whether to include security as one of their key domains. Since most students appear to struggle with understanding different ciphers, they never made the decision to investigate how security functions beyond the straightforward cipher exchange acknowledgement requests in SSL of the presentation layer [1]. Therefore, every IT student requires a simpler reference to understand ciphers, whether they are of different varieties in an easy manner. A step visualization simulation for illustrating a cipher working with all the components running over every input character is considerably better than 500 pages of method in theory mode from a published author because a picture truly is worth a thousand words. The target audience of this website is students who are in their second year of study in an IT or computer-related field, as most of them encounter this problem. In the modern era, understanding encryption techniques is crucial since malevolent individuals frequently search the internet for unencrypted data streams. Sensitive information may be exposed in the event of an attack, and if it isn't securely encrypted, government data, financial information, and other vital information may be leaked to foreign parties. Therefore, these encryption algorithms need to be studied, and they can be studied more effectively by aspiring engineers or IT students if they do so use a visualization tool that will improve their understanding of all the phases occurring in the code of certain algorithms [9]. An example of one of these tools is CrAlSim, which offers a simulator for crucial cryptographic methods.

CHAPTER 2
LITERATURE SURVEY

LITERATURE SURVEY

A few existing systems are studied and analyzed as follows:

dCode's Tools List [3]

Fig 2.1 dCode's Tools List

- Provides encryption and decryption for various transposition, substitution, poly alphabetic, etc ciphers.
- Provides a cipher identifier, where the user can input a cipher text, and give some clues to the system which when analyzed, provides a list of all the possible ciphers the cipher text may have passed through.
- It has tools for other categories like Acere cipher in Music, Trevanion Cipher in Steganography and various other puzzle solving modes.
- It has extensive tools and categories but limited use for just encoding and decoding and no simulation.

Crypto Corner [5]

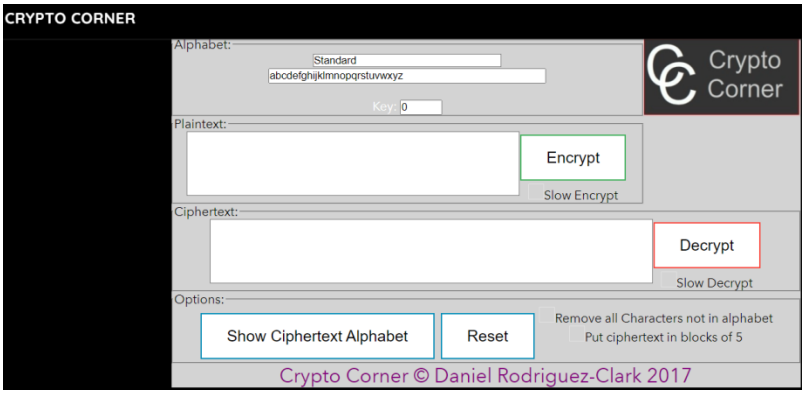


Fig 2.2 Crypto Corner

- Provides encryption and decryption for various transpositions, substitution, poly alphabetic, etc ciphers.
- Provides a cipher identifier, where the user can input a cipher text, and give some clues to the system which when analyzed, provides a list of all the possible ciphers the cipher text may have passed through.
- It has tools for other categories like Acere cipher in Music, Trevanion Cipher in Steganography and various other puzzle solving modes.
- It has extensive tools and categories but limited use for just encoding and decoding and no simulation.

Stylesuxx Steganography Online [6]

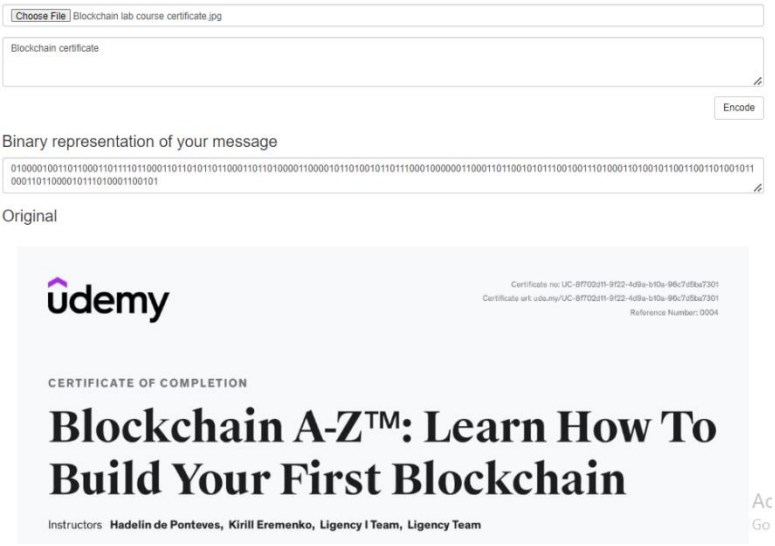


Fig 2.3 Stylesuxx Steganography Online

- Images can be encoded, and a binary representation is applied over the image which encodes the text message into the image.
- For decoding an image, the bin file or text file of the encoded image can be uploaded and original image and text can be decoded.
- It works well if the image is large and then more text or message can be encoded into the image.
- At no time will the disguised image or message be communicated over the internet; instead, everything takes place inside your browser.

Python Tutor [7]

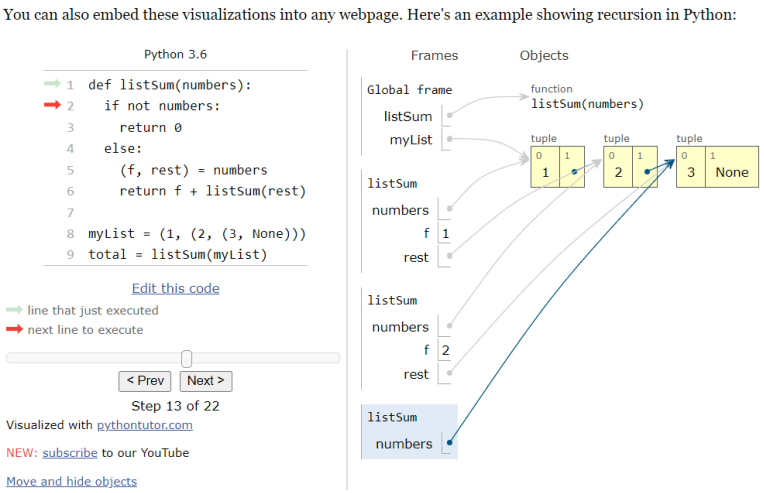


Fig 2.4 Python Tutor

- This site demonstrated how to embed visualizations of an execution of a code on a webpage.
- Previous and next buttons are used to change frames and objects of code based on the flow of execution and callback from previous lines.
- It works for simple recursive or a few subroutine functions only.
- It doesn't support framework like flask or toolkit like tkinter.

SourceTrail [8]

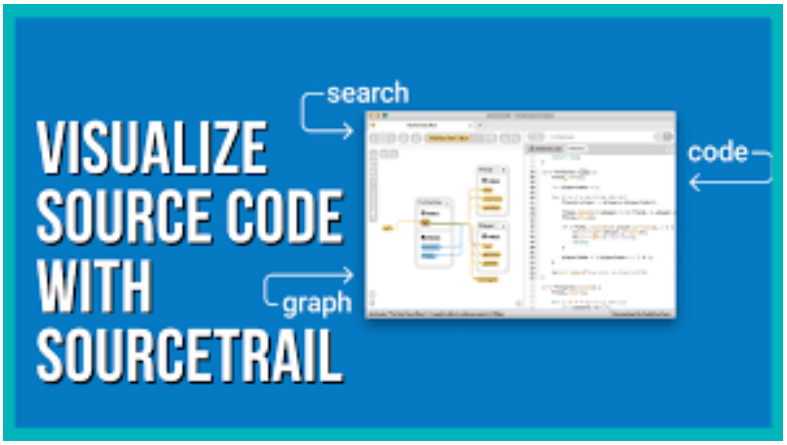


Fig 2.5 SourceTrail

- It is a cross-platform source code viewer.
- Software developers may quickly and thoroughly analyze and navigate unfamiliar source code with the help of this cross-platform developer tool, which combines interactive graph visualization, a condensed code view, and a powerful search algorithm.
- Automatically creates diagrams that depict dependencies, method calls, and class connections; demonstrates very clearly how each element interacts with other software elements.
- There isn't any language support beyond C++, JavaScript, and Python, but it should be available soon.

Graph Buddy [10]

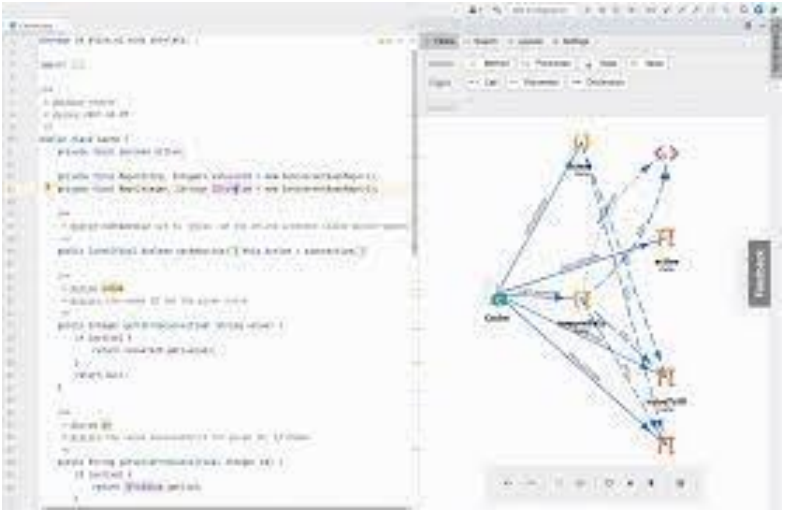


Fig 2.6 Graph Buddy

- It is a program designed to display code structure as 2D/3D graphs.
- The project's primary technologies are TypeScript, Scala, Neo4j (a graph database), React, and Vis.js (a library that helps in visualizations).
- The Graph Buddy panel's top menu allows you to filter components according to a specific kind. The nodes on the canvas can be organized using layouts.
- It provides interface buttons, graph canvas like highlighting, data positioning and flash messages.

Paper Name: Evolution of Visual Cryptography Basis Matrices with Binary Chromosomes [11]

- Based on the stacking of seemingly random shadow pictures, or "shares," visual cryptography is a type of secret sharing that only human vision can decrypt.
- The recovered image contrast and pixel expansion are the primary optimization metrics.
- Genetic algorithms are binary chromosomes that are encoded, and a proposed objective function is used to assess each chromosome's fitness value.
- Contrast-optimal visual cryptography techniques have been developed with the feature of producing the best outcomes at the lowest possible cost.

Paper Name: An AES cryptosystem for small scale network [2]

- Applications of wireless technology in tiny networks, such as reading sensors in wireless sensor networks or the UAV-C2 data link, in which data is protected by the wireless module with the cryptographic portion, has been utilized frequently in cryptographic processes.
- It is a scheduler which controls real time operations and manages tasks based on priority.
- Feature is that encryption efficiency was improved based on the fact that all messages were processed sans loss.
- Limitation is that this system works only for a small-scale network like Zigbee module.

CHAPTER 3
SYSTEM
DESIGN

SYSTEM DESIGN

Proposed solution

It becomes necessary to write functions based on stepwise treatment of the message data in order to emulate various cryptographic techniques. This happens when different functions are constructed in accordance with the code's flow, synchronized with each cipher generation, setting color divisions, and building loops with several steps and conditions. The ultimate outcome of the simulation is displayed by presenting the generated string, by choosing to compare the raw text, or in a different format depending on the user-selected cipher. The simulator [11] parses all the results of computations into a table that demonstrates the inner workings of each step, whether it be the input of letters or numbers into a formula or the rounding of a code.

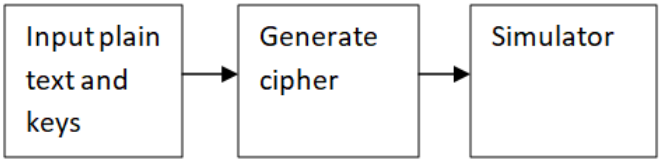


Fig 3.1 Block diagram of CrAlSim

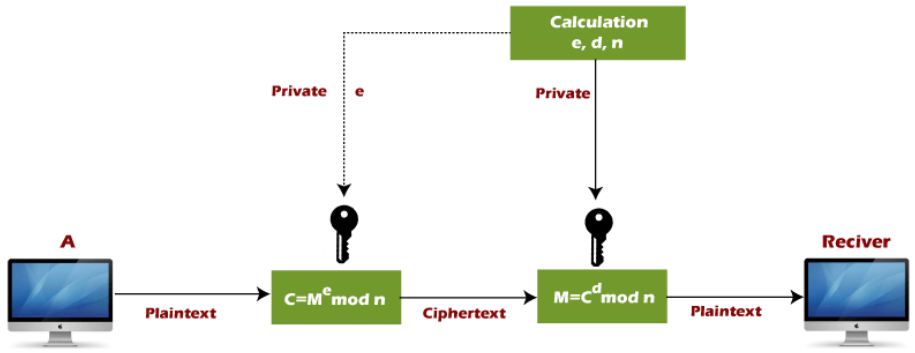
Detailed explanation about modules in Fig 3.1 is given below:

1. Enter plain text and keys: After choosing the encryption type, he requires, the user must enter his plain text message, which can either be in conventional format or add some basic punctuation along with numbers. The key or keys that need to be chosen are indicated in the note. The requirements for which kinds of keys can be chosen must be met, for example, for the Affine cipher, two keys need to be chosen and they must be co-prime, whereas a key for the AES block cipher can be 128, 192, or 256 bits long. The only permitted value selected for our project is 128 bits, or 16 characters.
2. Create a cipher: Here, we select the option for encryption. We may generate a cipher by clicking the "encrypt" button by entering the correct information for the plain text message and keys. As with the Affine cipher, the transformation of all alphabets is displayed in the following output, allowing the user to calculate how the original message and the encrypted keyword changed.
3. Simulator: It is made up of JS code that verifies all the requirements for encryption to start, then generates a graphical representation of the code's operation. Values are fetched into matrices or tables based on each step, and each new alphabet or number that is being worked on is highlighted by a color or style that denotes the current step in progression. Finally, final transformations are duly saved in an all-calculated-values table so the user can review the formula.

Methodology

Algorithms are visualized using straightforward functions that focus on events that may be represented graphically, allowing students or other users to see such events as moving pictures or color highlights and providing a better illustration of the various operational areas of the code. The choice of cipher determines which category of the algorithm needs to be seen in the first step. Monoalphabetic ciphers like Affine are helpful for single word strings since their encryption may be based on just a specific length, while AES (Advanced Encryption Standard) cipher can be used for a block of letters. The user must then submit the correct sort of plain text message because different ciphers have requirements based on the length of the string or the standard language being used. The key is then chosen or created in accordance with the requirements, as RSA requires extremely high values for proper security. To create encrypted text or decode plain communication, utilize the encryption and decryption buttons. The specific directions for the operation, such as which type of key to choose or what conversions are taking place, must be taken note of, after which utilizing the start, play, pause, or reset buttons to manage the simulation's progression.

For asymmetric ciphers like RSA, which follow public key encryption method, visualization is based more on suggestions [12]. Suggesting all or minimal required possible values for private key and public key pairs, especially the numbers E and D, which are calculated beforehand from L and N, which in turn are found after doing calculations on prime numbers P and Q.



RSA

CHAPTER 4
ANALYSIS

ANALYSIS

Process model

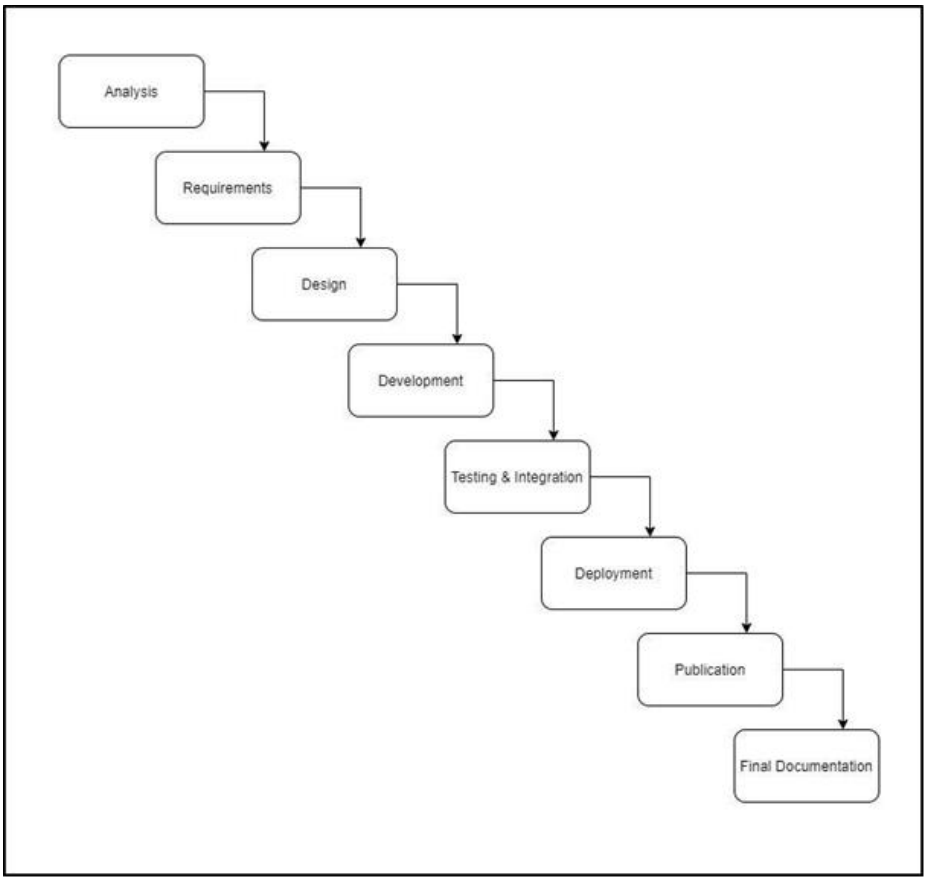


Fig 4.1: Waterfall Model

The Waterfall Model will be used in the creation of our project. The following justifies the use of this model in our project:

- When using the waterfall method, each phase is developed after the one before it is finished. This makes the waterfall model's phases highly precise and well defined. The waterfall model got its name because the phases flow from a higher level to a lower level, much like a cascade.

- The Waterfall Model is the most appropriate for this project because the requirements are steady and don't change too often.

- It makes departmentalization and monitoring possible. A product can move step-by-step

through the phases of the development process model by having a timeline set up with deadlines for each development stage.

- The waterfall model proceeds through stages that are simple to comprehend and explain, making it simple to apply.

- The rigor of the approach, with each phase having clear deliverables and a review process, makes it simple to manage.

- The phases in this model are processed and finished one at a time sans overlapping.

Feasibility Analysis

1. **Technical feasibility**: Technical viability focuses on the organization's technical resources (hardware and software) and helps assess whether the technical team can turn concepts into functional solutions. Most of the software needed for our project is open-source and easily accessible (such as JavaScript, HTML, and CSS)

2. **Economic feasibility**: An analysis of the project's costs and benefits often forms part of this evaluation. Since the majority of the essential software is open source, the project will be built at a low cost. The main expenses will be for the computer system and the servers used to host the application.
3. **Legal feasibility**: This evaluation looks at any potential legal infractions of the planned project, such as zoning laws, data protection laws, or social media laws. Since all permits and laws will be adhered to and incorporated into the project, there are no legal difficulties with the project.

4. **Operational feasibility**: This assessment entails conducting research to evaluate whether—and how effectively—the needs of the organization can be satisfied by completing the project. This project's major goal is to improve the visualization of encryption methods and make it accessible to people so they can better comprehend cryptography.

Timeline Chart

Table 4.1: Timeline Chart

Task Name	Start Date	End Date	Duration (Days)
Analysis	01-Mar-22	30-Apr-22	60
Design	01-May-22	30-Jun-22	60
Development: Cipher algorithms	01-Jul-22	15-Aug-22	45
Development: Algorithms input page	16-Aug-22	15-Oct-22	60
Development: Simulation page	16-Oct-22	30-Nov-22	45
Development: Theory page	01-Dec-22	15-Jan-23	45
Testing & Integration	16-Jan-23	31-Jan-23	15
Publication	01-Feb-23	28-Feb-23	27
Final Documentation	01-Mar-23	31-Mar-23	30

Gantt Chart

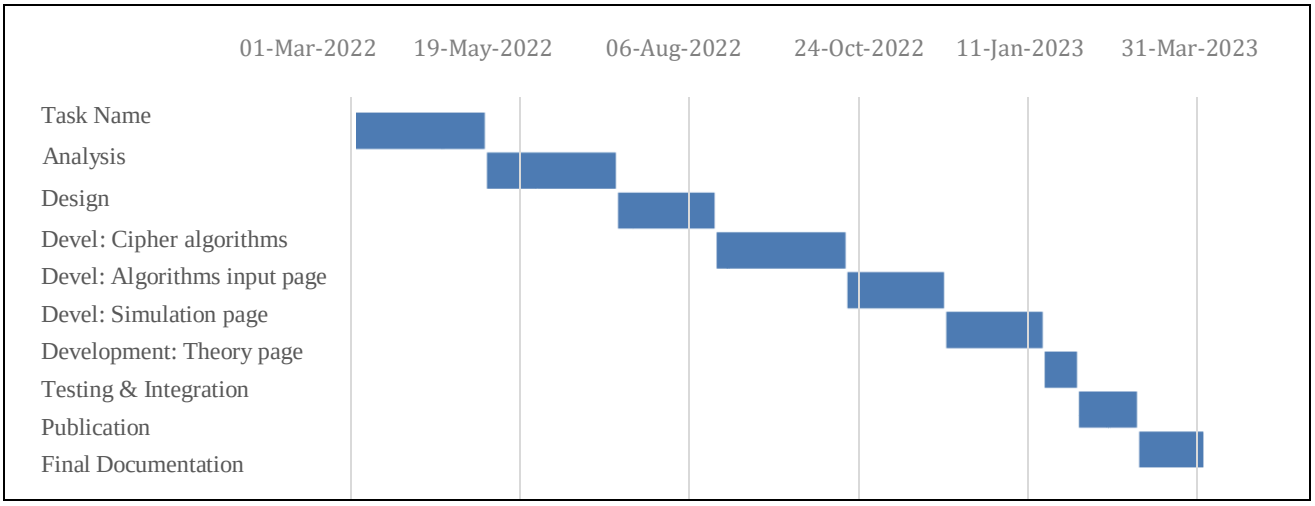


Fig 4.2: Gantt chart

CHAPTER 5
DESIGN

DESIGN

UML Diagrams

Class Diagram

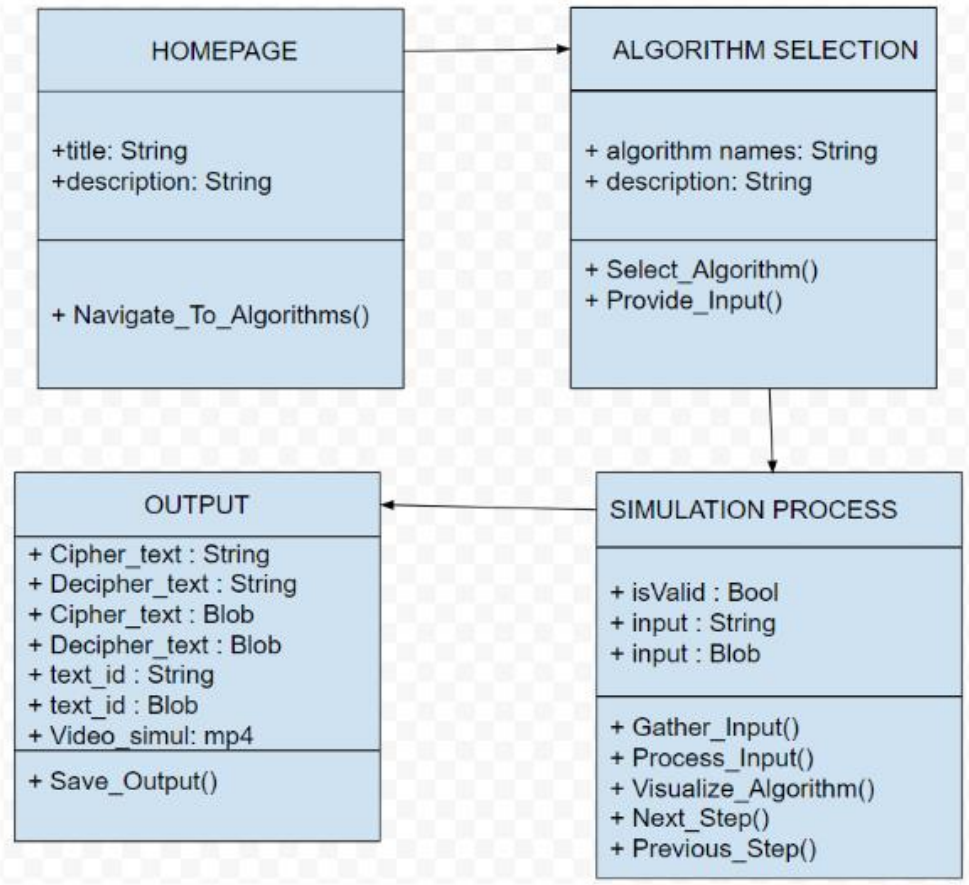


Fig 5.1: Class Diagram

Use Case Diagram

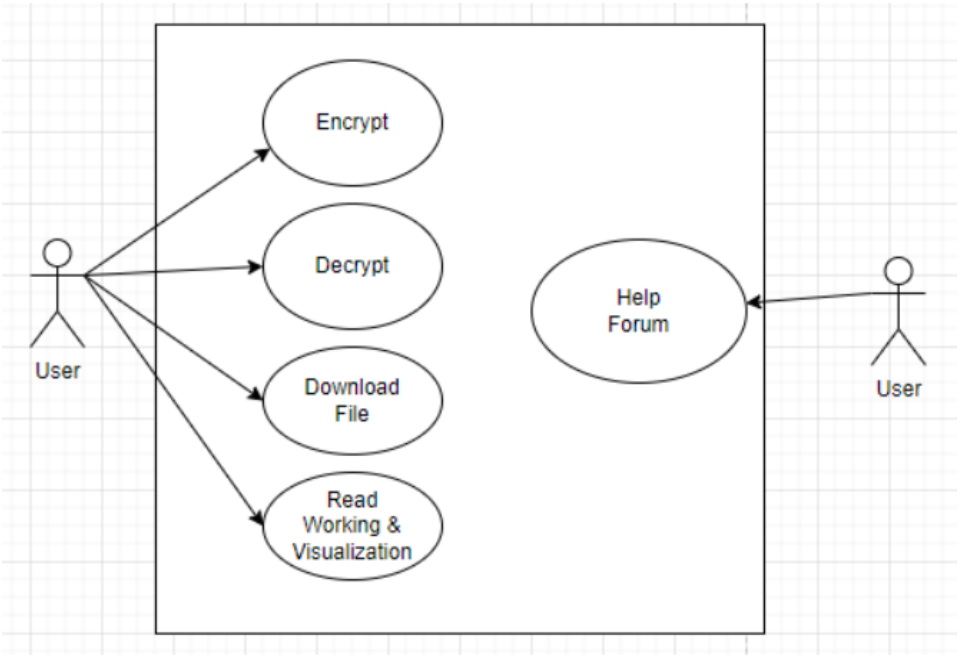


Fig 5.2: Use Case Diagram

Activity Diagram

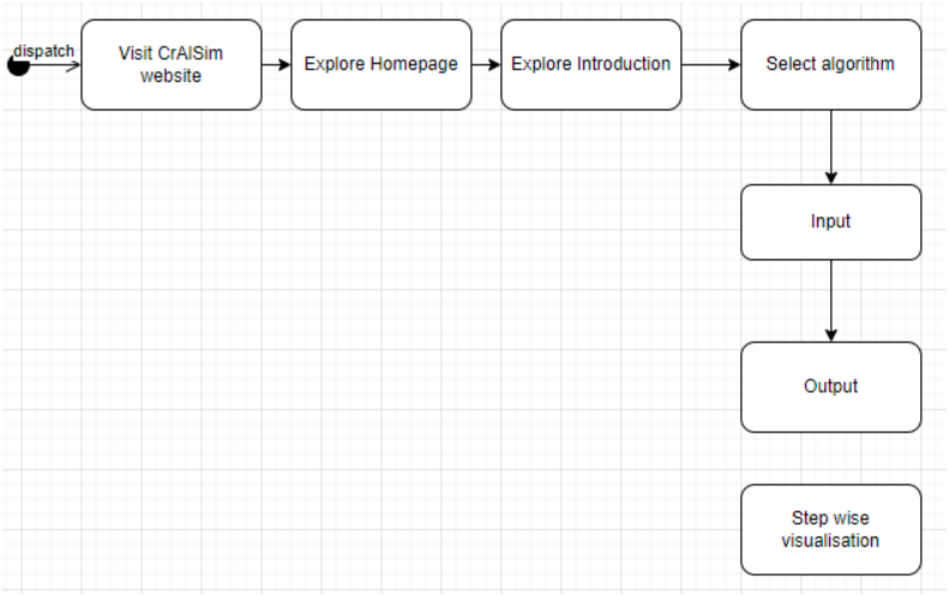


Fig 5.3 Activity Diagram

Sequence Diagram

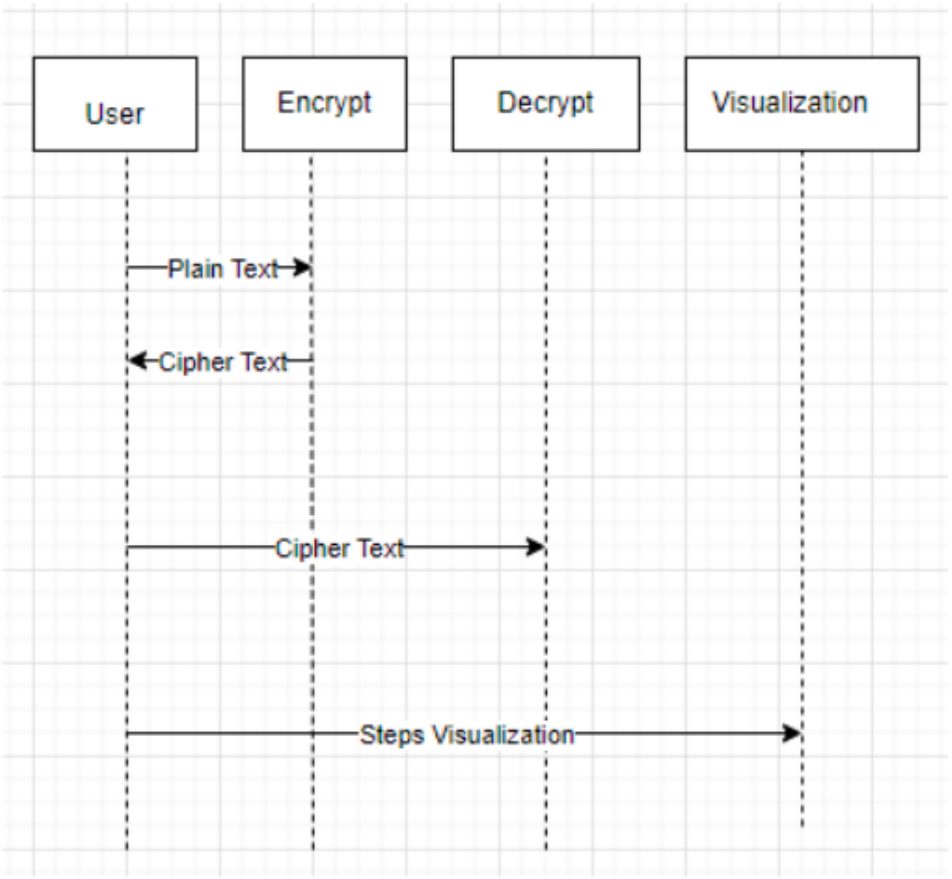


Fig 5.4: Sequence Diagram

Hardware and Software Requirements

Software Requirements:

Table 5.1: Technology Stack

Modules	Technology
Frontend (Cross platform application)	JavaScript, HTML, CSS
Toolkit	Android development tools
Affine monoalphabetic cipher algorithm AES block cipher algorithm RSA asymmetric algorithm	JavaScript and Data Structures.
IDE	Visual Studio Code

Hardware Requirements:

- Computer
    1. Core i3 Processor (Minimum)
    2. 4 GB RAM (Minimum)
    3. SSD
    4. HD Display
- Keyboard

Chapter 6
Implementation

Implementation

The encryption process begins as soon as the first module is run. The user navigates to our website's homepage and chooses the cryptographic method they wish to work with, whether it is an affine cipher of the monoalphabetic type, AES, Advanced Encryption Standard or DES, Data Encryption Standard [14] from the category of block ciphers, or RSA [13] from the category of asymmetric ciphers. The user then types his plain text message, choosing the appropriate key after each set of letters, punctuation, or numerals. The inverse requirement isn't met for affine ciphers since alternative combinations result in two alphabets that have the same character as ciphers, therefore the key A and B should be co-prime of modulo m, or else the GCD of the key and modulo m should be 1. When all the conditions are met, the user can finally click the encryption button to create the cipher.

The simulator operates on simulation [4], or graphical visualization of the internal workings of the algorithms, in the second module, where a graphical user interface (GUI) is rendered. Internal cipher code function operates in synchronization with the highlighting of changes in the code that relate to plain text messages. Consequently, value changes take place, which are displayed in tables, matrices, or by moving the values of objects. Each cycle of the function is shown for each block of code or one character, and set controls, such as the play, pause, skip and start buttons, are provided to let the reader decide whether to follow the story through to the end or just focus on one transformation. The internal workings of all calculations are displayed in a table, alternatively, similar to the AES simulation, functionality is provided by the All Value button where all XOR and multiplication operations are taken into account for each round. The final output result is transformed back into a string since calculations for hexadecimal numbers take place inside the function. As a result, module one returns the generated cipher to the user [15].

Flow chart for CrAlSim system is shown below:

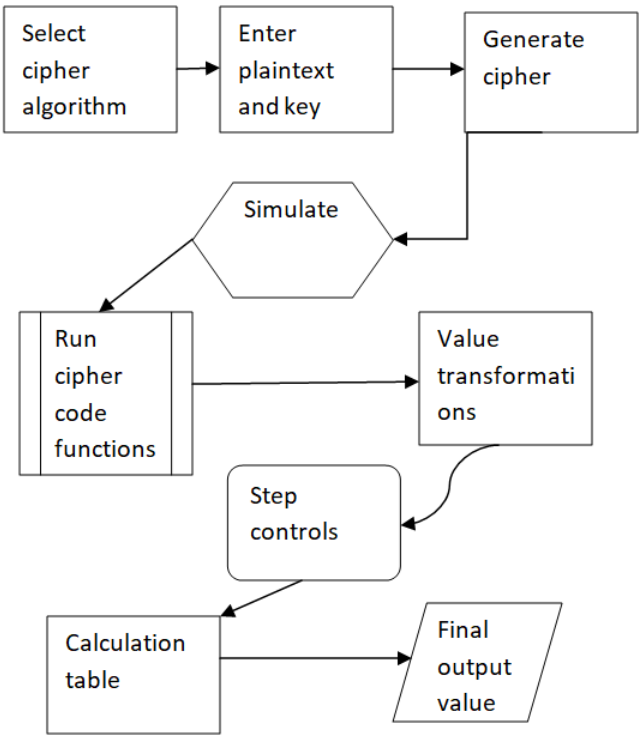


Fig 6.1: Flow chart for working of CrAlSim

As seen in Fig. 6.1, CrAlSim is made up of two modules: one that depicts the fundamentals of encryption and another that actually simulates how an algorithm function internally.

The user must first read the landing page and comprehend the need for cipher simulation before the simulation procedure can begin. The user then navigates to the menu bar's Algorithms tab. He can then choose whatever type of cipher he wishes to practice on from that list. After choosing a cipher, the user can access the create cipher page, which offers the ability to encrypt and decrypt plain text communications using valid keys and input strings that follow a specified pattern. He can view comments or alerts if the system discovered any input field errors after selecting the create cipher option. In this iteration of the model, modeling of the decryption process is not possible, although simulation of the encryption process can be observed on the Simulator page.

Simulator cycles through the ciphers many code functions, and it has different tables and color highlights in place to show the current data item in use. Every major step of the algorithm is covered, including the application of the formula (ax+b) %m to each character of plain text in Affine, each event of adding keys, XORing, multiplying, shifting, and substituting arrays in AES, and the generation of numerous large prime numbers for the private key of RSA. Each process is visualized at the designated time. The approach focuses on value transformations occurring at the current stage, and buttons like skip steps, next, and if available, play and pause are utilized to control the next function display or transmutation of an alphabet. The All Value button, which records each computed value of an object or array in a table and displays all outputs from the first to the last round of an event, is available to keep track of all calculations. If the final output result was originally in another format at an intermediate phase, it is subsequently transformed back to string and given to the user.

Algorithms used:

Affine

An example of a monoalphabetic substitution cipher is the Affine cipher, which maps each letter of the alphabet to its numerical counterpart, encrypts it using a straightforward mathematical formula, and then converts it back to the original letter. The cipher is essentially a regular substitution cipher with a rule dictating which letter goes to which because of the formula utilized, which encrypts each letter to one other letter and back again.
Working modulo m (the length of the alphabet employed) is essential to the entire process. The letters of an alphabet of size m are first translated into integers in the range of 0 to m-1 in the affine cipher. Two numbers—let's call them ‘a’ and ‘b’—make up the 'key' for the Affine cipher [18]. The usage of a 26-character alphabet is assumed in the discussion that follows (m = 26). The choice of ‘a’ should be made so that it is comparatively prime to m (that is, ‘a’ shouldn't share any variables with m).

AES

A symmetric block cipher algorithm with a block/chunk size of 128 bits is the AES Encryption algorithm, also referred to as the Rijndael algorithm. These distinct blocks are converted using keys that are 128, 192, and 256 bits long. It then connects these blocks to create the cipher text after encrypting each one separately.
It is founded on an SP network, also referred to as a substitution-permutation network. It is made up of several interconnected operations, such as permutations and substitutions, which involve rearranging bits to create different combinations of inputs and outputs.
An AES feature is [17]:
SP Network: Unlike the DES algorithm, it operates on an SP network structure rather than a Feistel cipher structure.
Key Expansion: In the initial stage, just one key is used; afterwards, numerous keys are employed in separate rounds.
Byte Data: The AES encryption technique operates on bytes rather than bits when processing data. Thus, throughout the encryption process, the 128-bit block size is treated as 16 bytes.
Key Length: How many rounds are required depends on how long the key is that is being used to encrypt the data. There are 10 rounds for a 128-bit key size, twelve rounds for a 192-bit key size, and fourteen rounds for a 256-bit key size.

RSA

Asymmetric cryptography uses the RSA algorithm. Asymmetric really implies that it utilizes both the public and private keys, which are two separate keys. As implied by the name, the private key is kept secret while the public key is distributed to everyone.

Asymmetric cryptography illustration:

1. When requesting data from the server, a client (for instance, a browser) transmits the server its public key.
2. Using the client's public key, the server encrypts the data before sending it.
3. This data is delivered to the client, who decrypts it.
4. Since the encryption is asymmetric, even if a third party obtains the browser's public key, they are unable to decrypt the data. Only the browser can do it.

Test Cases

Table 6.1 Test Cases

Test Case ID	Test Case Description	Test Data	Expected Result	Actual Result	Status
T01	Enter valid i/p for ‘a’ and ‘b’	a=5, b=3, message: swapnil	pjdaqrg	pjdaqrg	Pass
T02	Enter invalid i/p to check inverse function	a=4, b=2, message: swapnil	wmckciu	Invalid i/p alert	Fail
T03	Enter standard key of length 5	key: zoquf, plain text: swapnil	rkqjshz	rkqjshz	Pass
T04	Enter random key of different length than 5	key: erosion, plain text: swapnil	wnohvwy	Error as key length is 7≠5 (pre defined)	Fail
T05	Enter i/p and key in standard plain text	i/p: swapnil, key: mynameismyna meis (128 bits)	52u2fcgcC TfXYYcz a5mW5g==	52u2fcgcC TfXYYcz a5mW5g==	Pass
T06	Enter same i/p but with different orientation of pi	i/p: swapΠ, i/p: swapπ	35n1aHw m/pFIUR To5pPFV g==	46bLBFD ZCFZOXJ 7za4bk/w ==	Fail

Test cases 01 and 02 are examples of Affine encryption algorithm which belongs to the category of monoalphabetic cipher.
Test case 01 passed as both the key values are co-prime to each other as conditions stated and plain message is in standard format too.
Hence expected and actual results are the same.
Test case 02 failed as both the keys have a common factor 2, i.e., their GCD isn’t 1 and hence inverse of transformed alphabets won’t be ingenious.
The cipher fails here as inverse of ‘c’ has same inputs ‘a’ and ‘n’.
Hence the test case is rejected.
An alert message is provided by our tool if user enters any incorrect values.

Test cases 03 and 04 are examples of Vigenère encryption algorithm which belongs to the category of polyalphabetic cipher.
Test case 03 passed as length of the key was the same as entered by the user beforehand.
Hence expected and actual results are the same.
Test case 04 failed as the length of the key ‘erosion’ is not the same as mentioned beforehand (5).
This leads to a formation of a different keyword and hence the corresponding tabula rectum formed

becomes incorrect.
Hence the test case is rejected.
An alert message is provided by our tool for user to either enter correct keyword or correct key length.

Test case 05 and 06 are examples of AES encryption algorithm which belongs to the category of block cipher.
Test case 05 had the correct input stream text and length of key bits, hence its output generated is same and it passed.
Test case 06 was checked against two different inputs having same actual meaning but change in symbol orientation for pi (Π, π).
Here expected and actual results based on meaning of the input text should be same, but the system couldn't process different symbols and yielded dissimilar results.
Hence system discrepancy caused the test case to be rejected.

CHAPTER 7

RESULTS

RESULT

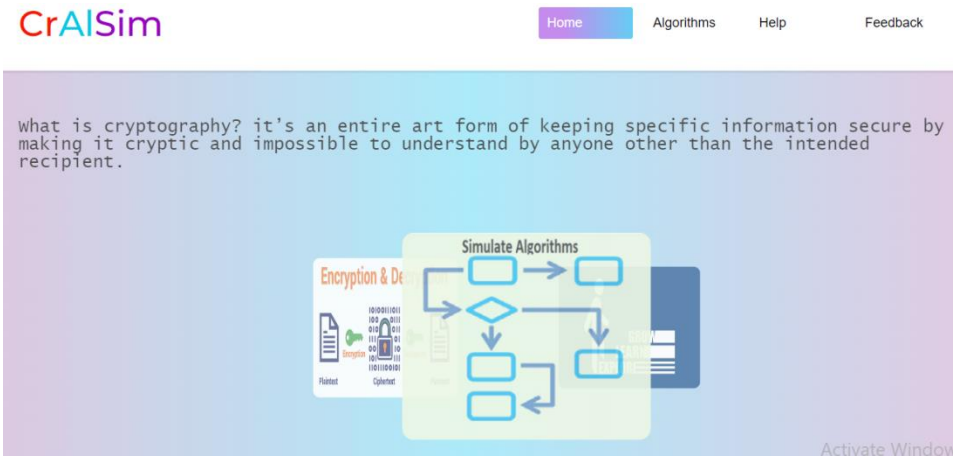


Fig 7.1 Home Page

This is the landing page of the CrAlSim website.
Simulation flow process and encryption, decryption photos are moving in a conical manner just to encapsulate the feel of animations which are going to be used in the algorithms.

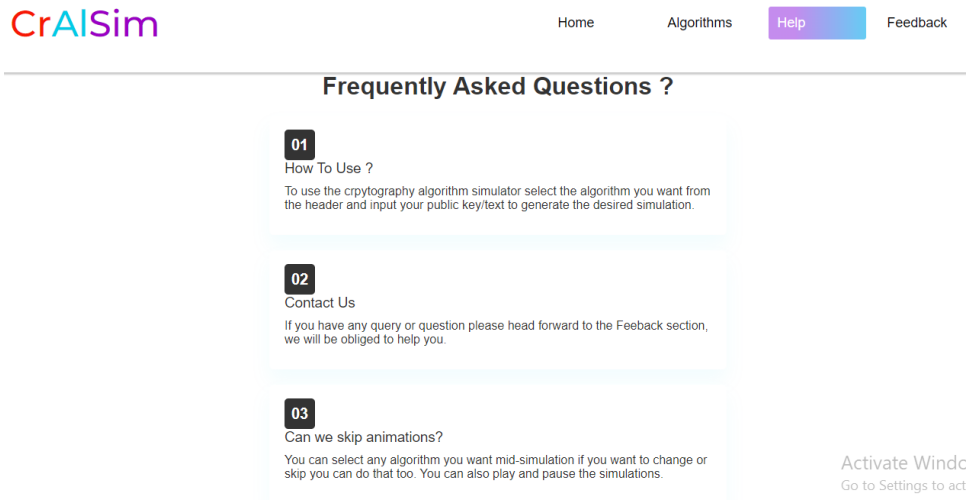


Fig 7.2 Help Page

This is the HELP page of the CrAlSim website.
FAQs are mentioned which give the basic idea of how to use the tool, which is simply selecting algorithms and performing encryption and simulation tasks.
Users can contact us if they have any query by going to the Feedback section.
Skipping steps is a good feature of our website, and that's mentioned for users to know.

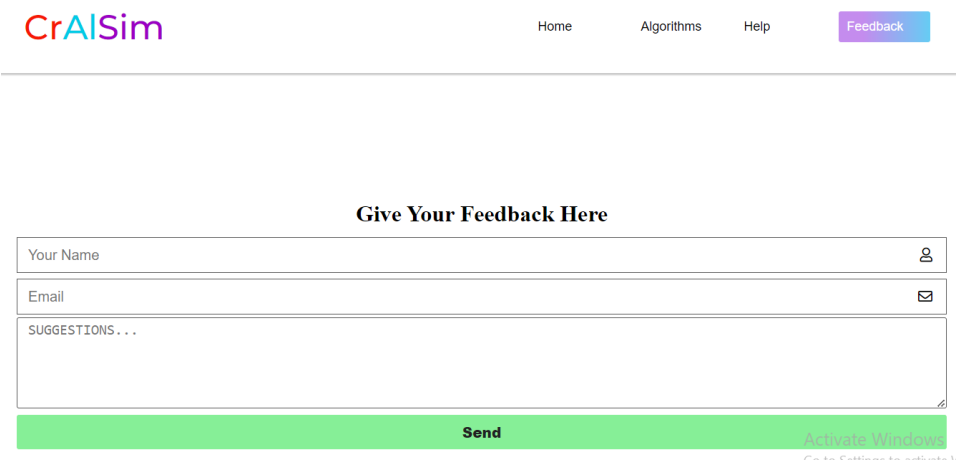


Fig 7.3 Feedback Page

Simply by putting in name, email and suggestions, users can give us their feedback.
We revert to their issues with solutions at the earliest.

Affine:

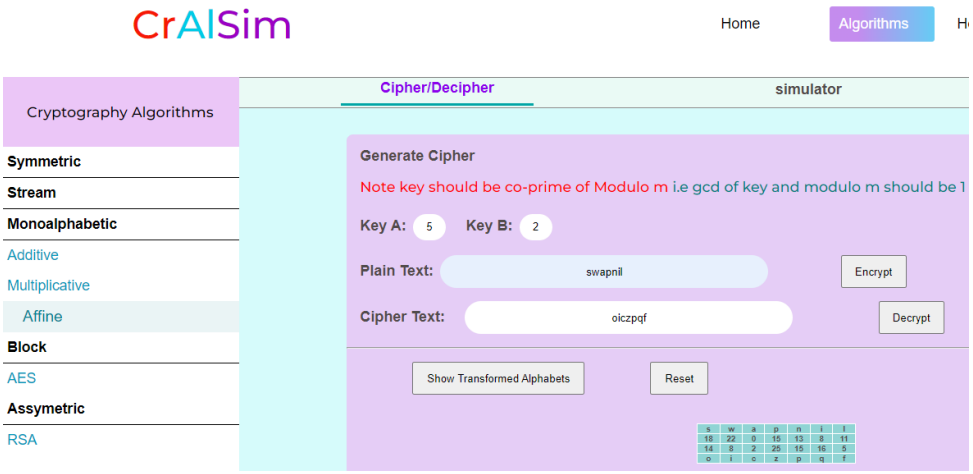


Fig 7.4 Affine Main Page

Key A is 5 and key B is given 2 as input, they satisfy the condition of being co-prime.
Plain text is provided, and cipher is generated, and all transformed alphabets are represented in correspondence to their numeric values with letter ‘a’ starting from 0 and ‘z’ ending at 25.
In case of the keys having a common factor, a note is mentioned for users to read.
Reset button sets values of all keys and plain text to null.

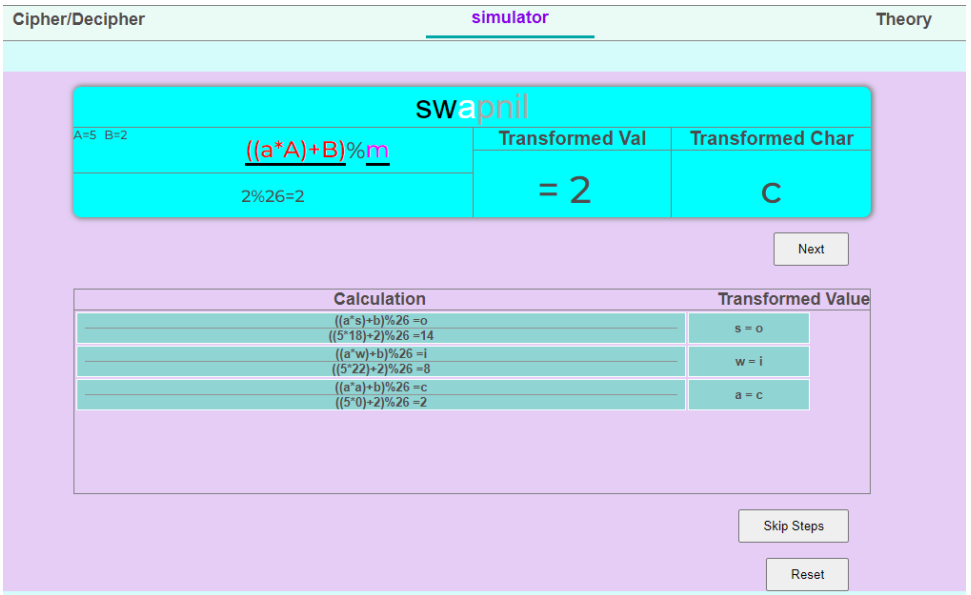


Fig 7.5 Affine Simulator

Each letter in the plain text 'swapnil' is fed into the formula (ax+b) %m, and further calculations are displayed each time the Next button is pressed.
The calculations for the letters, ‘s’ and ‘w’ have been completed, so they are highlighted in black. The letter 'a' is now undergoing conversion, so it is displayed in the color white.
The remaining characters will be displayed in gray.
Each derivation and its resulting value, along with the computations and modifications they imply, are displayed in the transformation table.

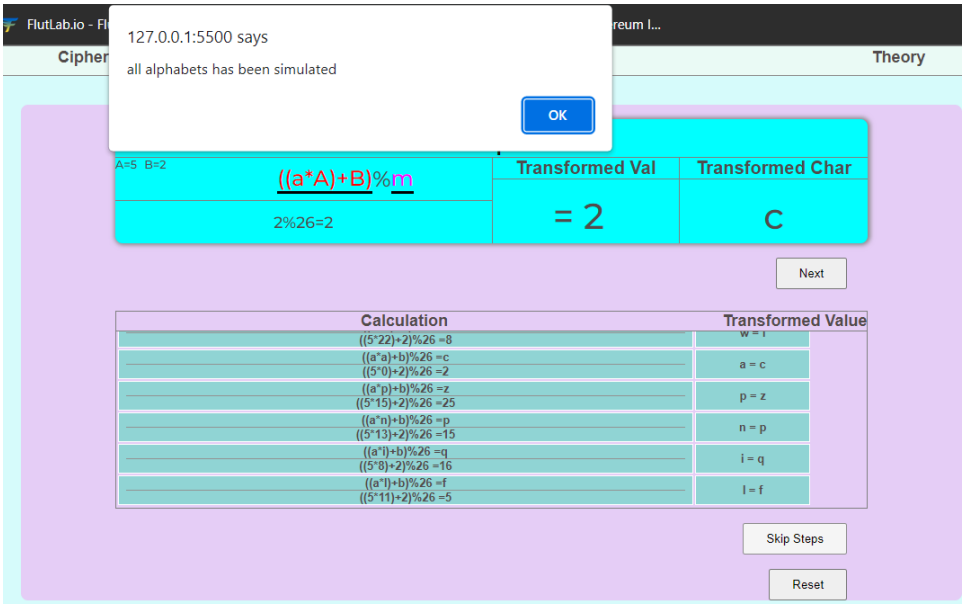


Fig 7.6 Affine Skip steps

Affine cipher is used to generate the entire output if the procedure is understood for just one letter because it can be difficult to go through each letter's transformation step by step.
Hence skip steps functionality is aided to provide the final calculation table, and it also pops up with an alert message that simulation of every alphabet is done if the user persists with clicking on Next or Skip buttons.

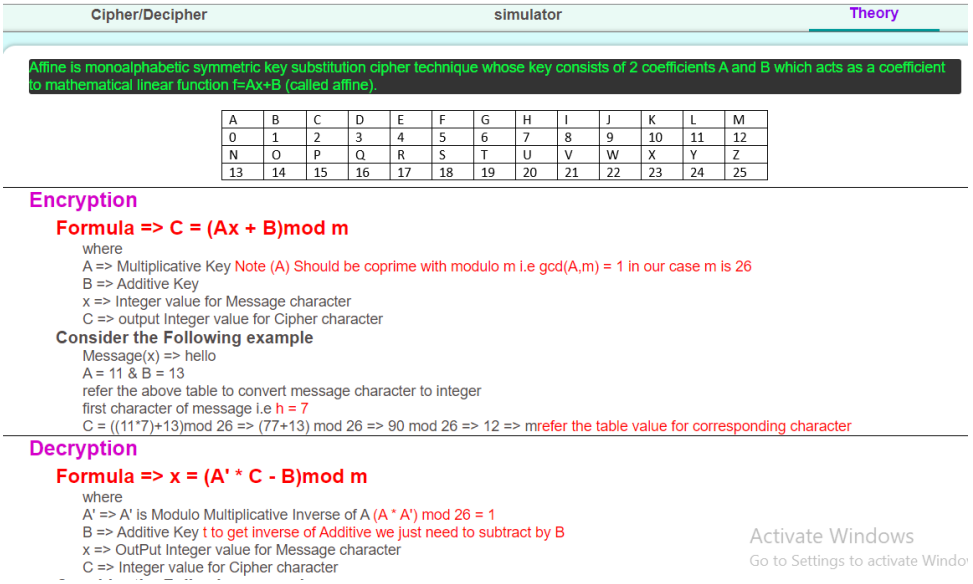


Fig 7.7 Affine theory page

Theory page of Affine cipher gives a side view of all the notations used in the formula for cipher generation and plain text decryption.
Also, a 'hello' message is supplemented to aid basic users in understanding the simulation flow.
They can use the same solved example as a reference by using it in the simulator and tally with the explained theory to comprehend the working of the cipher better.

AES:

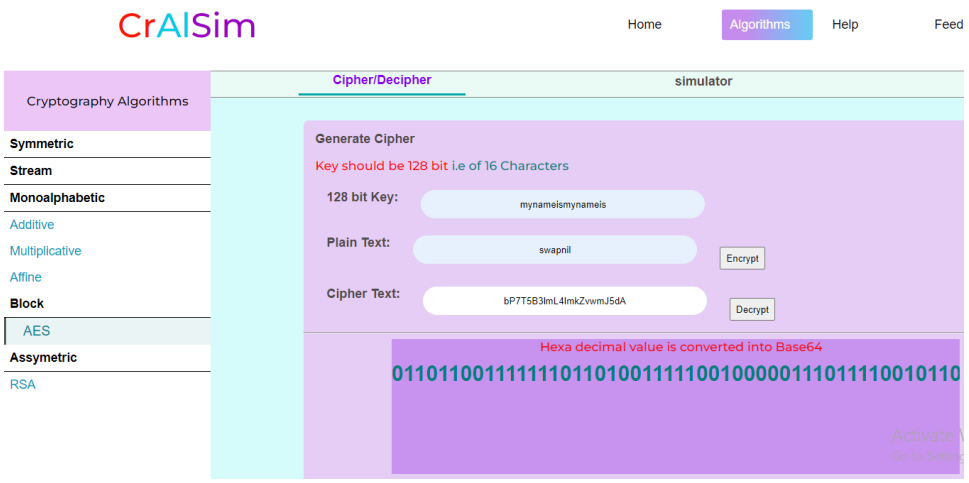


Fig 7.8 AES main page

AES is a block cipher, with 128 bits i.e., 16 characters chosen as default key length for the tool we've

designed.
Encryption and decryption functions can be used on the generate cipher page itself, with hexadecimal to base64 conversions being shown.
The cipher text produced after encryption for the plain text 'swapnil' and the 128 bits key 'mynameismynameis' was 'bP7T5B3lmL4ImkZvwmJ5dA'.

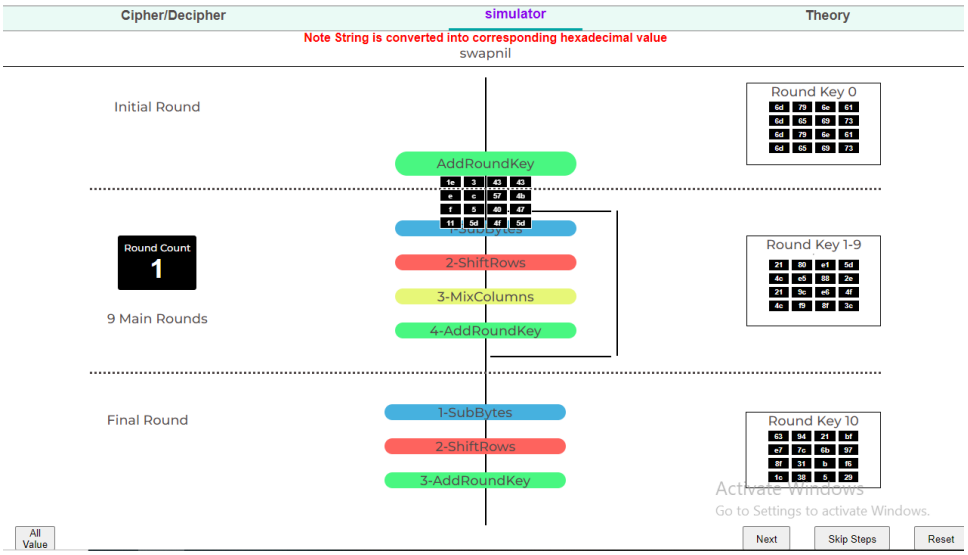


Fig 7.9 AES initial round

This is the AES simulator tab [16].
Only the XOR operation is carried out between the plain text converted into hexadecimal format and placed in a 4*4 array in the first round, along with the event round key word [0..3], and the output is calculated to the repetitive steps later in round 1 (just like a matrix is created in hill cipher where each letter is represented by a number modulo 26).

Events:

The original round array encounters many of the events and changes.
The S-box, also known as a lookup table, is used to implement substitution in the SubBytes event.
The following event, MixColumns, does matrix multiplication, and the final event, Add Round Keys, shifts the rows of the incoming 4*4 array to the left a set number of times.
The previous stage's resultant output is XORed with the appropriate round key.

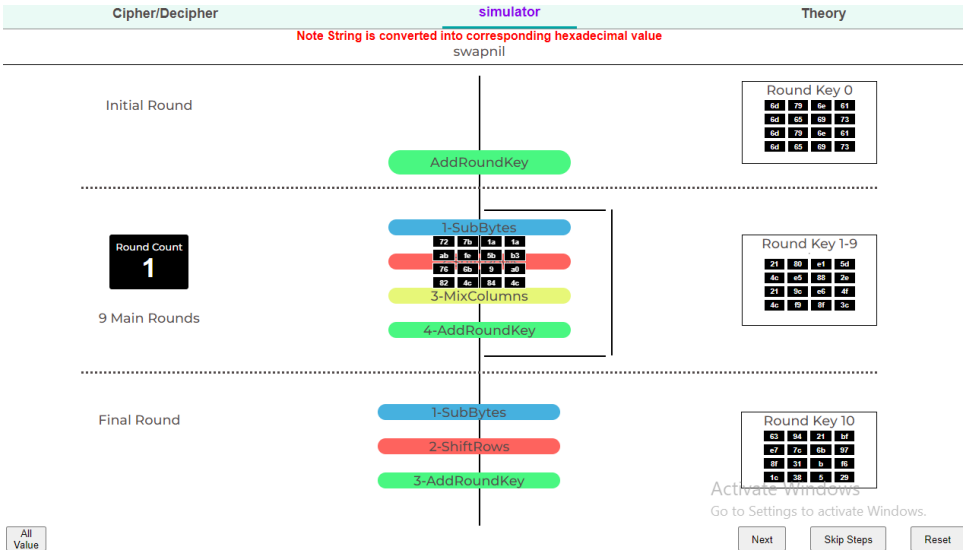


Fig 7.10 SubByte

In this event, the state array's bytes are split into two equal portions and converted to hexadecimal. These components- rows and columns are mapped using an S-Box substitution box to create new values for the final state array.

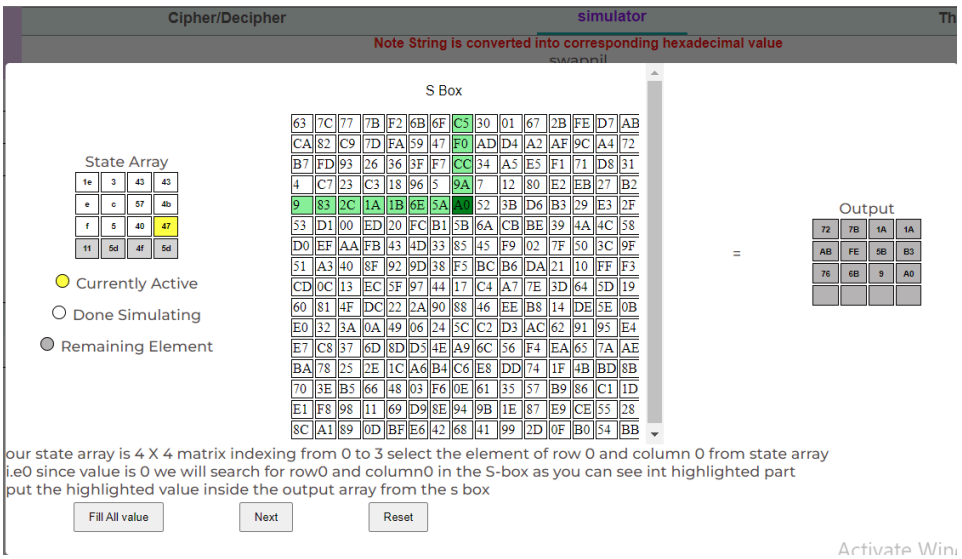


Fig 7.11 SubByte inner calculation

After clicking on the sub byte event this page opens.
How the resultant 4*4 array is created is shown in this step of visualization.
Numbers are chosen based on the mapping of row and column from the S Box already available.
New array is created and passed to the next event.

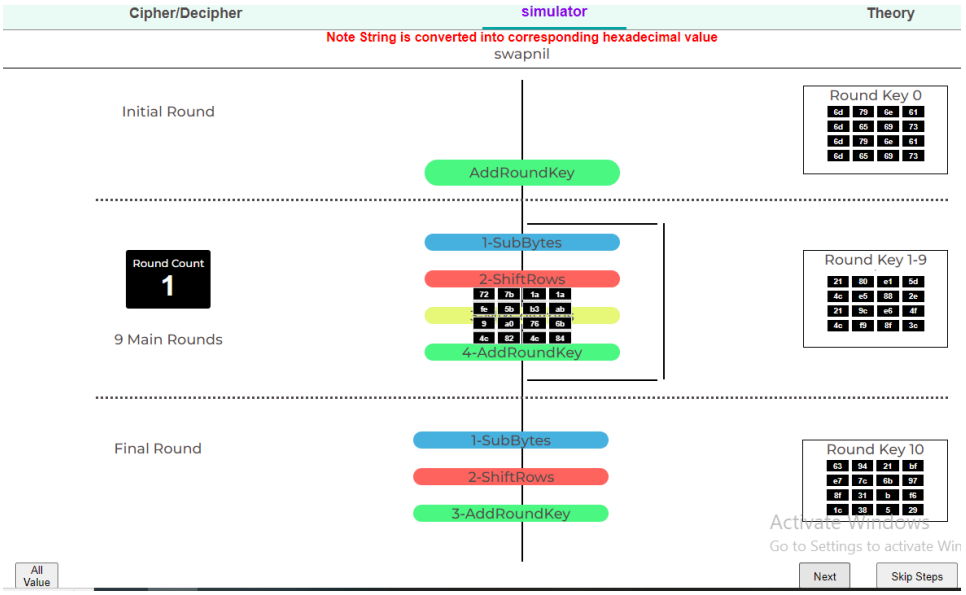


Fig 7.12 Shift rows

This event switches the row items around.
There is a first row skip.
The items in the second row are moved one position to the left.
Additionally, it moves the third row's elements two successive spaces to the left and moves the last row three spaces to the left.

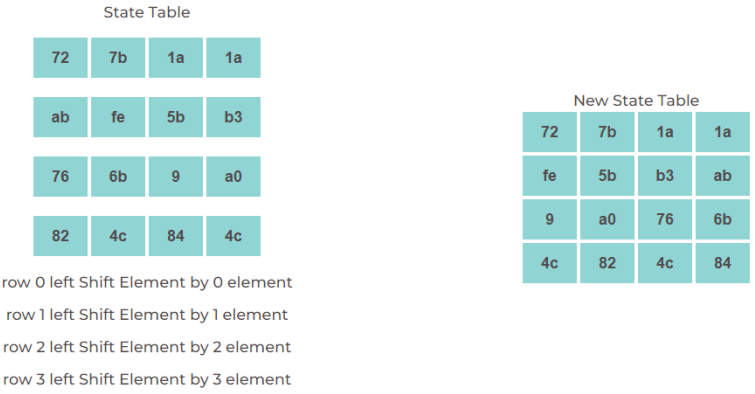


Fig 7.13 Shift rows inner calculation

Row elements are shifted to the left by the number of times in the row number they appear in, as stated quite clearly on the page that comes when clicking on the shift rows event.

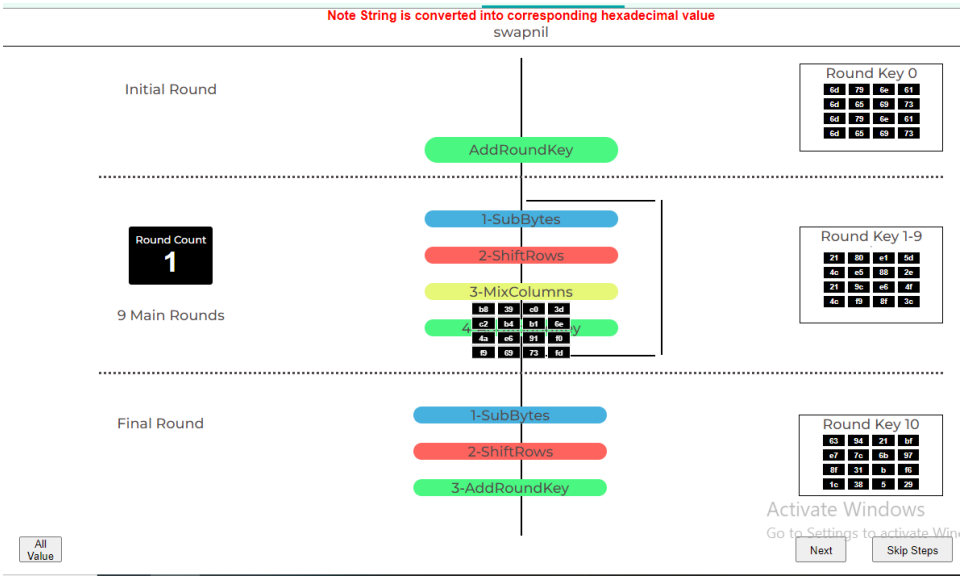


Fig 7.14 MixColumns

To create a new column for the following state array, it multiplies a constant matrix by each column in the state array.
Your state array for the following step is obtained once all the columns have been multiplied by the same constant matrix.
This specific action should not be carried out in the final round.

:

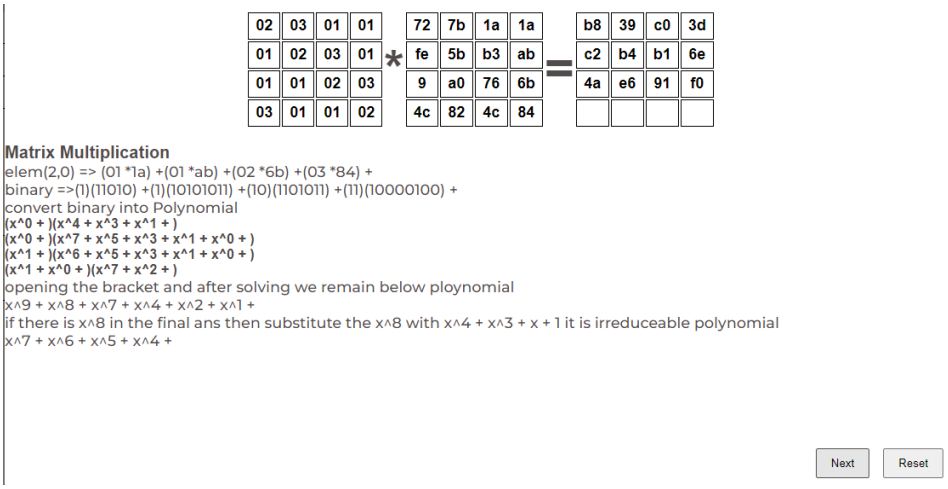


Fig 7.15 Mix column inner working

Matrix multiplication is shown with the live example of current state of array entry.

Add Round Key event:
After obtaining the state array in the previous phase, the appropriate round key is XORed with it. The resultant state array becomes the ciphertext for the particular block if this is the final round; otherwise, it serves as the new state array input for the following round.

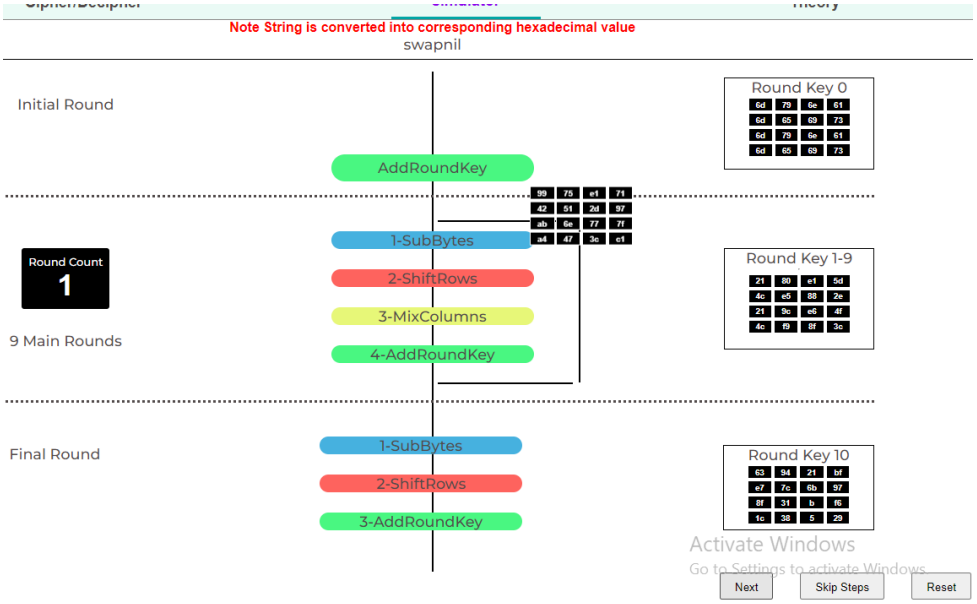


Fig 7.16 Repetition of rounds

The output image demonstrates how the array of add round key event returned as input to SubBytes after nine iterations of the aforementioned events that are repeated again in a cyclic loop.
The number of active rounds is kept track of by a counter.

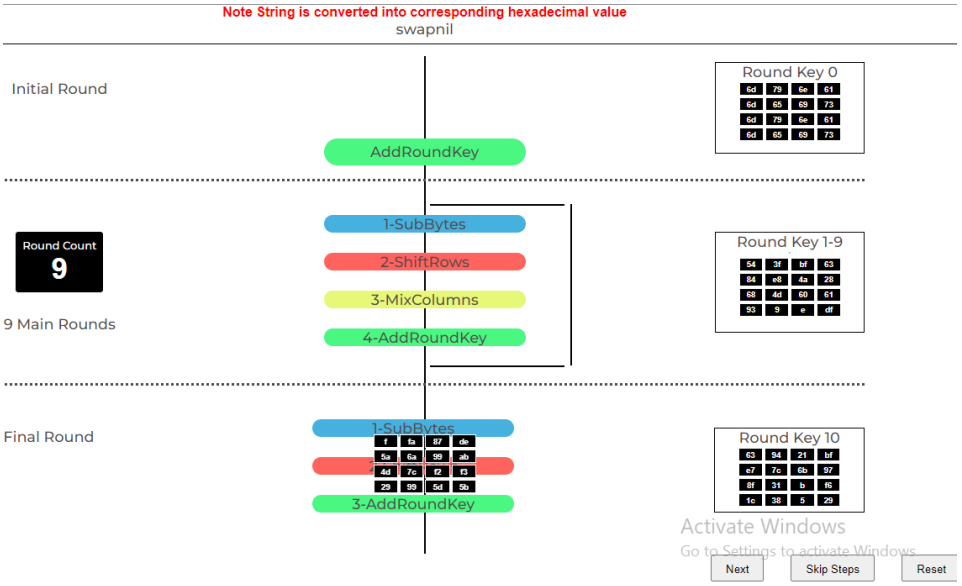


Fig 7.17 Final round

The final round output is produced following all in-between process modifications, and the result depicts how the array flows through the process without a MixColumns event.

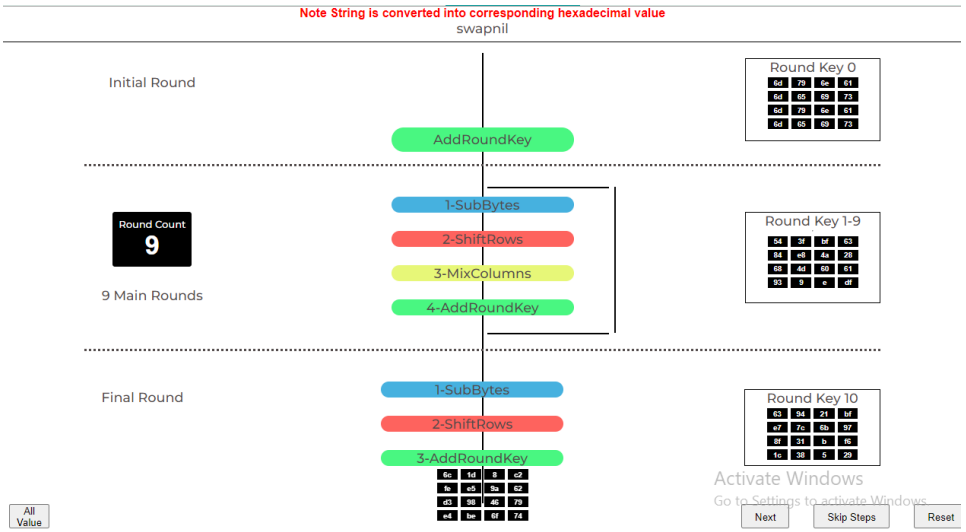


Fig 7.18 Final o/p

Final output matrix produced is our resultant encrypted hexadecimal value 4*4 array by AES. It is then internally converted into a string and displayed on the cipher generated page.

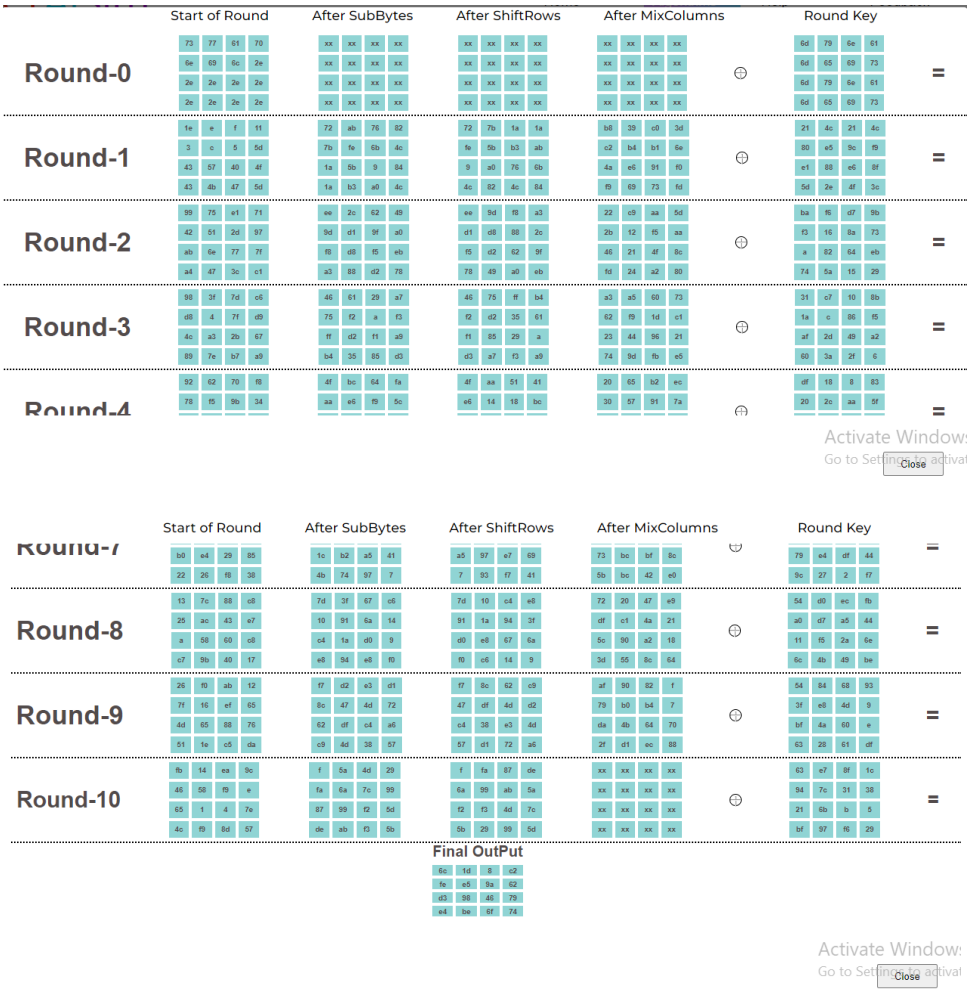


Fig 7.19 Calculation table for AES

Calculation table can be viewed by clicking on the All Value button.
Keeping track of all matrix transpositions is tedious, so we've created a calculation table to note the resultant matrix in each step of every event of every round.

RSA:

It is challenging to factorize a huge integer, which is the foundation for RSA.
The public key comprises of two numbers, one of which is the result of multiplying two significant prime numbers.
Additionally, the private key is created using the same two prime numbers.
The private key is therefore compromised if someone is able to factor the huge integer.
As a result, the key size completely determines how strong an encryption is, and if we double or treble the key size, the encryption strength improves dramatically.
RSA keys can normally be 1024 or 2048 bits long, but experts think that 1024-bit keys may soon be cracked.
However, it appears to be an impossible feat at this time.

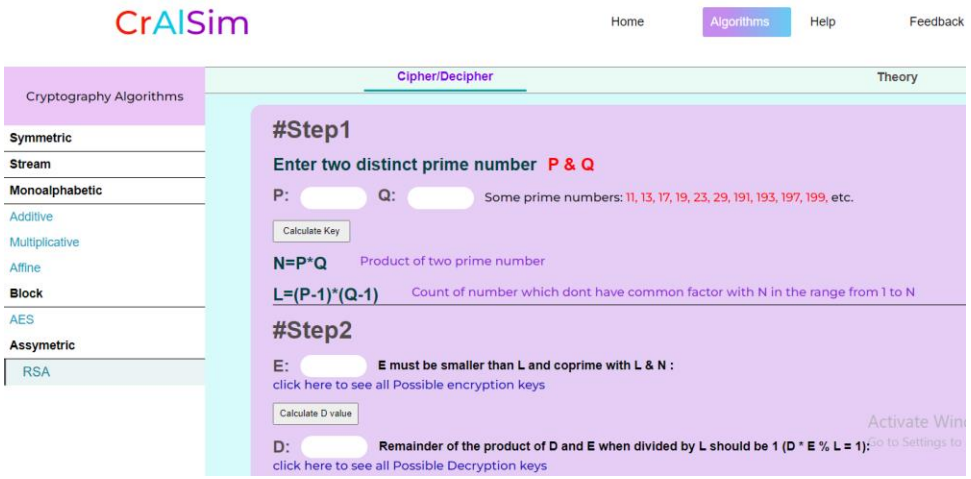


Fig 7.20 RSA main page

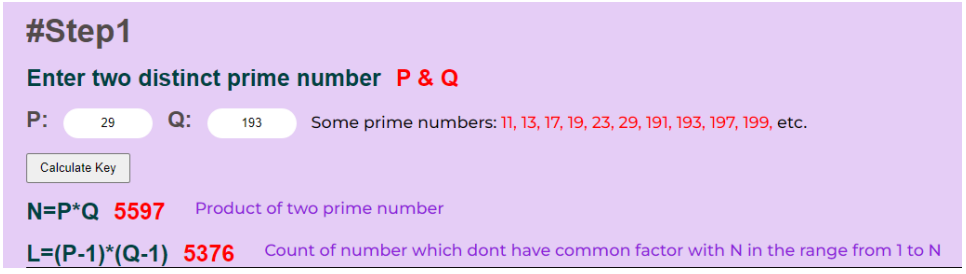


Fig 7.21 RSA Step 1

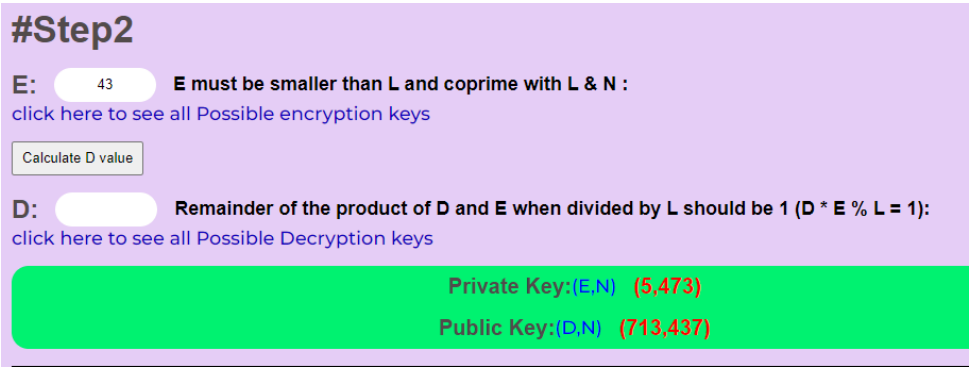


Fig 7.22 RSA Step 2

Fig 7.23 Possible values

Final step:

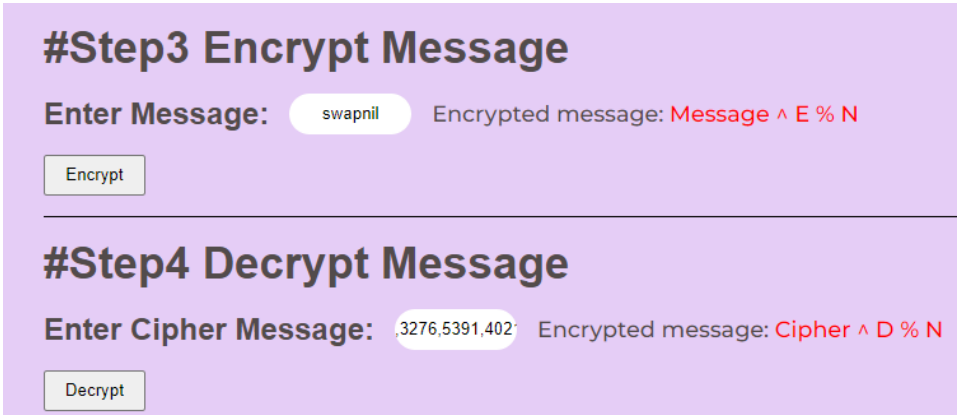


Fig 7.24 RSA encryption

Final step includes encrypting message ‘swapnil’ and hence the cipher message was generated. Private and public key formation was explained, and the simulation provides possible values to be chosen from too.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

This research carefully displayed the graphical picture of the internal operation of a few types of cryptographic algorithms and concentrated on the visualization tool for cryptographic algorithms that was proposed.
On the internet, there are many encryption tools, but not many visualization tools for these techniques.
The simulation stages are not stored in a relational database.
The primary algorithms in their categories are covered by CrAlSim, including Affine and AES. A simple demonstration of RSA is also provided [3].
Furthermore, complicated calculations, such as the initial round without an event and the 8 rounds of 4 distinct events in a loop for the 128-bit key in AES, can be visualized and understood rather than having students memorize the theory for each step.
Henceforth CrAlSim algorithm simulator intends to solve the problem of visual comprehension of cryptographic ciphers.

8.2 Future Scope

- To perform encryption on not just plain text but rather a whole file, say a txt or pdf or docx file shall be attempted to be encrypted with all its content inside.
- To include more enhancements, more detailed solutions of various rounds and events happening inside an algorithm, such as key generation, function callback visualization, more layouts and better counters for loops.
- To implement a greater number of algorithms of various categories like P-Box and S-Box block ciphers and more traditional ciphers.
- To embed a video-based feature so users can download MP4 file of an already performed simulation so they can keep a sample copy for further application and not perform the same simulation each different time he uses our simulator tool in website.
- Create comparison factors for two similar inputs which may generate different ciphers as output, as addressed in Vigenère test case 04.
- Increase color coding and simulate in depth fundamental and functional changes of the code like showing calculations of hexadecimal to string and other numeric conversions.
- Create study-based tutorials so after learning the functional working of code, the user can relay his understanding to positional changes of a new example plain message.

REFERENCE

REFERENCE

[1] L. Ogiela, M. R. Ogiela and U. Ogiela, "Cognitive information systems in secure information management and personalized cryptography," The joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS) were held in Kitakyushu in 2014, Japan, 2014, pp. 1152-1157.

[2] U. Arom-oon, Third Asian Conference on Defence Technology (ACDT), Phuket, Thailand: "An AES cryptosystem for small network" 2017, pp. 49-53.

[3] dCode's Tools List : https://www.dcode.fr/tools-list

[4] N. Buckley, A. Nagar and S. Arumugam, "Evolution of Visual Cryptography Basis Matrices with Binary Chromosomes," 8th EUROSIM Congress on Modelling and Simulation, Cardiff, UK, 2013, pp. 7-12.

[5] Crypto Corner: https://crypto.interactive-maths.com/

[6] Stylesuxx Steganography Online: https://stylesuxx.github.io/steganography/

[7] Python Tutor: https://pythontutor.com/

[8] SourceTrail: https://www.sourcetrail.com/

[9] "Visual Simulation Technique of Decision Making of Interactive Stand Management Methods," 2014 International Conference on Virtual Reality and Visualization, Shenyang, China, pp. 459-466. Y. Li, H. Zhang, H. Ju, and X. Jiang.

[10] Graph Buddy: https://plugins.jetbrains.com/plugin/13467-graph-buddy

[11] Bloom, Yuval & Fields, Ilai & Maslennikov, Alona & Rozenman, Georgi. (2022). A Simulated Undergraduate Experiment on Quantum Cryptography. Physics. 4. 104-123. 10.3390/physics4010009.

[12] An Effective Implementation of the RSA Algorithm Using FPGA and Big Prime Digit, Gurpreet K. and Vishal A. (2013), International Journal of Computer & Communication Engineering Research (IJCCER), Volume 1 - Issue 4.

[13] Yahia Alemami, Mohamad A Mohamed, and Saleh Atiewi. (2023). Advanced encryption method using chaotic map and advanced encryption standard. 13. 10.11591/ijece.v13i2.pp1708-1723. International Journal of Electrical and Computer Engineering (IJECE).

[14] A. Singh, "Comparatively analysis of AES, DES and SDES algorithms [9]" The research paper "Comparative analysis of encryption algorithms for various types of data files for data security," written by K. P. Karule and N. V. Nagrale, was published in 2014 in the International Journal of Computer Science and Technology. The paper's pages 200–202 were included in volume 8491, issue 3.

[15] "Security enhancement of AES based encryption using dynamic salt algorithm," in Proceedings of the 2018 International Conference on Applied Engineering, ICAE 2018, 2018, pp. 1-6, by M. M. Bachtiar, T. H. Ditanaya, S. Wasista, and R. R. K. Perdana.

[16] "A modified advanced encryption standard algorithm," M. I. Reddy, Journal of Mechanics of Continua and Mathematical Sciences, no. 1, 2020, pp. 112-117.

[17] A. Hafsa, A. Sghaier, M. Zeghid, J. Malek, and M. Machhout, "An improved co-designed AES-ECC cryptosystem for secure data transmission," International Journal of Information and Computer Security, 2020, vol. 13, no. 1, pp. 118–140.

[18] Jan Carlo Arroyo and Allemar Jhone Delima. (2020). An Affine Cipher Based on Keystream for Dynamic Encryption. 10.30534/ijeter/2020/06872020, International Journal of Emerging Trends in Engineering Research, 8. 2919–2922.

PAPER PUBLISHED

International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 04 | Apr 2023 www.irjet.net p-ISSN: 2395-0072

CrAlSim: A Cryptography Algorithm Simulator

Avinash Singh[1], Swapnil Mane[2], Mihir Bist[3], Samuel Jacob[4]

[1]*B.E. Information Technology, Dept. of Information Technology, Vidyalankar Institute of Technology, Maharashtra, India*
[2] *B.E. Information Technology, Dept. of Information Technology, Vidyalankar Institute of Technology, Maharashtra,India*
[3]*B.E. Information Technology, Dept. of Information Technology, Vidyalankar Institute of Technology, Maharashtra, India*
[4]*Professor, Dept. of InformationTechnology, Vidyalankar Institute of Technology, Maharashtra, India*

Abstract- *Encryption algorithms are an essential part of Information Security, as they safeguard the messages sent between users, and also protect the integrity and confidentiality of the system. For students to understand the internal working of complex algorithms, it's requisite that they don't mug up the steps happening in the cipher, but rather will learn better if it happens to be a graphical visualization of every event happening in the algorithm. This paper describes working of cryptographic algorithm simulation system. CrAlSim is based on matrix formation, stepwise color-based value change in code of the crypto algorithms, highlighted by each function showing the calculation right from the first to last step of encryption. The system created highlights how based on user inputs for a plain message or a key wherever required, how the actual inside conversions happen in an algorithm which led to the encryptions, be it a simple monoalphabetic cipher like Affine or block cipher like AES (Advance Encryption Standard). The encryption functions adjusted with the simulations are programmed in JavaScript, and simply HTML and CSS are used to display the web elements and styling, so understanding the source code even for creating of other algorithm visualization systems is comprehensible and apprehensible for a normal user. This paper elaborates the mechanism of CrAlSim and discusses the results obtained using it.*

Key Words: **Simulator; Encryption; Decryption; AES; Affine; RSA; Peer to peer message confidentiality; Algorithm simulation.**

1. INTRODUCTION

Nowadays, students often face the dilemma of choosing whether to take up security as one of their core domains. Since most students seem to face the problem in understanding various ciphers, they never chose to look further on how security works more than just simple cipher exchanging acknowledgement requests in SSL of presentation layer [1]. So, every IT student needs an easier guide to comprehend ciphers, be it of various types in an easy format. Since a picture is worth a thousand words, a step visualization simulation for demonstrating a cipher working with all the components running over every input character is much better than 500 pages of algorithm in theory mode of a published author. As most students face this issue after reaching their second year in IT or Computer branch, they are the target audience of this website. Encryption algorithms are very essential to be learnt in the current era, as malicious users most often are looking for unsecured data streams over the internet. If any attacks are launched, it can prove to be hazardous if sensitive data is revealed, and Government information, data related to financial sectors and other critical data can be leaked to foreign parties if it isn't encrypted properly. Hence these encryption algorithms need to be studied, and can be better studied by trainee engineers or IT students if they learn it via a visualization tool which will enhance their perception of all stages happening in the code of certain algorithms [9]. Such a tool is CrAlSim which provides a simulator for essential cryptographic algorithms.

1.1 Simulation Process

Visualization of algorithms is done using simple functions which target over the events which can be made graphical as so students or any user perceives those events in motion pictures or color highlights and hence better representation of various operational fields of code can be shown. First step is selection of cipher as in which category of the algorithm needs to be visualized. For a single word string, monoalphabetic ciphers such as Affine are useful as their encryption can be based on just a certain length and for a block of letters; AES (Advanced Encryption Standard) cipher can be chosen. User then needs to input the proper type of the plain text message, as certain criteria is set for certain ciphers based on length of string, or it's standard language used. Then the key is selected, or generated as per conditions, as RSA needs very high values for proper security. Encryption and decryption buttons should be used to generate cipher text or decipher plain message. Note must be made of the special instructions in the process, explaining as in which kind of key to select, or what conversions are happening. Then controlling the flow of simulation using the start, play, pause or reset buttons.

2. LITERATURE SURVEY

Through this section, we summarize some of the existing research work on either encryption or general simulation systems.

In January 2017, U. Arom-oon had proposed an AES cryptosystem for a small-scale network [2]. For low power sensor devices, the scale of the network based on wireless communication is quite small. To protect electronic data, FIPS 197 standard [12] encryption is used. Say a UAVs wireless communication network has microcontroller which works on real time operating system, so the main basis of security is chosen from the code book which is one mode of AES method. In real time scenarios, scheduler decides which role to get the nod based on the event having the top priority. Hence no clash of system calls happen in the system. The basis of the accomplishment is decided by the quality of security in communication of UAVs which consist of control and telemetry commands. The target hardware is implemented on the arm cortex-M4.

In December 2020, C. R. Martin proposed a system of A New Simulation Algorithm for PDEVS (Discrete Event Systems Specification) Models [5]. The algorithm deals with the timing of code executions, as functions can't be controlled in their flow of system calls just to wait for an animation packet to be displayed. In case of simulations, the user deals with the working of only general models and the corresponding DEVS simulations occur when the scheduled events are triggered. When system call occurs simultaneously inside a function for different events, processing happens with event A being executed firstly and later only event B is executed. But if animation call asks for the inputs simultaneously for different events, it may lead to incorrect and non-requisite results for simulation. So, in this paper, proposal of a new algorithm is laid out which makes sure that the model is such that output section of result is sent only on the condition that it's given to a corresponding event at the appropriate simulation time received by the model.

In November 2021, A. Kabulov and M. Berdimurodov proposed a system of optimal representation in the form of logical functions of microinstructions of cryptographic algorithms (RSA, El-Gamal) [6]. According to the present study, the algorithms can be described by a set of micro-instructions consisting of a sequence of operators. Based on an analysis of graphs, logic, algebraic, and matrix representations of algorithms, an implementation minimal logical representation for hard-drive implementation is constructed.

In September 2020, RahmaIsnaini Masya; Rizal Fathoni Aji; Setiadi Yazid proposed the Comparison of Vigenere Cipher and Affine Cipher in Three-pass Protocol for Securing Image [7]. The aim of this study was to utilize Vigenere Cipher and Affine Cipher in Three-pass Protocol and compare their encryption results and execution time at each stage of the protocol. The findings indicate that Affine Cipher outperforms Vigenere Cipher in terms of encryption results within the Three-pass Protocol. Different types of files, such as text, binary encoding for images, audio, and video, are frequently exchanged through the internet. The study focuses on securing images using classical cryptography algorithms. Vigenere Cipher and Affine Cipher are implemented in Three-pass Protocol to eliminate the need for key exchange. The key is kept secure by each sender and recipient, making it impossible for attackers to obtain it. But, the execution time for Affine Cipher in Three-pass Protocol is longer than that of Vigenere Cipher.

In August 2014, Y. Li, H. Zhang, H. Ju and X. Jiang proposed a Visual Simulation Technique of Decision Making of Interactive Stand Management Methods [10]. To adapt to intensive forest management, a technique platform was created that employs multiple visual simulation techniques. Thirteen stand management-related activities were customized and presented graphically using the WF technique. A flow model for decision-making during interactive stand management was developed based on human-computer interaction. The GDI+ drawing technique was used to simulate statistics of stand structure and state in 2D, while the MOGRE technique was used to simulate the stand scene in 3D. The effectiveness of the method was demonstrated by carrying out a case study of accretion cutting in a Chinese fir plantation, which showed that the method was flexible, visible and operable, and could be used to create stand management flow models., The aforementioned approach has the ability to replicate the decision-making process involved in selecting the most suitable method from a set of techniques based on the decision-making criteria. It can be effectively implemented in stand management practices and can optimize the method selection by taking into account the specific stand management objectives, ultimately enhancing the overall scientific management proficiency.

3. PROPOSED SYSTEM

With a view to simulate various cryptographic algorithms, there arises the feeling of obligation to create functions based on stepwise handling of the message data. This occurs when various functions are written as per the flow of the code, synchronized with each cipher generation, setting color divisions, creating loops for multiple steps and conditions. After simulation is complete, final result is shown by displaying the resultant string, by rather selecting the comparison of the plain text, or in different format based on user selected cipher. The simulator [11] parses all the values of calculations in a table which show the inside configurations happening in each step, be it alphabets or numbers fed into a formula, or passing through rounds of a code.

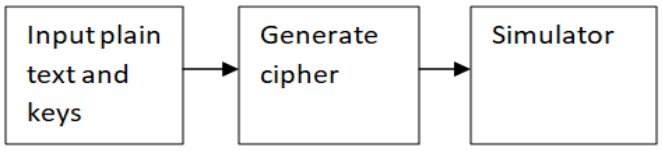


Fig 3.1: Block diagram of CrAlSim

Detailed explanation about modules in Fig 3.1 is given below-

1. Input plain text and keys module: In module one, once the user selects the type of cipher he needs, he must input his plain text message, which can be either in standard format, or can include basic punctuation in conjunction with numbers. The key or keys to be selected are mentioned in the note, for the criteria needs to be satisfied over which types of keys can be selected, e.g., for Affine cipher two keys are to be selected and they should be co-prime, in AES block cipher, a key can be 128, 192, or 256 bits. For our project permissible value chosen is 128 bits only, i.e., 16 characters.

2. Generate cipher module: In module two, we choose the encryption option. By giving in proper inputs for plain text message and keys, we can generate a cipher by clicking on the encrypt button. Following output is also shown on a general scale as in for Affine cipher, transformation of all alphabets is shown, hence user can tally how the change happened in original message and encrypted keyword.

3. Simulator: It consists of a JS code which checks all the necessary conditions for encryption to begin, and then creates the graphical representation of the working of the code, based on each step, value is fetched into matrices or tables, each new alphabet or number currently under operation is highlighted by a color or style which indicates the current step in progression, final transformations are duly saved in a all calculated values table so user can go through all the formula and numeric changes happened over each step or round.

4. IMPLEMENTATION AND RESULTS

4.1 Implementation

As process is initiated in the first module, encryption process occurs. User goes to homepage of our website and selects the cryptographic algorithm he wants to work upon, be it affine cipher of monoalphabetic type, AES of the category included in block cipher or RSA [13] a part of asymmetric cipher. Then the user enters his plain text message, can be in alphabetical form of characters, punctuation marks or numbers, then an appropriate key is selected. For affine cipher, key A and B should be co-prime of modulo m i.e., GCD of key and modulo m should be 1 as other combination result in two alphabets having same character as a cipher, so the inverse condition fails. Finally when all the clauses are satisfied, user can click on encryption button and generate the cipher.

In the second module, a graphical user interface (GUI) is rendered, as this is a tab where simulator works on the simulation [4], or graphical visualization of internal working of the algorithms. In correspondence to the highlighting of changes occurring in the code with respect to plain text message, internal cipher code function runs in synchronized manner. Based on that, value transformations happen which are exhibited via tables, matrices, or moving of object values. Each round of the function is demonstrated for each block of code or a single character, and to control the narrative over whether to watch all the steps or just comprehend one of the transformations and then skip to the final output, set controls are provided such as play, pause, skip and start buttons. Internal working of all calculations is shown in a table, or as in AES simulation, All Value button functionality is provided where all XOR, multiplication operations are accounted for each round. Final output value is converted back into a string, as calculations inside the function happen for hexadecimal values. So, the resultant cipher is returned back to the user in module one [15].

Flow chart for CrAlSim system is shown below:

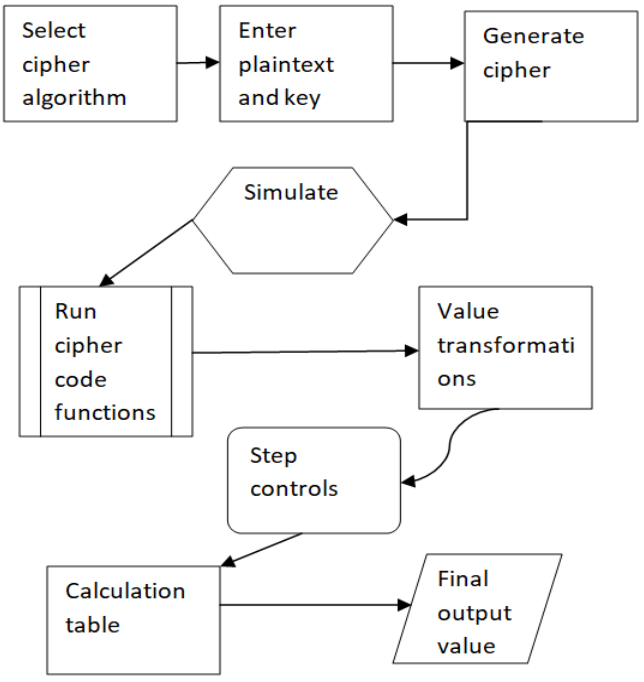


Fig 4.1: Flow chart for working of CrAlSim

As shown in Fig4.1, CrAlSim comprises of two modules, one displaying the basic encryption and other the actual simulation of internal working of algorithm.

4.2 Results

Examples when checked through the system showed accurate visualization graphics as intended and suited for the cryptographic algorithms.

AES:

Fig 4.2.1: AES MAIN PAGE

For plain text 'swapnil' and using 128 bits key 'mynameismynameis', cipher text generated after encryption was 'bP7T5B3lmL4lmkZvwmJ5dA'.

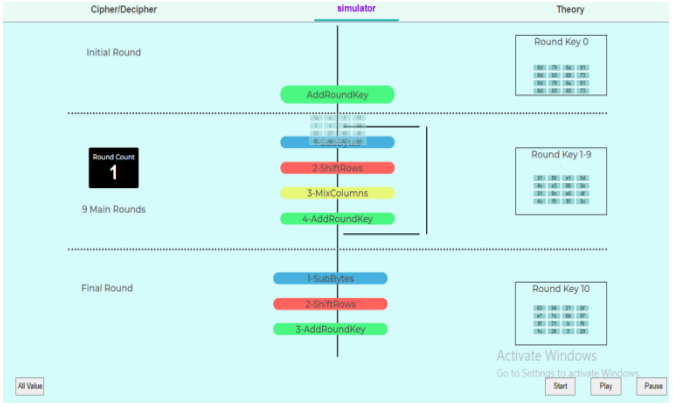


Fig 4.2.2: Initial round value pass

In initial round only XOR operation is performed between plain text converted into hexadecimal format put in a 4*4 array, (just like a matrix is created in hill cipher where each letter is represented by a number modulo 26) [8] along with the event round key word [0..3], and the output is calculated to the repetitive steps later in round 1.

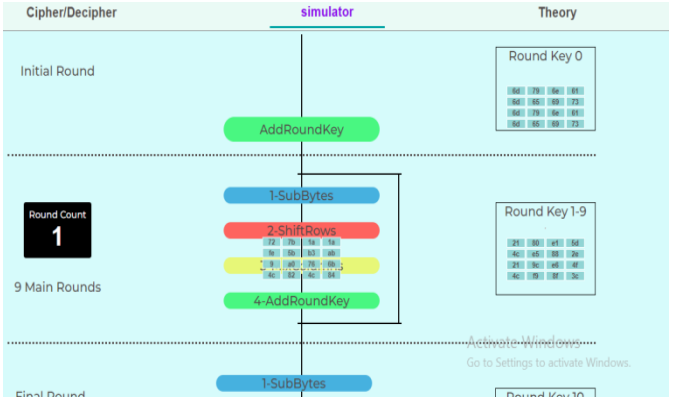


Fig 4.2.3: Events in a round

The initial round array meets various events, and undergoes transformations. SubBytes implements substitution using a lookup table also called the S-box, in next event, rows of the incoming 4*4 array are shifted certain times to left, MixColumns event does matrix multiplication and in final event Add Round Keys the resultant output of the previous stage is XOR-ed with the corresponding round key.

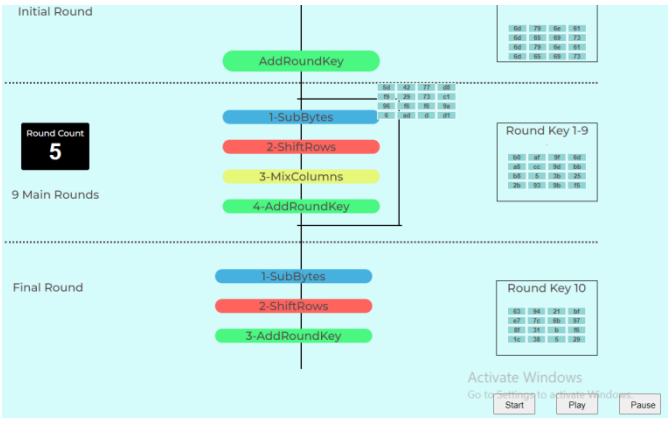


Fig 4.2.4: Repetition of rounds

All the above events are repeated again in a cyclic loop 9 times, the output photo shows how the array of add round key event went back as input to SubBytes. A counter is set to keep a track of the number of rounds ongoing.

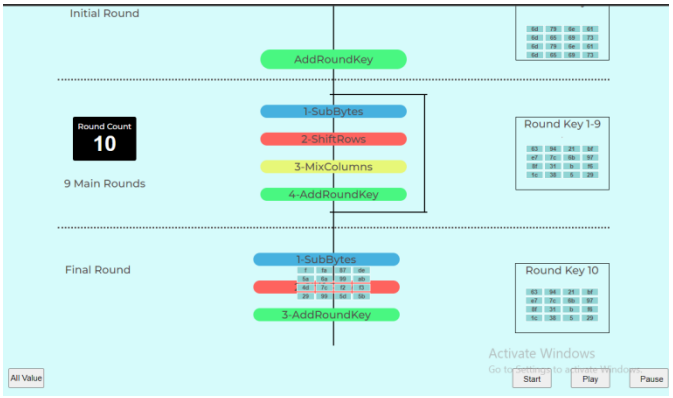


Fig 4.2.5: Final round of AES

After all the in between process changes, final round output is generated and the output shows how the array is flowing through the process, with no MixColumns event.

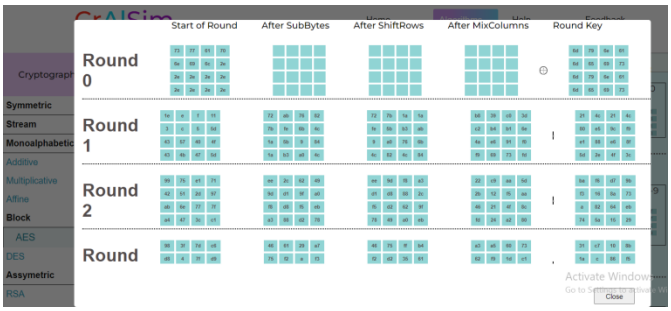


Fig 4.2.6: Calculation table

All intermediary calculations of events in each round are shown in the All-Value page. Each XOR result, along with mathematical operations of left shifts, matrix multiplication and substitutions can be seen in depth.

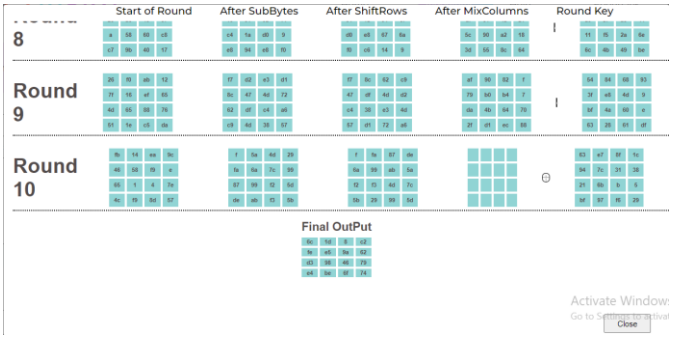


Fig 4.2.7: Final Output

Final output value is generated in a 4*4 array of hexadecimal value, which was converted back to base64 and shown in a proper string to user as the encrypted value in module one. As the output size remains same for an encrypted data, it can be compared to hashing which is irreversible, and helps in preventing possible issues of two encrypted strings having same input [16].

Affine:

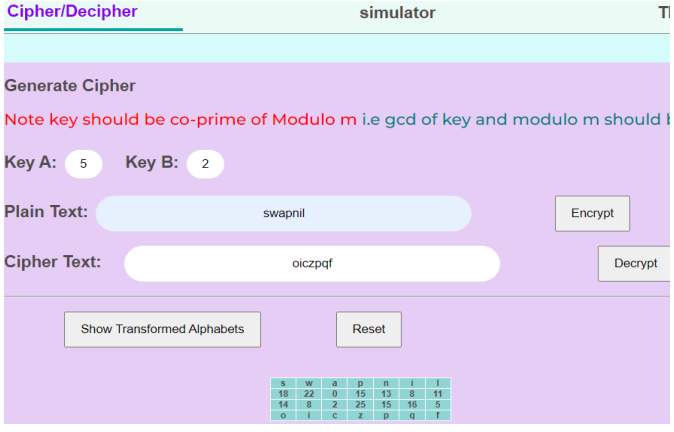


Fig 4.2.8: Affine MAIN PAGE

Key A is 5 and key B is given 2 as input, they satisfy the condition of being co-prime. Plain text is provided, and cipher is generated, and all transformed alphabets are represented in correspondence to their numeric values with letter 'a' starting from 0.

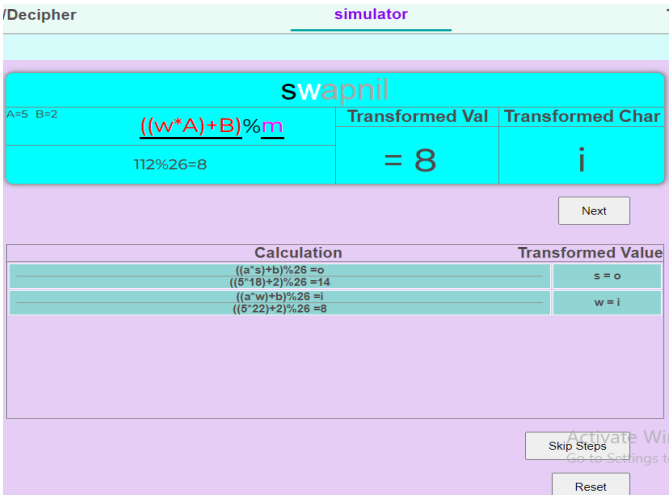


Fig 4.2.9: Simulator for Affine

For plain text 'swapnil', each letter [19] is fed into the formula (ax+b) %m, and subsequent calculations are shown each time Next button is clicked. Calculation for letter 's' is done, so it's highlighted in black, current letter under process is 'w' as it is seen in color white and yet to converted characters in gray. Every derivation and its resultant value are shown in transformation table with their calculations and changes implied.

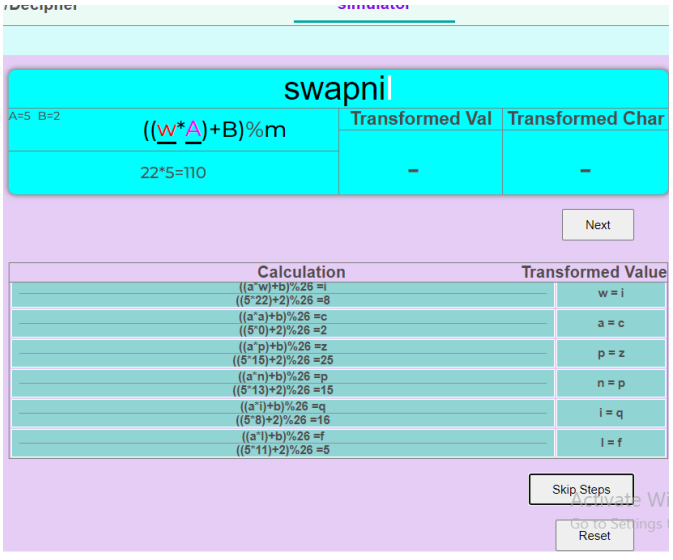


Fig 4.2.10: Skip Steps functionality

If the process is comprehended for one letter, going through each letter transformation can be tedious, hence skip steps leads us to eventual values and the output is completely generated with affine cipher.

5. CONCLUSIONS

This paper focused on the proposed visualization tool for cryptographic algorithms, and elaborately exhibited the graphical view of the internal working of a few types. There are several encryption tools available over the internet, but not many visualization tools for these algorithms. No relational database is used to store the simulation steps. The factor which makes CrAlSim advantageous is that the algorithms covered are the primary ones in their categories, like Affine in monoalphabetic cipher and AES in block cipher, simple demonstration is also made for RSA [3], and that the complex calculations like the initial round with no event and 8 rounds of 4 distinguished events in a loop for 128 bits key in AES all can be visualized and understood rather than students mugging up theory of each step. The existing simulator on further upgrade can be imbibed with more enhancements like adding more in-depth simulations of the multiple rounds and events, adding a video based feature to download a mp4 file of an instance simulation, adding comparison factors to two similar inputs but different ciphers generated as output, and also more highlighting of certain events in color scheme as what to visualize is different from the perception of each user.

REFERENCES

[1] L. Ogiela, M. R. Ogiela and U. Ogiela, "Cognitive information systems in secure information management and personalized cryptography," The city of Kitakyushu hosted the 2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and the 15th International Symposium on Advanced Intelligent Systems (ISIS), Japan, 2014, pp. 1152-1157, doi: 10.1109/SCIS-ISIS.2014.7044798

[2] U. Arom-oon, "An AES cryptosystem for small scale network," 2017 Third Asian Conference on Defence Technology (ACDT), Phuket, Thailand, 2017, pp. 49-53, doi: 10.1109/ACDT.2017.7886156.

[3] P. M. Aiswarya, A. Raj, D. John, L. Martin and G. Sreenu, "Binary RSA encryption algorithm," 2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kumaracoil, India, 2016, pp. 178-181, doi: 10.1109/ICCICCT.2016.7987940.

[4] N. Buckley, A. Nagar and S. Arumugam, "Evolution of Visual Cryptography Basis Matrices with Binary Chromosomes," 2013 8th EUROSIM Congress on Modelling and Simulation, Cardiff, UK, 2013, pp. 7-12, doi: 10.1109/EUROSIM.2013.12.

[5] C. R. Martin, G. G. Trabes and G. A. Wainer, "A New Simulation Algorithm for PDEVS Models with Time Advance Zero," 2020 Winter Simulation Conference (WSC), Orlando, FL, USA, 2020, pp. 2208-2220, doi: 10.1109/WSC48552.2020.9384028.

[6] A. Kabulov and M. Berdimurodov, "Optimal representation in the form of logical functions of microinstructions of cryptographic algorithms (RSA, El-Gamal)," 2021 International Conference on Information Science and Communications Technologies (ICISCT), Tashkent, Uzbekistan, 2021, pp. 1-4, doi: 10.1109/ICISCT52966.2021.9670149.

[7] R. I. Masya, R. F. Aji and S. Yazid, "Comparison of Vigenere Cipher and Affine Cipher in Three-pass Protocol for Securing Image," 2020 6th International Conference on Science and Technology (ICST), Yogyakarta, Indonesia, 2020, pp. 1-5, doi: 10.1109/ICST50505.2020.9732873.

[8] R. Mahendran and K. Mani, "Generation of Key Matrix for Hill Cipher Encryption Using Classical Cipher," 2017 World Congress on Computing and Communication Technologies (WCCCT), Tiruchirappalli, India, 2017, pp. 51-54, doi: 10.1109/WCCCT.2016.22.

[9] M. Kocheta, N. Sujatha, K. Sivakanya, R. Srikanth, S. Shetty and P. V. Ananda Mohan, "A review of some recent stream ciphers," 2013 International conference on Circuits, Controls and Communications (CCUBE), Bengaluru, India, 2013, pp. 1-6, doi: 10.1109/CCUBE.2013.6718558.

[10] Y. Li, H. Zhang, H. Ju and X. Jiang, "Visual Simulation Technique of Decision Making of Interactive Stand Management Methods," 2014 International Conference on Virtual Reality and Visualization, Shenyang, China, 2014, pp. 459-466, doi: 10.1109/ICVRV.2014.37.

[11] Bloom, Yuval & Fields, Ilai & Maslennikov, Alona & Rozenman, Georgi. (2022). Quantum Cryptography—A Simplified Undergraduate Experiment and Simulation. Physics. 4. 104-123. 10.3390/physics4010009.

[12] Walter Tuchman (1997). "A brief history of the data encryption standard". Internet besieged: countering cyberspace scofflaws. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA. pp. 275–280.

[13] Gurpreet K. and Vishal A., (2013), "An Efficient Implementation of RSA Algorithm using FPGA and Big Prime Digit", International Journal of Computer & Communication Engineering Research (IJCCER), Volume 1 - Issue 4.

[14] Alemami, Yahia & Mohamed, Mohamad A & Atiewi, Saleh. (2023). Advanced approach for encryption using advanced encryption standard with chaotic map. International Journal of Electrical and Computer Engineering (IJECE). 13. 10.11591/ijece.v13i2.pp1708-1723.

[15] A. Singh, "Comparatively analysis of AES, DES and SDES algorithms [9]" In 2014, the International Journal of Computer Science and Technology published a research paper titled "Comparative analysis of encryption algorithms for various types of data files for data security," authored by K. P. Karule and N. V. Nagrale. The paper was published in volume 8491, issue 3, and spanned pages 200-202.

[16] M. M. Bachtiar, T. H. Ditanaya, S. Wasista, and R. R. K. Perdana, "Security enhancement of AES based encryption using dynamic salt algorithm," in Proceedings of the 2018 International Conference on Applied Engineering, ICAE 2018, 2018, pp. 1–6, doi: 10.1109/INCAE.2018.8579381.

[17] M. I. Reddy, "A modified advanced encryption standard algorithm," Journal of Mechanics of Continua and Mathematical Sciences, no. 1, pp. 112–117, 2020, doi: 10.26782/jmcms.spl.5/2020.01.00027.

[18] A. Hafsa, A. Sghaier, M. Zeghid, J. Malek, and M. Machhout, "An improved co-designed AES-ECC cryptosystem for secure data transmission," in International Journal of Information and Computer Security, 2020, vol. 13, no. 1, pp. 118–140, doi: 10.1504/IJICS.2020.108145.

[19] Arroyo, Jan Carlo & Delima, Allemar Jhone. (2020). A Keystream-Based Affine Cipher for Dynamic Encryption. International Journal of Emerging Trends in Engineering Research. 8. 2919-2922. 10.30534/ijeter/2020/06872020.

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

Avinash Singh

In recognition the publication of the manuscript entitled

CrAlSim: A Cryptography Algorithm Simulator

published in our Journal Volume 10 Issue 4 April 2023

Editor in Chief

Impact Factor : 8.226

www.irjet.net

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

Swapnil Mane

In recognition the publication of the manuscript entitled

CrAlSim: A Cryptography Algorithm Simulator

published in our Journal Volume 10 Issue 4 April 2023

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

Mihir Bist

In recognition the publication of the manuscript entitled

CrAlSim: A Cryptography Algorithm Simulator

published in our Journal Volume 10 Issue 4 April 2023

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

Samuel Jacob

In recognition the publication of the manuscript entitled

CrAlSim: A Cryptography Algorithm Simulator

published in our Journal Volume 10 Issue 4 April 2023

Impact Factor : 8.226

www.irjet.net

Editor in Chief

E-mail : editor@irjet.net

DrillBit

The Report is Generated by DrillBit Plagiarism Detection Software

Submission Information

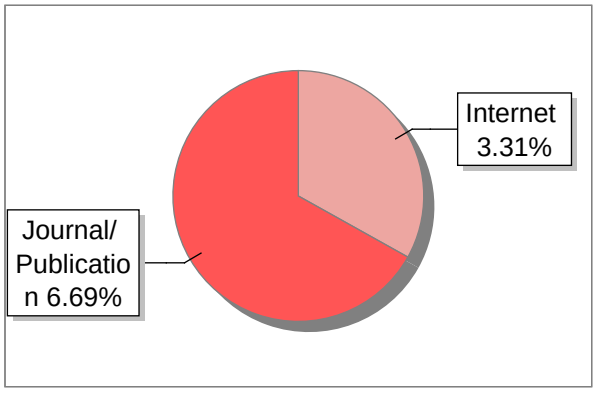
Author Name	Samuel Jacob
Title	CrAlSim: A Cryptography Algorithm Simulator
Paper/Submission ID	726072
Submission Date	2023-04-24 12:12:57
Total Pages	66
Document type	Dissertation

Result Information

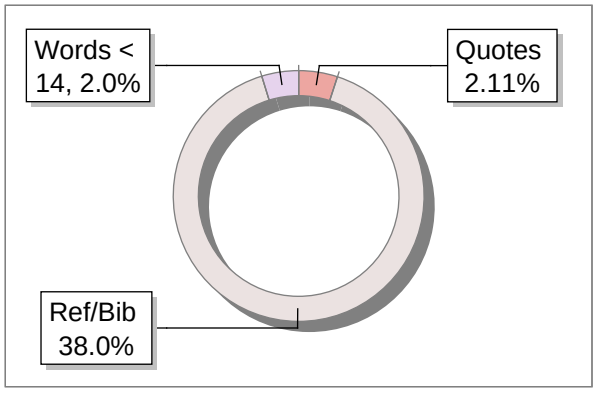
Similarity **10 %**

1 10 20 30 40 50 60 70 80 90

Sources Type



Report Content



Exclude Information

Quotes	Not Excluded
References/Bibliography	Not Excluded
Sources: Less than 14 Words Similarity	Not Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

A Unique QR Code use to View/Download/Share Pdf File