Creating a new Cryptographic algorithm using Collatz Conjecture

by

Shoumya Shuprabho Rasheed

18201055

Rakibul Hasan Remon

18201139

Monwar Labib

21101095

A thesis submitted to the Department of Computer Science and Engineering

in partial fulfilment of the requirements for the degree of

B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering

Brac University

September 2022

© 2022. Brac University

All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is our original work while completing a degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material that has been accepted or submitted for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Shoumya S. Rasheed

Shoumya Shuprabho Rasheed
18201055

Rakibul Hasan Remon

Rakibul Hasan Remon
18201139

Monwar Labib

Monwar Labib
21101095

Approval

The thesis titled “Creating a new Cryptographic Algorithm using Collatz Conjecture” submitted by

1. Shoumya Shuprabho Rasheed (18201055)
2. Rakibul Hasan Remon (18201139)
3. Monwar Labib (21101095)

Of Summer 2022 has been accepted as satisfactory in partial fulfilment of the requirement for the degree of B.Sc. in Computer Science and Engineering on September 28, 2022.

Examining Committee:

Supervisor:
(Member)

[illegible]

Mobashir Monim
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Associate Professor and Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Owing to the increasing need for the security of information and data access, due to the steep increase in the rate at which there are more methods of breaking the existing algorithms, which primarily rely on a prime number, this thesis looks at the possible use of the Collatz Conjecture to create a new cryptographic algorithm as an alternative to existing methods. This research document discusses network security and different encryption and decryption schemes. We have highlighted and analysed AES, RSA, and a few others and how each model implements different mathematical behaviour. In this age of fast communication, users are often more concerned about their information being transmitted as swiftly as possible without overthinking its security. Cryptography is a method that has been used for data security for many years now, and new improvised methods have been introduced to enhance the already existing norms. Our proposed model attempts to create safe and secure communication over a networked system. The main objective of this research is to find a novel data security method that would strengthen the already existing ones.

Keywords: Keywords: Cryptography; Encryption; Decryption; Collatz Conjecture; Symmetric; Asymmetric; Diffie-Hellman

Dedication

We dedicate our thesis to our friends, family, and, most importantly, our teachers.

Acknowledgement

First and foremost, all praise to the Almighty Allah for whom we have managed to complete our thesis smoothly. Secondly, to our Supervisor, Mobashir Monim, for his kind support and advice in our work. He helped us whenever we needed help. Moreover, we sincerely express our gratitude to all our respective teachers, companions, and staff for all the support throughout. Finally, we are forever indebted to our parents for their continuous support and prayers. It is only because of them we got this amazing opportunity to learn and complete our undergraduate degree. With their kind support and prayer, we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$3DES$	Triple Data Encryption Algorithm
AES	Advanced Encryption Standard
$ASCII$	American Standard Code for Information Interchange
$D-H$	Diffie-Hellman
DES	Data Encryption Standard
ECC	Elliptical curve Cryptography
$GCHQ$	Government Communication Headquarters
$IDEA$	International Data Encryption Algorithm
MDS	Maximum Distance Separable
PDF	Portable Document Format
PHP	Hypertext Preprocessor
PHT	Pseudo Hadamard Transform
$PWLCM$	Piece Wise Linear Chaotic Map
RAM	Random-Access Memory
ROM	Read-only Memory
RSA	Rivest-Shamir-Adleman
$S-BOX$	Substitution Box
TM	Trade Mark
URL	Uniform Resource Locator
$UTF-8$	Unicode Transformation Format 8
$WWII$	World War II
$X-OR$	eXclusive OR

Chapter 1

Introduction

Technology has advanced over the years at the speed of light. We have achieved so much in the past few decades that nothing seems impossible now. With the ever-growing world of information and data, it has become one of the most crucial and, at the same time, gruelling tasks to make sure that data is secured and not going into the hands of a transgressor. Modern technology involves a great deal of data that requires confidentiality. Cryptography is a method by which that is achieved. [18] In the beginning, the Roman cryptography method was known as the Caesar Shift Cipher. A particular number was fixed (three being a common choice), and that number shifted the letters of a message. Then, the letters of the recipient of this message were moved back by the same number, and the original message was obtained. Since that time, it has been upgraded quite a few times. It involves techniques that ensure data security for reliable and safe communication.

When describing cryptography, two kinds of texts are in use; plaintext and ciphertext. Plaintexts are transparent data understandable to everyone who has access to it. Ciphertexts are plaintexts that have been encrypted using cryptography.

Cryptography is mainly divided into two major classes; transposition and substitution. Furthermore, two basic [1] and almost quite outdated methods are commonly used for cryptography; monographically, involving one character at a time, and poly-graphically involving more than one character.

The rearrangement of characters to encrypt information is the primary role of cryptography through transposition. The characters in the plaintext each have a different frequency of occurrence distributed over a wide range, but a cryptanalyst can easily deal with this. A cryptanalyst performs analysis on a ciphertext to transform it back to plaintext.

The primary motivation behind this research is to create a new system for information and data security that would increase the strength of the reliability of communication over the network. And to bridge the gap between security issues and safe communication, cryptography will play a major role.

1.1 Research Problem

As the internet expands, it is more susceptible to the breach of sensitive data, and now more than ever, cyber security requires a robust system to ensure information security. As the world advances with technology every day, all kinds of information are transmitted over the internet. Networked computers, smartphones, and all other personal computers have made information and data less physical [6] and thus more open to the possibility of getting into the hands of transgressors.

Over the past few decades, the increased number of networked computers and systems resulted in a surge in cybercrime. Therefore, more opportunities are created for unauthorised activities due to billions of internet users and countless information held by governments and businesses. The lack of knowledge of internet users and their vulnerability contributes to this global issue.[6]

Even after meteoric advancements in information, communication, and the implication of data science in various fields of technology, numerous threats to information encryption are still a matter of concern. As of 2021, according to [20], the Covid-19 pandemic has influenced the practice of remote work, which has significantly raised the rate of cyber threats and has opened up new trends in cyber security. This indicates that we are still lagging in information security and have not yet achieved a solid solution.

Therefore, cryptography and its advancements are essential to combat different forms of cyber threats, especially those involving data theft, data breach, and other unavoidable vulnerabilities affecting data transfer.

1.2 Research Objectives

As mentioned earlier, the main goal of this research is to create an algorithm for cryptography that enhances the efforts to stop all forms of threats to the information system. To achieve that, we will try to incorporate a relatively new technique using an unconventional mathematical sequence to build the model. The use of this quite mysterious yet extremely intriguing mathematical sequence, the Collatz Conjecture, will be the most challenging part.

Chapter 2

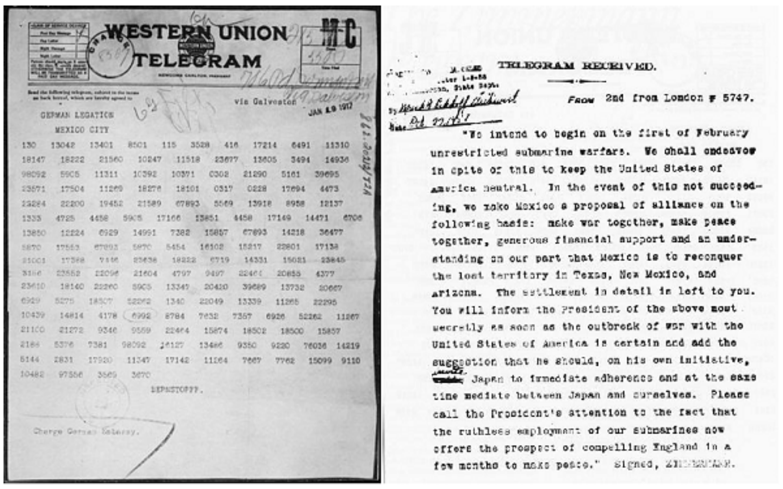
Historical Cryptography

2.1 World War I

In early 1917, British cryptographers discovered a German-encoded telegraph during the early stages of World War I. This communication was known as the Zimmerman Telegram. These cryptographers deciphered the message successfully, altering the direction of cryptanalysis. Using this decoded correspondence, they urged the US to join the war [14].

The Zimmerman Telegram was a top-secret message sent between German Foreign Secretary Arthur Zimmerman and German Ambassador Heinrich von Eckardt in Mexico. According to the letter, if Mexico joined the German side, it might reclaim its territories of New Mexico, Texas, and Arizona. Despite this offer, Mexico decided that reclaiming its former territories was neither practicable nor desirable.

The telegraph was delivered at the height of World War I [14]. Until then, the United States has attempted to preserve its neutrality. The British and other allies had appealed to the US for help, and public opinion in the US was progressively shifting toward war. The British conveyed the decoded telegram to the United States on February 24, 1917, and On April 6, 1917, the US declared war against Germany and its allies. [5]



WESTERN UNION TELEGRAM

via Galveston JAN 19 1917

GERMAN LEGATION
MEXICO CITY

```
130    13042  13401  8501   115    3528   416    17214  6491   11310
18147  18222  21560  10247  11518  23677  13605  3494   14936
98092  5905   11311  10392  10371  0302   21290  5161   39695
23571  17504  11269  18276  18101  0317   0228   17694  4473
22284  22200  19452  21589  67893  5569   13918  8958   12137
1333   4725   4458   5905   17166  13851  4458   17149  14471  6706
13850  12224  6929   14991  7382   15857  67893  14218  36477
5870   17553  67893  5870   5454   16102  15217  22801  17138
21001  17388  7446   23638  18222  6719   14331  15021  23845
3156   23552  22096  21604  4797   9497   22464  20855  4377
23610  18140  22260  5905   13347  20420  39689  13732  20667
6929   5275   18507  52262  1340   22049  13339  11265  22295
10439  14814  4178   6992   8784   7632   7357   6926   52262  11267
21100  21272  9346   9559   22464  15874  18502  18500  15857
2188   5376   7381   98092  16127  13486  9350   9220   76036  14219
5144   2831   17920  11347  17142  11264  7667   7762   15099  9110
10482  97556  3569   3670
```

BERNSTORFF.

Charge German Embassy.

TELEGRAM RECEIVED.

FROM 2nd from London # 5747.

"We intend to begin on the first of February unrestricted submarine warfare. We shall endeavor in spite of this to keep the United States of America neutral. In the event of this not succeeding, we make Mexico a proposal of alliance on the following basis: make war together, make peace together, generous financial support and an understanding on our part that Mexico is to reconquer the lost territory in Texas, New Mexico, and Arizona. The settlement in detail is left to you. You will inform the President of the above most secretly as soon as the outbreak of war with the United States of America is certain and add the suggestion that he should, on his own initiative, invite Japan to immediate adherence and at the same time mediate between Japan and ourselves. Please call the President's attention to the fact that the ruthless employment of our submarines now offers the prospect of compelling England in a few months to make peace." Signed, ZIMMERMANN.

Figure 2.1: Zimmerman Telegram

2.2 World War II

German field agents used the Enigma machine to encode and decode letters and communications throughout World War II. The Enigma machine, like the Feistel function of the 1970s, was one of the first automated ways of encrypting data using an iterative cipher. It utilised a series of rotors to allow the user to encode or decode a message with the use of power, a fluorescent bulb, and a reflection. The starting position of the rotors was determined with each encrypted data and was based on a predetermined sequence that was based on the calendar, enabling the system to be used even if this was compromised.

When the Enigma was in use, each keypress caused the rotors to move in alignment from their predefined positions, resulting in the formation of a new letter. With a text in hand, the user would input each letter into the machine using a typewriter-like key. The rotors would align, and a character would illuminate, exposing to the operator the letter's real identity. When the operator hits the key, the illuminated letter transforms into encrypted text. Although the rotors' continually [22] changing the internal amount of energy was not random, it did produce a simple substitution cipher that may be unique. [19]

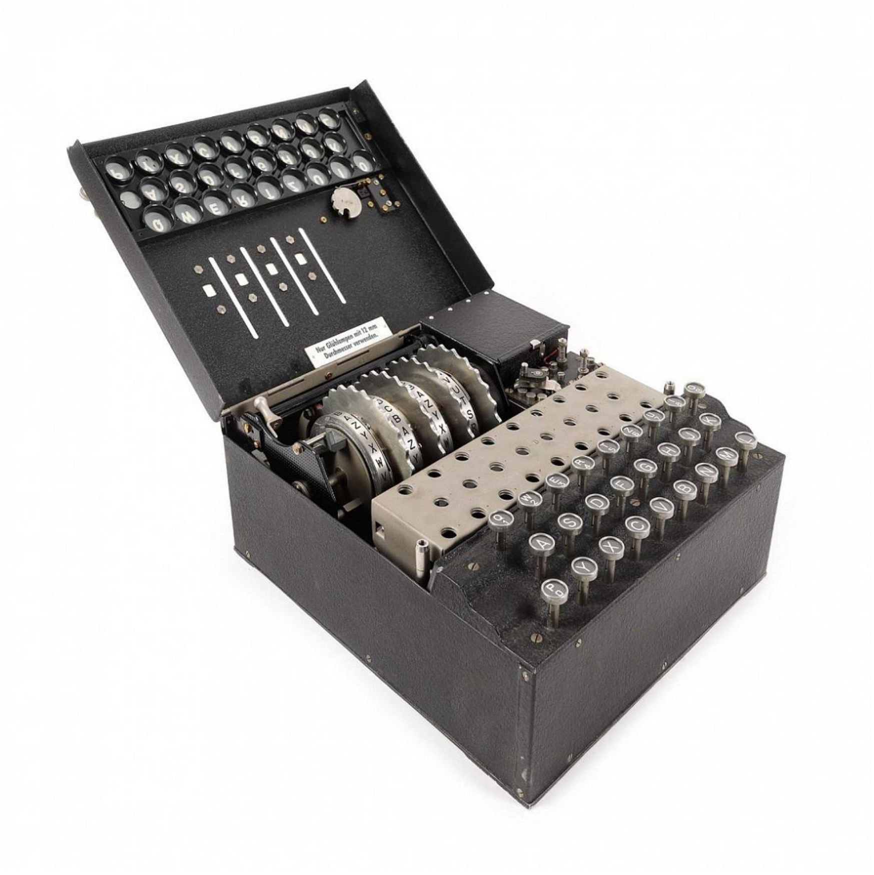


Figure 2.2: The Enigma machine

Chapter 3

Literature Review

To understand cryptography and its applications, we must understand how it has been used over the years. It is not a modern concept; rather, it has been in practice for a long time to secure communication. Before the introduction of modern encryption and decryption devices, cryptography was used to cover sensitive information using basic written texts by soldiers in wars, military leaders, [15] spies, etc. With the introduction of modern technology, it is now used not just for information confidentiality but also for message integrity checking, identification authentication, interactive proofs, etc.

In today's world, every commercial application needs a higher level of security and confidentiality of information. As a result, to increase the quality of confidentiality of information, numerous steps have been taken to improve the cryptographic algorithm. Cryptographic algorithms must be designed with an even distribution of computational power and security. As mentioned earlier, algorithms nowadays use either symmetric or asymmetric caters according to their needs. Symmetric and asymmetric algorithms are used for the conversion of plaintext to ciphertext. A ciphertext is a parametrized family which consists of encryption functions. The reason for this construct to ciphertext is that hackers cannot invert the encryption function except for people who know the key, as a result, they can see that there is a message but cannot read it as they do not have the key and therefore do not know which function from the family to invert from.[15]

As we have seen before, cryptography's importance and application are very well-acknowledged. However, extensive research on different cryptographic algorithms and its model has not been done for a very long time [15]. With the ongoing development of modern technology, the demand for different encryption techniques has been on the rise. As we learn and research more about the history of different cryptography, we must be proactive and keep in mind the possible future developments.

As mentioned, we will focus on our cryptographic model incorporating Collatz conjecture or the 3n+1 mathematical sequence. To do so, it is important to have deep knowledge of the existing mathematical model for symmetric and asymmetric cryptographic algorithms.

3.1 Definition

3.1.1 Cryptography

Cryptography studies secure communication between a sender and receiver. Its main goal is to convert messages and data over a communication channel so that only the sender and its dedicated recipient can interpret the message. Cryptography includes encryption, hashing, etc. Various cryptographic methods have been developed over the years using complex mathematical logic to create encryption and decryption algorithms. These algorithms are used to increase the integrity of communication.

3.1.2 Symmetric Cryptography

For encryption and decryption, the same key is used in this method of cryptography. The sender and receiver must have a shared key when converting plaintext to ciphertext and vice versa.

3.1.3 Asymmetric Cryptography

To tackle the major challenges of key creation and distribution, asymmetric cryptography was developed. Each sender and recipient have their own key in this sort of encryption. An asymmetric cryptosystem contains a public and private key for each user. At all times, the secret key must be kept private. However, the public key can be distributed. Encrypted data may only be decrypted using its complementary private key.

3.1.4 Hashing

The job of hashing is to calculate a fixed-size bit string value from a file that contains a block of data. Hashing converts this data into a shorter fixed length that contains the original string. The hash value contains the summary of the entire file. Hashing helps to gain faster access to elements in the string.

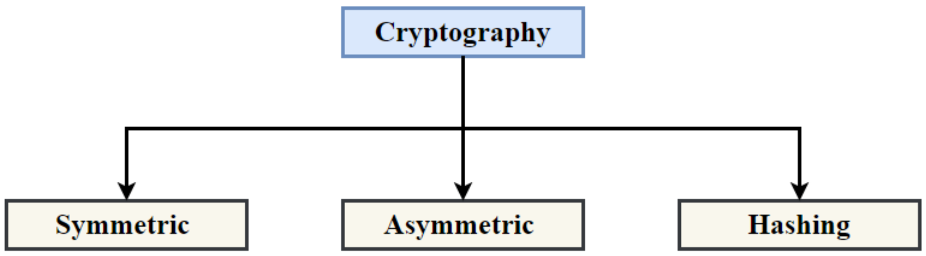


Figure 3.1: Types of Cryptography

3.1.5 Encryption

To simply define encryption, it is the conversion of information and data into unintelligible codes that would prevent unauthorised access to it. It is a process to ensure the digital protection of data using mathematical or other logical methods. Encryption enhances the integrity and maintains privacy in a communication network.

3.1.6 Decryption

The process of returning data to its original state after it has been encrypted is known as decryption. During decryption, the system brings out and transforms the jumbled data into letters and visuals that not only the reader but also the system can comprehend. Decryption can be done manually or automatically. However, a combination of keys or passwords can also be used.

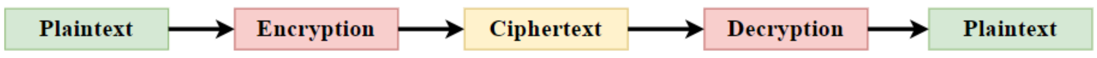


Figure 3.2: Basic Workflow of Encryption and Decryption

3.1.7 Collatz Conjecture

Collatz conjuncture is a unique and simple approach initially devised by German mathematician Lothar Collatz in 1937 [2]. Although it is impossible to explain this theorem mathematically, it is theoretically proven. If any number is even, n/2 would be used. In the case of odd numbers, it will be 3n+1. Using this approach, the result will always be a positive integer of 1. Until the sequence reaches an infinite loop of 4, 2, and 1.

3.1.8 DIFFIE HELLMAN

The Malcolm Williamson of GCHQ originally created the Diffie-Hellman key exchange method. It was later improved, and only a shared secret could be created using this technique. Its security is given by the discrete log problem's computational complexity. The common symmetric key would be $(g^b)^a = g^{ba} = g^{ab}$ mod p.

3.1.9 Shift Cipher

Caesar Cipher is known as one of the most basic and earliest encryption systems. Another name is substitution Cipher. In this method, one letter is substituted by another letter, moving the alphabet down in certain places, and every letter shifts one bit right or left.

3.1.10 Byte Injection

The concept of byte injection is similar to that of a null byte injection, where null characters are injected into a URL that alters the information.

3.1.11 Byte X-OR

Byte X-OR or an eXclusive OR operation is all about a key size ranging from 1-256 values of a byte. For each corresponding bit, the byte X-OR takes patterns of 2-bit of equal lengths and performs eXclusive OR operations.

3.1.12 Byte Substitution

Byte substitution is a type of substitution cipher. Generally, a single letter or pair of letters from the regular text are substituted with other symbols or groups of symbols in a substitution cipher, a data encryption system.

3.2 Related Work

Here, we have reviewed some popular existing models of cryptography which are already in use or have been used in the past. We have tried to figure out the mechanism for each cryptographic system to determine how we can proceed with our model. We have highlighted some key operations used for cryptography and a few of the algorithms that have already incorporated them.

3.2.1 Diffie-Hellman

The Diffie-Hellman algorithm is named after White Diffie and Martin Hellman. The main function of this algorithm is to use a secure channel for the transfer of a public or a private key. The Diffie-Hellman algorithm works in the following way:(1)The receiver has the ownership of both the private and public keys generated by the Diffie-Hellman algorithm. (2)Sender acquires the public key that the receiver generates, which is then used in the Diffie-Hellman algorithm to establish a new set of public keys that will be used temporarily. The sender then takes the recently generated public keys which were created by the receiver to create an unplanned secret number known as the “session key”, which will be used to encrypt the forwarding data, including the temporary public key. Subsequently, the receiver can derive the session key mathematically; as a result, the rest of the ciphertext message can be decrypted easily. [7]

3.2.2 Byte Substitution

Polyalphabetic Substitution Cipher

This article [9] shows us how polyalphabetic cipher maintains data integrity using multiple subkeys for every plaintext letter. Cipher substitution has its fair share of use in many algorithms. Polyalphabetic cipher uses alphabetic substitution. In a polyalphabetic substitution cipher, the plaintext is enciphered upon its arrival in the text. Here each letter carries a one-to-many connection with its alternatives. The term “Polyalphabetic” refers to using multiple keys instead of a single one. Here each key is a stream of subkeys that solely depends on the position of the plaintext character.

For example: ‘b’ can be enciphered as ‘e’ at the start of the text and as ‘n’ in the middle. As a result, polyalphabetic cipher hides the initial letter frequency, making it difficult for attackers since they cannot use individual letter frequency static to divide the ciphertext.

Polygraphic Substitution Cipher

In [2] it demonstrates how a polygraphic substitution cipher is built up and works to perform tasks that are superior to Polyalphabetic Substitution Cipher. Technically we can say, Polygraphic substitution Cipher is an upgrade to Polyalphabetic Substitution Cipher, with the substitution that involves replacing a group of Plaintext messages with another different group of characters from the ciphertext. In Polygraphic Substitution Cipher, the plaintext letter is divided into groups of adjacent letters while maintaining the same length, then transformed with different letters while retaining the same size in cipher text.

AES

With the extensive use of computers and the growing need for computer technology, particularly in military and government [11] functions, computer security is becoming more important. The issue to be solved is how to ensure the confidentiality, integrity, and usefulness of a messaging system’s hardware and software and the message being processed, stored, and delivered. The AES algorithm works with 128-bit input and produces 128-bit output. The key used to encrypt the input data might be 128, 192, or 256 bits long. Encryption and decryption often employ the 128-bit key. It makes use of the same key for encryption and decryption. AES uses multiple rounds of different key sizes e.g.; 10 rounds for 128-bit keys. For any key size, the first round is common in which the plain text is X-ORed with the key to produce 128, 192, or 256 states. The operations shiftRows, subBytes, mixColumns, and addRoundKey are performed on the states by each middle round. There is no [12] mixColumns step in the last round, comparable to the middle of the step. The opposite of encryption is decryption. The Inverse mixColumn Steps are not included in the final decryption rounds.

3.2.3 Byte X-OR

Blowfish
This article [21] describes the history and working mechanism of Blowfish. Blowfish is a symmetric algorithm, 64-bit block cipher with a changeable length. It was created in 1993 by Bruce Schneier as a ”general-purpose algorithm.” It was developed to be fast, untied, and a possible replacement for DES and IDEA algorithms. Blowfish consists of a block size of 64 and a key size length that ranges from 32 to 448 bits. Blowfish also has 16 Feistel iterations, where each iteration acts on 64-bit blocks divided into two 32-bit words. One encryption key is used for the encryption and decryption process.

Blowfish is divided into two parts:
1. Data encryption: It is a 16-round Feistel network that consists of a key-dependent permutation and a key-dependent data substitution. S-boxes are used during the substitution method, which plays a vital part in the encryption process. The encryption process mainly works in X-OR logic gates.
2. Key expansion and subkeys: In this process, the bit keys are divided into several subkeys in this process. These subkeys play an essential factor in the algorithm. Eighteen 32 bits are in the P array with four 32 bits S-boxes that consist of 256 entries. The subkey calculation is as follows:

- The S and P boxes are adjusted to hexadecimal digits of pi.
- The first elements in the P array, P1, are X-ORed with the initial 32 bits. The same goes for P2, which is X-ORed with the next 32 bits. The process goes on till every P-array is X-ORed.
- The zero strings in the algorithm are encrypted.
- The subsequent arrays are replaced with the output.
- The output gets encrypted using blowfish by using modified subkeys.
- After the previous process is completed, the P3 and P4 are modified.
- It goes until all P arrays and S-boxes are reformed.

Twofish

This article [3] explains how Twofish was designed to overcome blowfish. Since blowfish had some limitations. Twofish is the successor of blowfish and uses a single key for both encryption and decryption purposes; it can have a block size of 128, 192, and 256 bits. Twofish can be used in network applications where keys change frequently and in systems where minimum or even no RAM or ROM is needed.

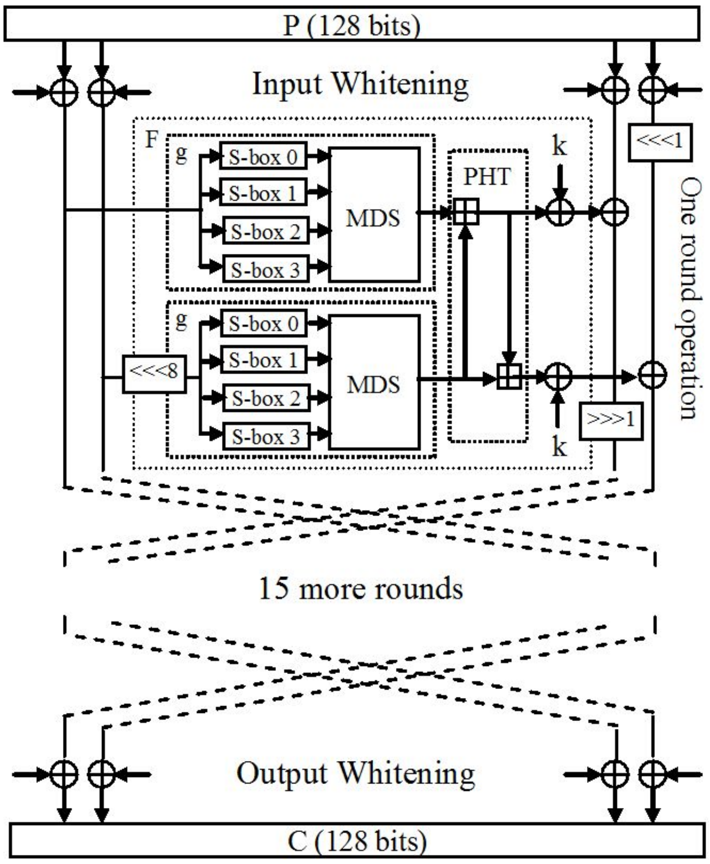


Figure 3.3: Twofish Mechanisms

Through this figure, we can see that Twofish is a Feistel network. This means that half of the function is passed through an F function, and the later half is X-ORed with the result that comes out from the F function.

During each round, two 32-bit words are passed into the F function. Four bytes are formed from each word, which is forwarded to four different key-dependent S-boxes. The result of the 4 output bytes is then combined with the help of the MDS matrix and formed into a 32-bit word. After that, the two 32-bit words were combined with the help of PHT, and then they were passed on to be added to two round subkeys and, later on, X-ORed with the right half of the text. 1-bit rotations occur before and after the X-OR operation happens. To prevent an attack, an additional round is added by pre-whitening and post-whitening to ensure security; they act as additional subkeys, which are then X-ORed before and after the last round in the text block.

3.2.4 Null Byte Injection

[4] shows how null byte injection is used to benefit the third party to get valuable information from a user. Null byte Injection is used to take advantage of avoiding sanity checks on the web framework by adding null byte characters, which are URL encoded and opened by the user. Later on, after the injection process becomes successful enough, it can then go on to help attackers to gain unauthorised access to system files. For example:

- An attacker wants to upload a PHP file like "ABC.php" with "ABC" being the malicious file, but the extension allowed by the system is only a .pdf file.
- So the attacker can reconstruct the file name as "ABC.php%00.pdf " and then uploads the file.
- The application reads the file, sees that it is a .pdf file extension, authenticates it, and throws the end of the string since it contains the null byte. This way, the "ABC.php" gains access to the system file and lets the attackers access a user's system files.

3.2.5 Shift Cipher

This article [17] displays that back in the ancient days how the shift cipher played an important role during critical times. Military establishments have had a huge effect on encryption. The constant need for private and secure connections began in the information age, around the 80s. Since the internet was introduced in the 1960s, it wasn't popular till 1989. The need for increased security of Information has become a must. Encryption helps to hide data by altering it into unreadable code. Leon Battista Alberti came up with an encryption method that involves a cipher disk, which involves sliding mechanical disks that change with several substitution ciphers by encryption. Around the 1500s, Blaise De Vigenere implemented Alberti's polyalphabetic cipher style, which then came to be known as Vigenere Cipher. The process is almost the same as the Caesar cipher, with a slight tweak that involves the key changing during the encryption process. The Vigenere Cipher employs a letter grid and the replacement technique. Vigenere Square or Vigenere Table is the grid utilised, which consists of 26 alphabets offset. The process starts off with a specific hidden word that will be used as an encryption key. The plaintext's first alphabet will be aligned on the x-axis, while the concealed word's first letter will be aligned on the y-axis. The plaintext character is then exchanged for the consonant letter. This process goes on until all the keywords are used. Example:

The plaintext to be encrypted is "Attack". Here the hidden word chosen by the person encrypting is "Lemon".

Since the character size doesn't match the encrypted plaintext, the keyword will get repeated until it matches the plaintext length. The substitution starts with the letter L, the row, and the letter A in the column; the joint letter is then found, L. This process goes on until the whole plaintext is substituted. The end product is

Plaintext: ATTACK.

Keyword: LEMONL.

Ciphertext: LXFOPV.

The decryption process uses a similar method, with the person having to discover the column that correlates with the ciphertext character in the keyword's row.

From the above discussion, it can be observed that the established models fulfilled their purposes. Although the established models had some limitations, there exists room for improvement. So in our proposed model, we have taken in the factors of limitations and tried to implement the methods uniquely. For ex: In a polyalphabetic substitution cipher, multiple keys are generated for a single plaintext character. In the polygraphic substitution cipher, the same situation occurs instead of a single plaintext character group of plaintext characters being taken, and each group must contain the same length. In our proposed model, we plan to execute byte substitution to create more chaos in the algorithm. We have tried implementing the same general concept for byte X-OR by incorporating different mathematical properties. Moving on to Null byte injection, we can see that it is used to breach safety protocols to cause harm to a person's system software. In our model, we made an effort to use Byte injection, keeping the same concept of Null byte injection with a possibility of an additional mathematical equation to create uniqueness. Shift cipher had a brief history from the early days, so in our proposed model, we intend to use a few tweaks to set it apart from the existing models and make it more complex than it already is.

Chapter 4

Work Plan

4.1 Concern with Existing Model

AES came into the limelight when DES and 3DES could not fulfil their purposes. The issue regarding DES was that it ran very slow, specifically [13] related to applications, and the 56-bit key size was the biggest downside. One of the main concerns with 3DES is that it cannot withstand massive cyber attacks [4]. Though AES replaced DES and AES, it also had its fair share of issues with AES being too simple, and the same encryption happens with every box. ECC has a few issues; first of all, it is the size of the encrypted message compared to RSA.[10] Another issue with the RSA algorithm is that the algorithm is quite complex to implement, leading to errors. Diffie hellman is vulnerable to attacks where a user's identity is not established; as a result, a hacker can alter the message's meaning. During key scheduling, Blowfish consumes a huge amount of time. So to sum it all up, keeping all those issues in mind, we will try to enhance the security system in our algorithm so that it can withstand the issues the previous algorithms had to deal with.

4.2 Architecture of the Proposed Model

In this section of the paper, we will be demonstrating the tentative structure of our proposed model. We will be looking into implementing the Collatz conjecture in our algorithm. As we know, cryptography can be of two types; symmetric and asymmetric. Our focus is tilted towards creating a symmetric cryptographic model; however, an alternative asymmetric model can also be implemented. Like any other cryptographic model, our proposed one will also feature an encryption and decryption mechanism.

Before going into the main demonstration, we will look into the framework we have set for our model. As we can see in table 4.1, we have designed our cryptographic algorithm following the basic concept of symmetric encryption. We have implemented a stream cipher scheme using an array of 256 numbers which will be acting as the encryption or decryption key. For our proposed system, we have decided to implement the Diffie-Hellman key exchange protocol followed by the application of the Collatz Conjecture. A detailed description of the architect is in the following sections of this paper.

Note: We have used the example of Alice and Bob to demonstrate communication between them wherever applicable.

Parameters of Proposed Model	**Classification**
Type of Cryptography	Symmetric
Number of keys	1
Key Size	256 number array
Key size in bits	Variable
Key Negotiation	Diffie-Hellman
Cipher Type	Stream

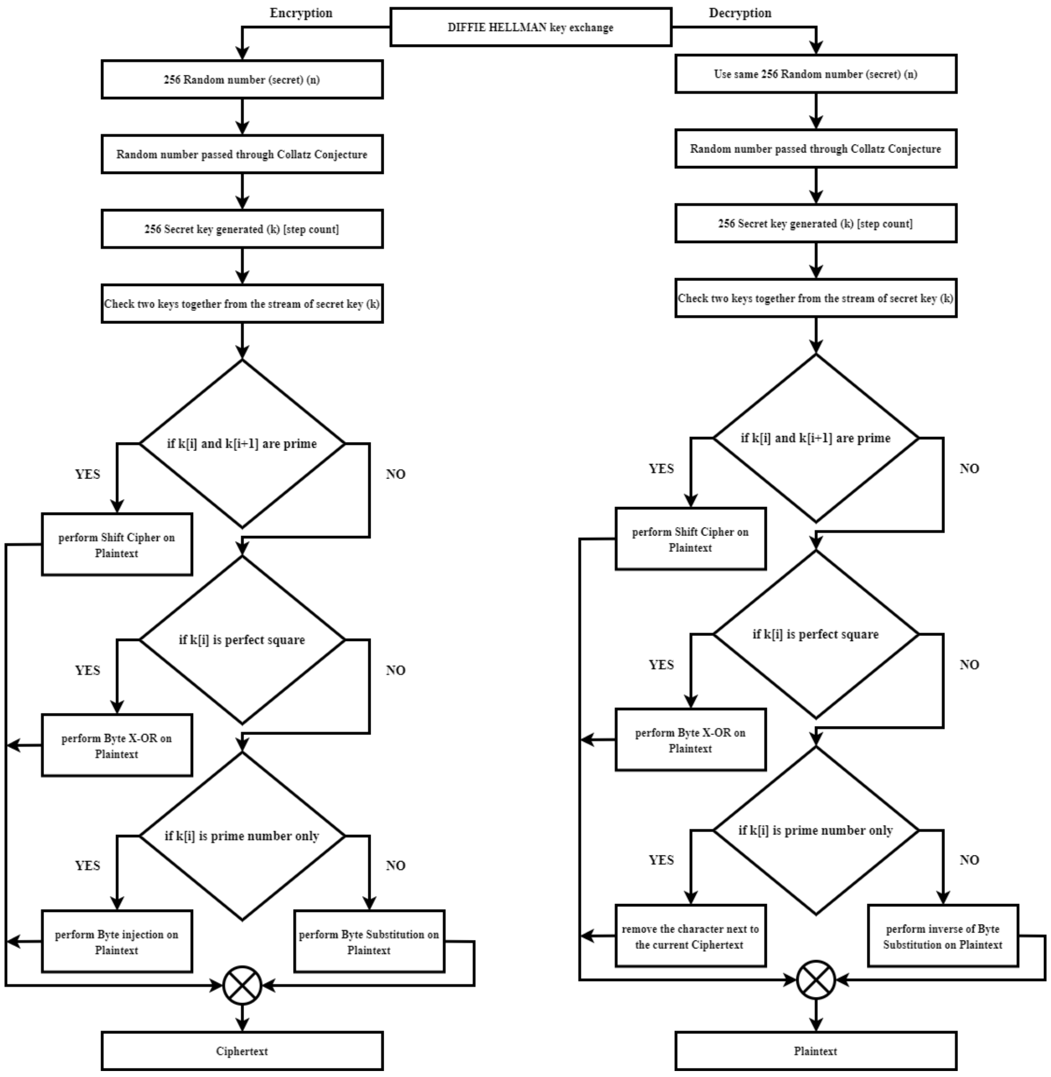
Table 4.1: Parameters set for the Proposed Model

Pseudocode of Algorithm:

```
Message = Plaintext
Secret Random Number (n) → Diffie Hellman key Exchange
Secret key (k) → step count from Collatz Conjecture
Define perfectSquareCheck (k):
   Check for Perfect Square Number
Define isPrimeCheck (k):
   Check for Prime Number
Define conPrimeCheck (k):
   Check for Consecutive Prime Number

Define encryption (Message, k)
   Ciphertext = “ ”
   for i in k
      if conPrimeCheck(i) == true:
         do Shift Cipher on Message
      elif perfectSquareCheck(i) == true:
         do Byte X-OR on Message
      elif isPrimeCheck(i) == true:
         do Byte$_InjectiononMessage$
      else:
         do Byte-Substitution on Message
   return Ciphertext
```

```
Define decryption (Message, k)
    Plaintext = “ ”
    for i in k
        if conPrimeCheck(i) == true:
            do Shift Cipher on Ciphertext
        elif perfectSquareCheck(i)== true:
            do Byte X-OR on Ciphertext
        elif isPrimeCheck(i) == true:
            remove character next to current Ciphertext
        else:
            do inverse Byte Substitution on Ciphertext
        return Plaintext
```



Encryption
DIFFIE HELLMAN key exchange
Decryption
256 Random number (secret) (n)
Random number passed through Collatz Conjecture
256 Secret key generated (k) [step count]
Check two keys together from the stream of secret key (k)
if k[i] and k[i+1] are prime
YES
NO
perform Shift Cipher on Plaintext
if k[i] is perfect square
perform Byte X-OR on Plaintext
if k[i] is prime number only
perform Byte injection on Plaintext
perform Byte Substitution on Plaintext
Ciphertext
Use same 256 Random number (secret) (n)
remove the character next to the current Ciphertext
perform inverse of Byte Substitution on Plaintext
Plaintext

Figure 4.1: Encryption and Decryption process

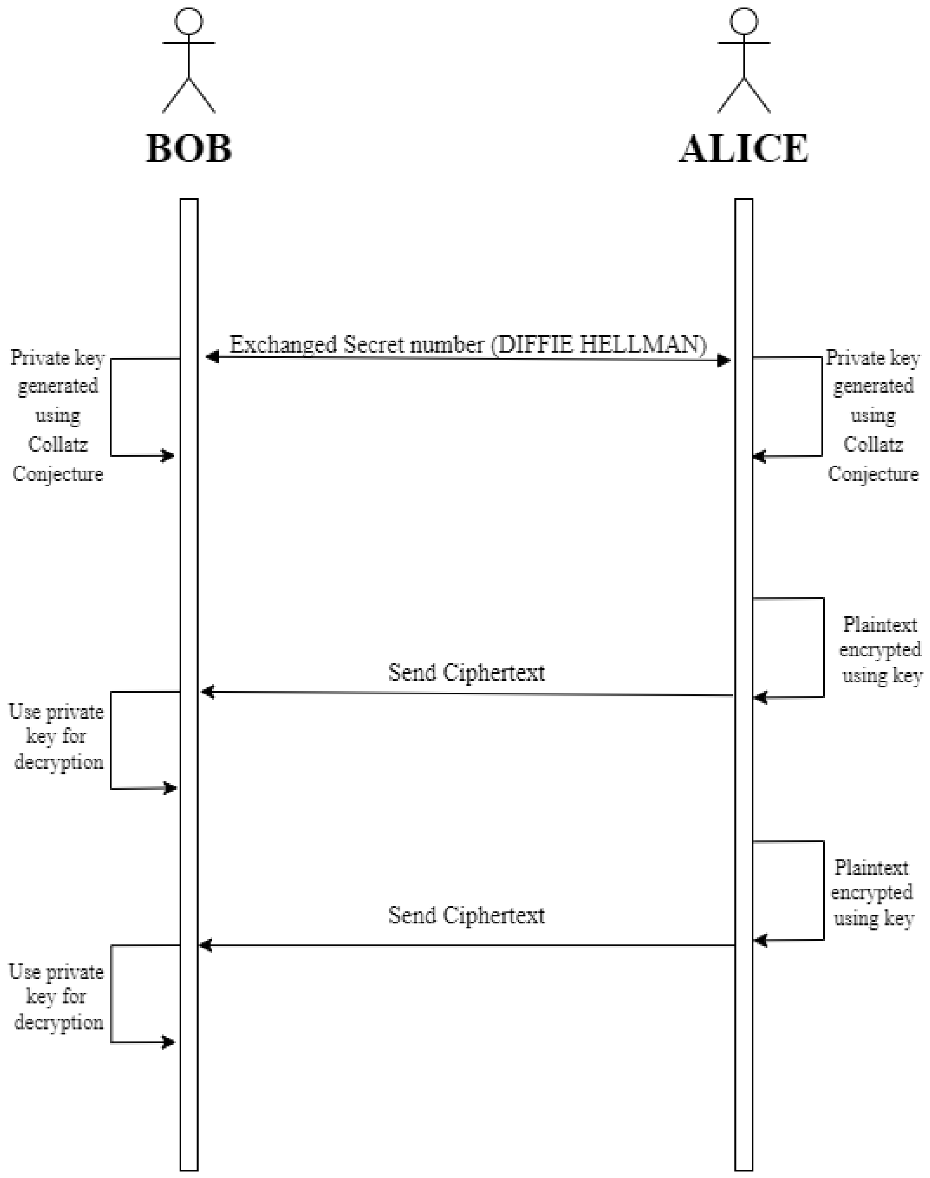

BOB
ALICE
Exchanged Secret number (DIFFIE HELLMAN)
Private key generated using Collatz Conjecture
Private key generated using Collatz Conjecture
Plaintext encrypted using key
Send Ciphertext
Use private key for decryption
Plaintext encrypted using key
Send Ciphertext
Use private key for decryption

Figure 4.2: Communication between Sender and Receiver

4.3 Motivation for Design Choices

As we have shown in the previous section, how we have mapped out our model, in this following section, we will be discussing how we reached our design implication.

Cryptography and concepts of mathematics go hand in hand. Our model uses different operations on plaintext to generate a ciphertext. While creating a blueprint of the model, the main objective was to create randomness in the encryption system, which would eventually lead to a chaotic system generating a ciphertext that would indicate no pattern and would be difficult to decrypt.

To achieve this, we have designed a simple python program that would take an array of random numbers and give us the count for different mathematical properties, which include; prime numbers, perfect squares, consecutive prime numbers, i.e. a prime number followed by another prime number, and numbers which are neither of these, we are referring to as numeral digits.

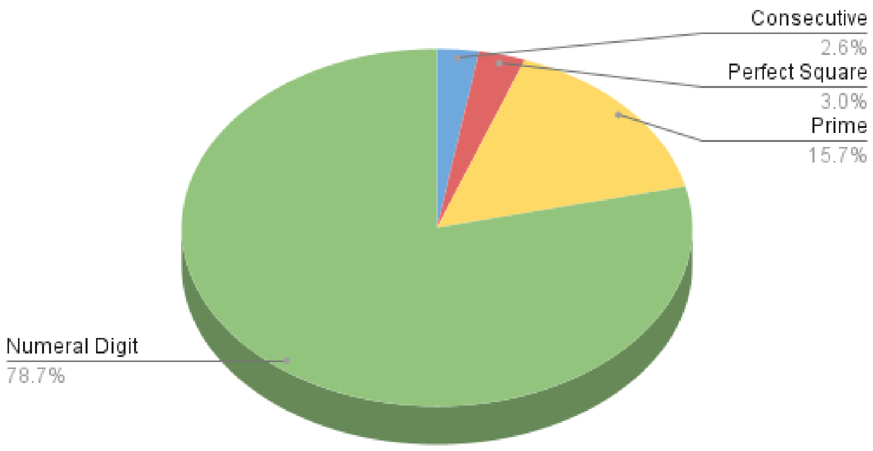


Figure 4.3: The average percentage of each mathematical property for n number of random arrays

As we can see in the figure, the percentage of each mathematical property indicates how often they appear in a random set of numbers. For our model, the key that we will generate will be negotiated through the D-H protocol but will follow a random pattern consisting of numbers with different mathematical entities. To make it into a system that creates complexity, we have mapped our model in such a way that the number which possesses the property with the highest percentage will be used to encrypt/decrypt the plaintext implementing the most robust function and the simplest form of encryption is assigned to the least occurring mathematical attribute.

Encryption and Decryption techniques in increasing order of complexity	**Assigned Mathematical properties (rate of occurrence)**
Shift Cipher	Consecutive Prime (2.6%)
Byte X-OR	Perfect square (3.0%)
Byte Injection	Prime Number (15.7%)
Byte_Substitution	Numeral Digits (78.7%)

Table 4.2: Design Choice Indicator

4.4 Key Negotiation and Generation Technique

4.4.1 Diffie-Hellman

Malcolm Williamson of GCHQ created the Diffie-Hellman key exchange method, which was then independently improved upon by Whitfield Diffie and Martin Hellman. Only a shared secret can be created using the Diffie-Hellman key exchange technique. In most cases, the generated shared secret consists of a common symmetric key. Users can create a shared symmetric key using it. The discrete log problem's computational complexity is what gives Diffie-Hellman its security.

Let p be a prime number and g be a generator, where this p and g are public.

Alice selects a secret value, a

Bob selects a secret value, b

Alice sends g^a mod p to Bob

Bob sends g^b mod p to Alice

Both compute shared secret g^{ab} mod p

$(g^b)^a = g^{ba} = g^{ab}$ mod p

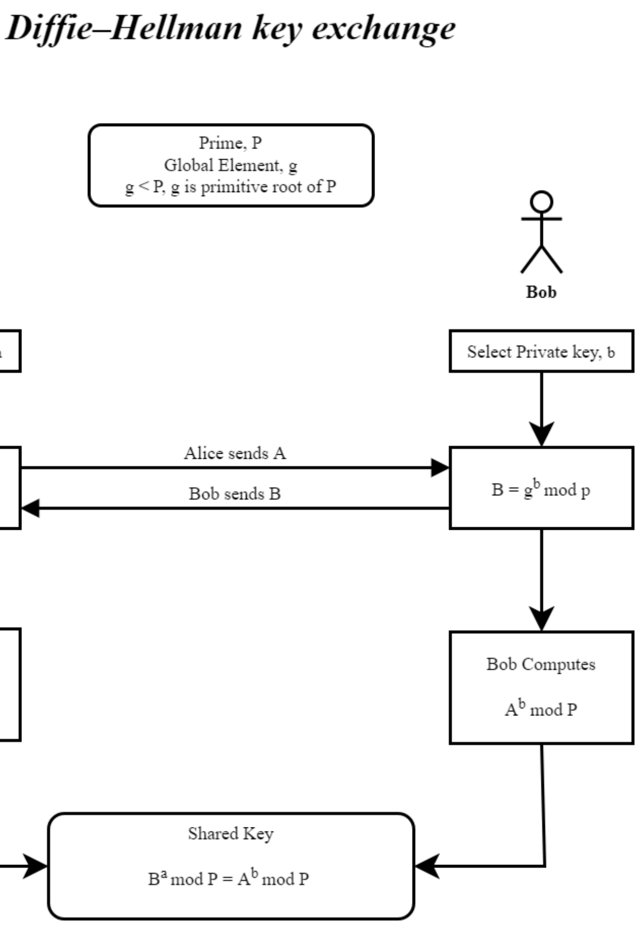


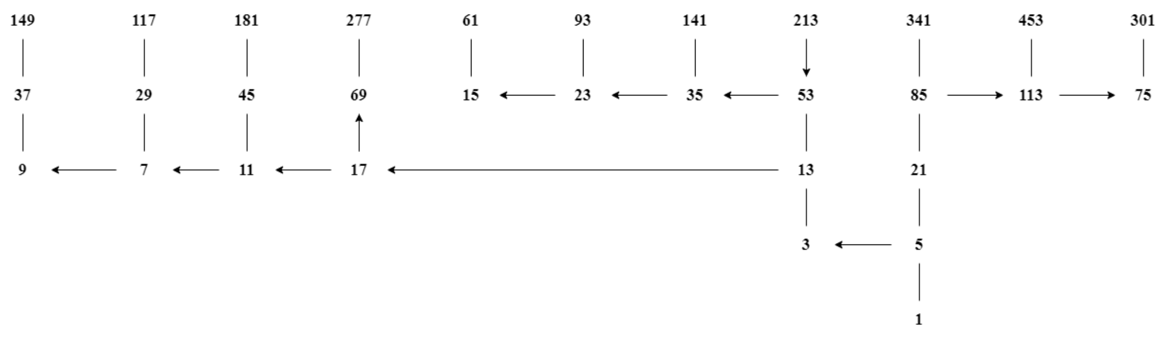
Figure 4.4: Demonstration of Diffie-Hellman

4.4.2 Step Count from Collatz Conjecture

A mystery in the world of mathematics, such that it is called a conjecture and not even a theorem. This is because the simple yet impossible to explain mathematical problem, also referred to as the 3n+1 sequence, is yet to be formally proven. In his abstract [8], Craig Alan Feinstein states that Collatz 3n+1 conjecture must have an infinite number of lines and, therefore, cannot be formally proven. To define the conjecture, we need to understand where the complication arises. If any positive integer is chosen and it is an even number, we divide it by 2; and if it's an odd number, we multiply it by 3 and add 1. This sequence was initially devised in 1937 by German mathematician Lothar Collatz [2]. Following this pattern, the result of any positive integer would be 1. This theorem is repeated until the sequence reaches an infinite loop of 4,2, and 1.

The conjecture is so mesmerising that mathematicians have toyed with it for several years. Some mathematicians have sought to deconstruct or rebuild the Collatz conjecture into even more manageable parts. Due to its unique and simple approach, there have been many applications concerning the conjecture.

Despite the straightforward and feeble outlook of sequence, it has intrigued numerous mathematicians to dig deep into the problem, and in most cases, it has left them with very few positive outcomes. Famous mathematician Paul Erdos rightly stated, "Mathematics is not ready for such problems".



$$C(n) = \begin{cases} n / 2 & \text{if } n = 0 \pmod 2 \\ 3n + 1 & \text{if } n = 1 \pmod 2 \end{cases}$$

Figure 4.5: 3n + 1 Conjecture

4.5 Methods used in the Algorithm

4.5.1 Shift Cipher

The Caesar Cipher method is rated as the most basic and earliest among the encryption systems. This technique is a substitution Cipher, where one letter is substituted by another letter that moves the alphabet down in specific places of the given text. This shifting can be shown such that A can be replaced by B, and B can be replaced by C. Every letter is shifting one bit right. The same procedure is for the left shift for Caesar Cipher.

Shift n in encryption phase:

E (x) = (x+n)

Shift n in decryption phase:

D (x) = (x-n)

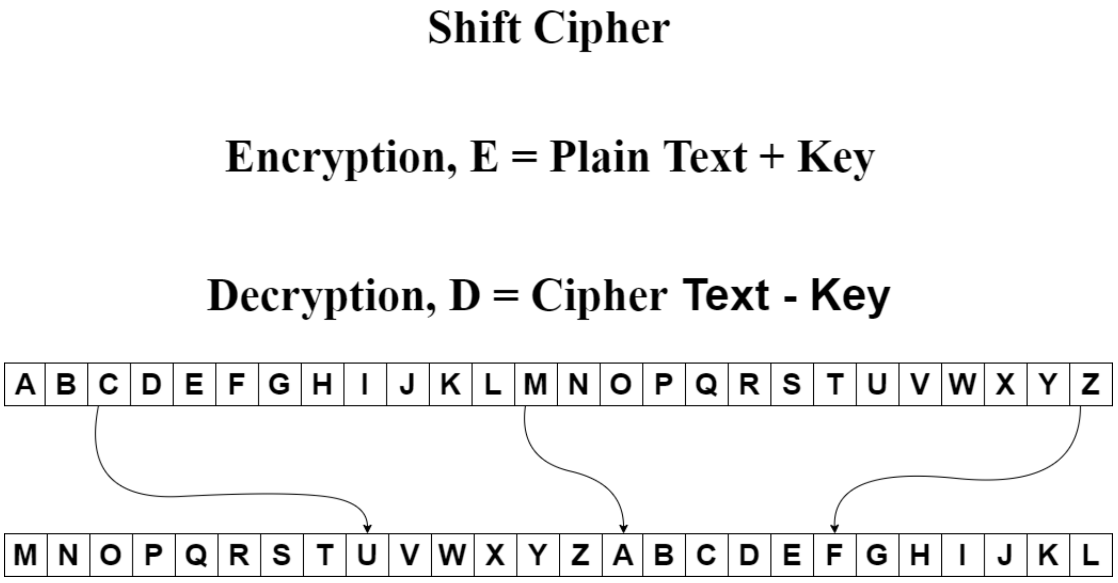

Shift Cipher
Encryption, E = Plain Text + Key
Decryption, D = Cipher Text - Key
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
M N O P Q R S T U V W X Y Z A B C D E F G H I J K L

Figure 4.6: Basics of Shift Cipher

4.5.2 Byte X-OR

Byte X-OR, also known as an exclusive OR operation, deals with a key size that ranges from 1-256 values of a byte. Byte X-OR takes 2-bit patterns that have to be of equal length and performs exclusive OR operations for each corresponding bit.

The output will only be 1 if only 1 of the bits is 1, and the output will be 0 if both of the corresponding values are 0 or 1.

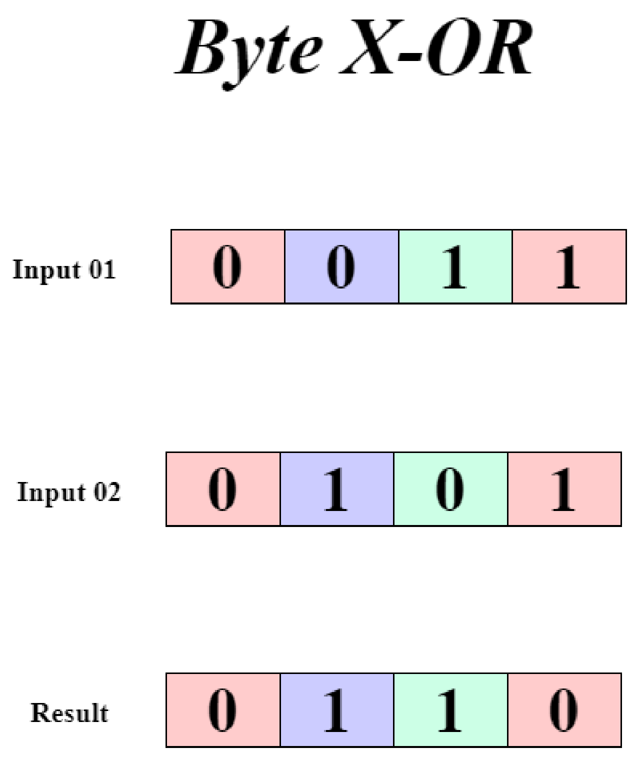


Figure 4.7: Basics of X-OR

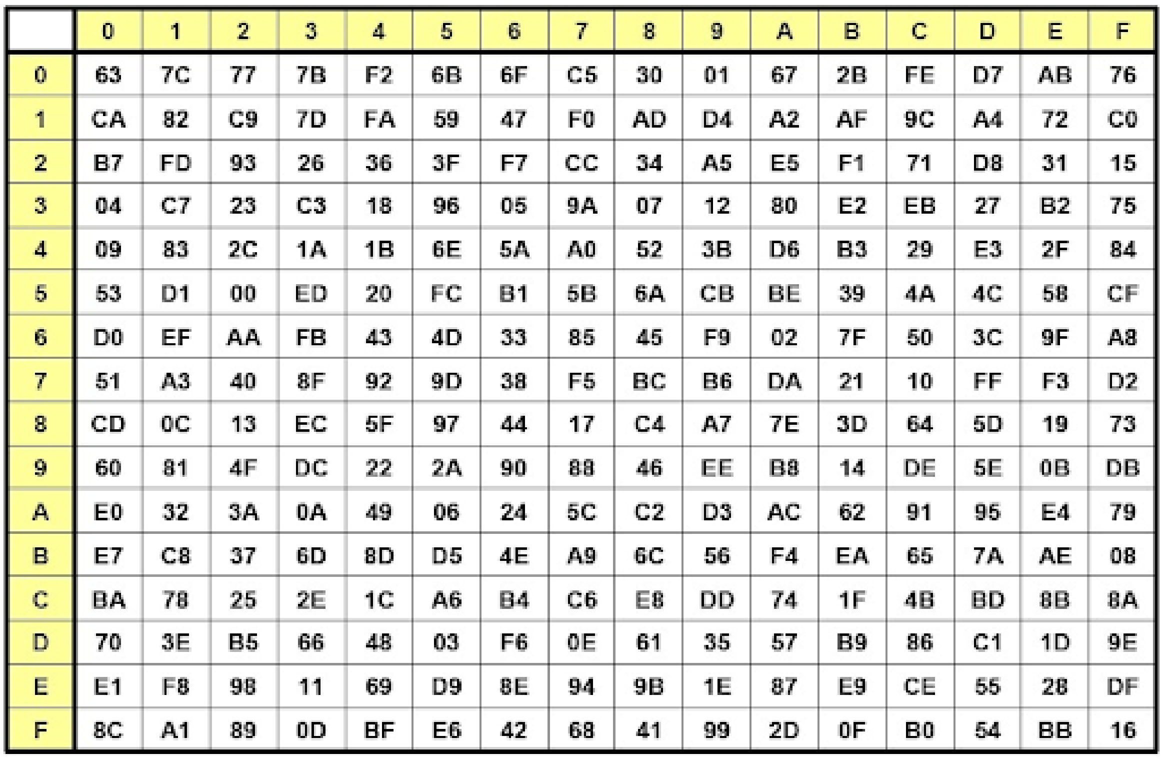
4.5.3 Byte Injection

Byte injection is a process that is usually used to alter the nature of a message. It involves the insertion (injection) of a random character (byte) into the string of information which alters the meaning or value of the original message.

For example, the string "Hello World!" may become "Hell5o Wocrld!G" after three-byte injections.

4.5.4 Byte Substitution

In the AES algorithm, a 16*16 matrix is used, which consists of byte values that are known to be an S-box that contains a permutation of all 256 8-bits. The non-linearity of an S-box is highly dependent on the dispersal of input data using an S-box. For the generation of the S-box, it needs to go through 2 processes. In the first step, a piece-wise linear chaotic map (PWLCM) is used to generate initial S box positions during the post-processing technique; the dispersing property measures the maximum non-linearity that follows a sequence. The second step follows reverse engineering, and the dispersion property is used within the design loop or systematic dispersal of the input substituting sequence. As a result, the controlled randomisation changes the probability distribution of S-box differentials. This leads to substituting S-box positions for output differences reverting to a known input difference.[16]



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Figure 4.8: S-box used for byte substitution in AES[S-BOX]

4.6 Complete Breakdown of Algorithm

Now that we have established the basics of the model, we will be looking into a further detailed demonstration of the encryption and decryption mechanism with simple plain text. For any cryptographic procedure, key creation and negotiation are very crucial steps. We are channelling our focus mostly on the encryption and decryption process and using the Diffie-Hellman key exchange scheme for our secret key encryption.

Theoretical Example

Secret Number Negotiation using Diffie-Hellman

Choosing a random large prime number, p = 79

Generator, g = 3

Alice:

Private key is 8

Public key = $g^{(Alice's\ private\ key)}$ mod p

= 38 mod 79

= 4

Bob:

Private key is 14

Public key = $g^{(Bob's\ private\ key)}$ mod p

= 314 mod 79

= 72

Shared Secret number:

Alice Compute:

Bob's Secret Number = $(Bob's\ public\ key)^{(Alice's\ private\ key)}$ mod p

= 728 mod 79

= 13

Bob Compute:

Alice's Secret Number = $(Alice's\ public\ key)^{(Bob'sprivatekey)}$ mod p

= 414 mod 79

= 13

Secret Number array(n) =[13, 12, 19, 26, 31, 99, 81, 76, 31, 94, 66, 54, 43, 20, 86, 51, 10, 162, 60, 67, 49, 40..]

- For the entire process, the above mechanism will be implemented for creating the Secret number array consisting of 256 numbers.
- After receiving the Secret number array, Alice and Bob will generate the Secret key array using the Collatz Conjecture, for encryption and decryption.
- In our example, the step to reach the 4-2-1 loop is 7 for its corresponding value of 13, which is the initial shared secret number in the array. 7 here is the private key for the cryptographic method. Figure 4.9 below demonstrates the process of key generation.

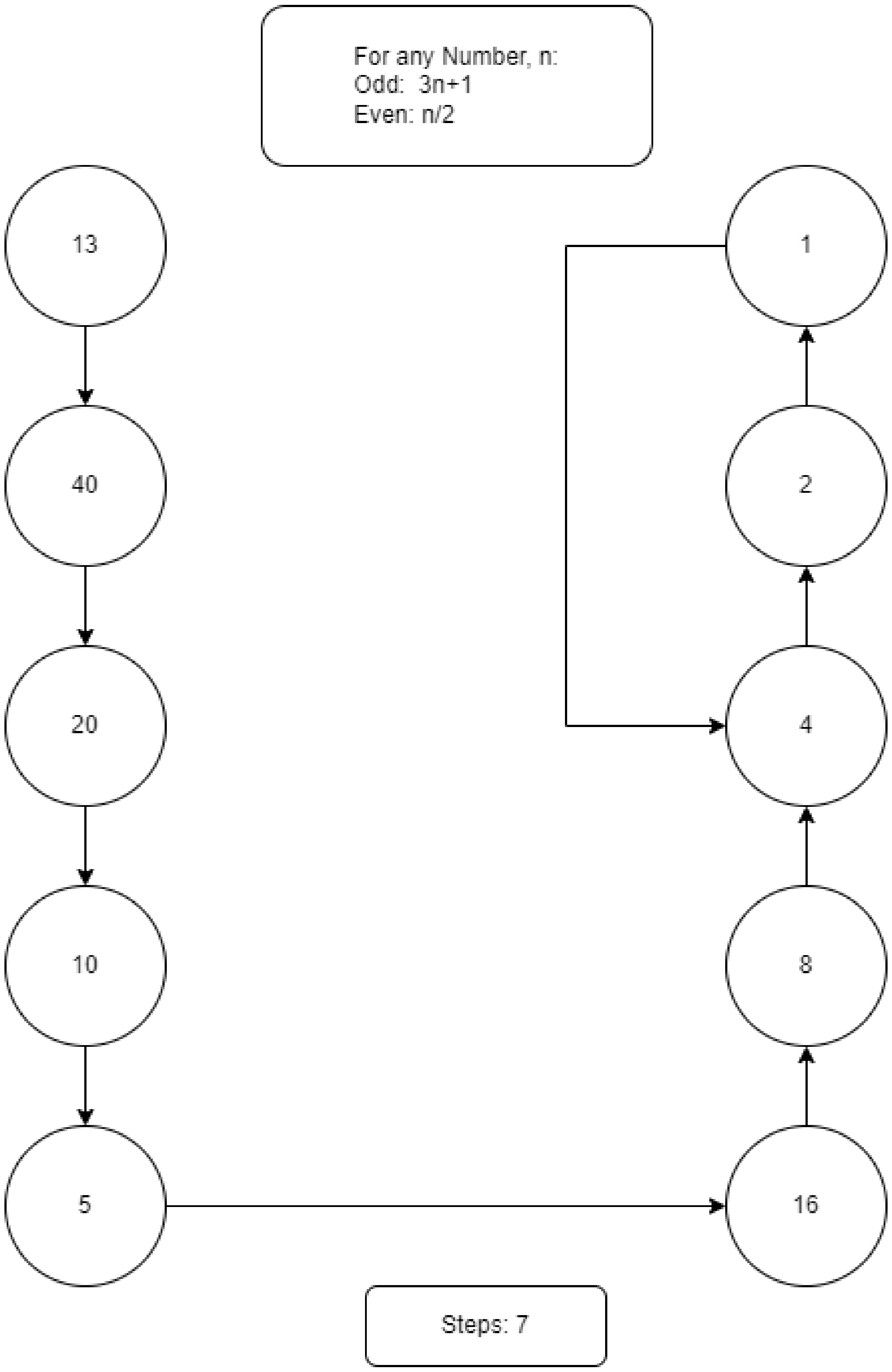


Figure 4.9: Tree demonstration of 3n+1 Conjecture

Plain Text	**Ascii/UTF-8**	**Shared Secret number from array**	**Using Collatz Conjecture to get the step count [key set]**
T	84	13	7
h	104	12	7
i	105	19	18
s	115	26	8
[space]	32	31	104
i	105	99	23
s	115	81	20
[space]	32	76	20
o	111	31	104
u	117	94	103
r	114	66	25
[space]	32	54	110
A	65	43	27
l	108	20	5
g	103	86	28
o	111	51	22
r	114	10	4
i	105	162	105
t	116	60	17
h	104	67	25
m	109	49	22

Table 4.3: Values used in Sample Problem

Table 4.3 demonstrates the first few stages of the working mechanism of the algorithm. Here we have assigned a key to each character in the plain text according to the index number of the key array. For e.g., the first value in the array is 13, and the first character in the plain text is T; thus, we get 13 as its private key for T. Similarly, we assign values from the array to each of the characters in the plain text. Lastly, we can see that we have also assigned a value to each character which indicates the step/hops it took for the key to reach the 4, 2, 1 loop in the conjecture. This step count is the unique factor used for encryption and decryption in this model.

Encryption

Now that we have achieved the step count, n for each character in the plain text, Alice will be applying the previously mentioned methods on the plaintext index after comparing the values of the step count.

Plaintext Message: **This is our Algorithm**

Secret Number array(n)= [13, 12, 19, 26, 31, 99, 81, 76, 31, 94, 66, 54, 43, 20, 86, 51, 10, 62, 61, 49, 66, 40. . . .]

Secret Step count array(key) = [7, 7, 18, 8, 104, 23, 20, 20, 104, 103, 25, 110, 27, 5, 28, 22, 4, 105, 17, 25, 22. . . ..]

Now to encrypt each character in the plaintext, Alice will compare the values of key in pairs to determine their mathematical characteristics

Shift Cipher [key= 7]

Plaintext	**Decimal Value (Plaintext)**	**Decimal Value + key**	**Decimal Value (Ciphertext)**	**Ciphertext**
T	84	84 + 7	91	[

- The first two numbers in the key set are 7 and 7.
- 7 and 7 are double prime numbers; thus, Alice would encrypt the next plaintext value using shift cipher.

Byte Injection [key= 7]

Plaintext	**Decimal Value (Plaintext)**	**Plaintext, (2 * key) + 128**	**Decimal Values (Ciphertext)**	**Ciphertext**
h	104	104, (2*7)+128	104, 142	**hŽ**

- Similarly, Alice will compare the next two values which are 7 and 18. Since the first value is a prime number, Alice will insert a character next to the current index in the plaintext. To get the character, Alice will simply find the Ascii/UTF-8 character using the formula:
  - (2 * key) + 128, where the key is the step count for that particular number.
  - Here, the key is 7; therefore, the character that generates is Ž.

Byte Substitution [key= 18]

Plaintext	Decimal Value (Plaintext)	Decimal Value + key	Decimal Value	Hex Value	S-Box Value	S-Box to Decimal Value	Decimal Value + 128	Decimal Value	Ciphertext
i	105	105 + 18	$(123)_{10}$	$(7B)_{16}$	$(21)_s$	$(33)_{10}$	33 + 128	161	¡

- Since, in the secret key sequence the next key is 18 therefore, for values that are simply neither prime nor a perfect square, Alice will carry out byte-substitution.
  - The plaintext value is added to the key, and its equivalent hex value is generated.
  - Using an S-box and the hex value, Alice will obtain the value which will be used to replace the plaintext.
  - As the value from the S-box is in hexadecimal, its value will be converted to decimal.
  - Lastly, 128 will be added to the value to get the equivalent Ascii/UTF-8 char which will give Alice the ciphertext.
  - The above-mentioned steps are followed each time byte-substitution is carried out.

Byte Substitution [key= 8]

Plaintext	Decimal Value (Plaintext)	Decimal Value + key	Decimal Value	Hex Value	S-Box Value	S-Box to Decimal Value	Decimal Value + 128	Decimal Value	Ciphertext
s	115	115 + 8	$(123)_{10}$	$(7B)_{16}$	$(21)_s$	$(33)_{10}$	33 + 128	161	¡

Byte Substitution [key= 104]

Plaintext	Decimal Value (Plaintext)	Decimal Value + key	Decimal Value	Hex Value	S-Box Value	S-Box to Decimal Value	Decimal Value + 128	Decimal Value	Ciphertext
[space]	32	32 + 104	$(136)_{10}$	$(88)_{16}$	$(C4)_s$	$(196)_{10}$	196 + 128	324	ń

Byte Injection [key= 23]

Plaintext	Decimal Value (Plaintext)	Plaintext, (2 * key) + 128	Decimal Values (Ciphertext)	Ciphertext
i	105	105, (2 * 23) + 128	105, 174	i®

Byte Substitution [key = 20]

Plaintext	Decimal Value (Plaintext)	Decimal Value + key	Decimal Value	Hex Value	S-Box Value	S-Box to Decimal Value	Decimal Value + 128	Decimal Value	Ciphertext
s	115	115 + 20	$(135)_{10}$	$(87)_{16}$	$(17)_s$	$(23)_{10}$	23 + 128	151	—

Byte Substitution [key= 20]

Plaintext	Decimal Value (Plaintext)	Decimal Value + key	Decimal Value	Hex Value	S-Box Value	S-Box to Decimal Value	Decimal Value + 128	Decimal Value	Ciphertext
[space]	32	32 + 20	$(52)_{10}$	$(34)_{16}$	$(18)_s$	$(24)_{10}$	24 + 128	152	˜

Byte Substitution [key= 104]

Plaintext	Decimal Value (Plaintext)	Decimal Value + key	Decimal Value	Hex Value	S-Box Value	S-Box to Decimal Value	Decimal Value + 128	Decimal Value	Ciphertext
o	111	111 + 104	$(215)_{10}$	$(D7)_{16}$	$(0E)_s$	$(14)_{10}$	14 + 128	161	Ž

Byte Injection [key= 103]

Plaintext	Decimal Value (Plaintext)	Plaintext, (2 * key) + 128	Decimal Values (Ciphertext)	Ciphertext
u	117	117, (2 * 103) + 128	117, 334	uŎ

Byte X-OR [key= 25]

Plaintext	Decimal Value (Plaintext)	Decimal Value ⊕ key	Decimal Value (Ciphertext)	Ciphertext
r	114	114 ⊕ 25	107	k

- Since for the above encryption, the corresponding key value is a perfect square(25), thus Alice will carry out byte X-OR between 25 and the corresponding plaintext char Ascii/UTF-8 value, as mentioned in the algorithm.

Byte Substitution [key= 110]

Plaintext	Decimal Value (Plaintext)	Decimal Value + key	Decimal Value	Hex Value	S-Box Value	S-Box to Decimal Value	Decimal Value + 128	Decimal Value	Ciphertext
[space]	32	32 + 110	$(142)_{10}$	$(8E)_{16}$	$(19)_s$	$(25)_{10}$	25 + 128	153	™

Byte Substitution [key= 27]

Plaintext	Decimal Value (Plaintext)	Decimal Value + key	Decimal Value	Hex Value	S-Box Value	S-Box to Decimal Value	Decimal Value + 128	Decimal Value	Ciphertext
A	65	65+ 27	$(92)_{10}$	$(5C)_{16}$	$(4A)_s$	$(74)_{10}$	74 + 128	202	Ê

Byte Injection [key= 5]

Plaintext	Decimal Value (Plaintext)	Plaintext, (2 * key) + 128	Decimal Values (Ciphertext)	Ciphertext
l	108	108, (2 * 5) + 128	108, 138	lŠ

Byte Substitution [key= 28]

Plaintext	Decimal Value (Plaintext)	Decimal Value + key	Decimal Value	Hex Value	S-Box Value	S-Box to Decimal Value	Decimal Value + 128	Decimal Value	Ciphertext
g	103	103 + 28	$(131)_{10}$	$(83)_{16}$	$(EC)_s$	$(236)_{10}$	236 + 128	364	Ŭ

Byte Substitution [key= 22]

Plaintext	Decimal Value (Plaintext)	Decimal Value + key	Decimal Value	Hex Value	S-Box Value	S-Box to Decimal Value	Decimal Value + 128	Decimal Value	Ciphertext
o	111	111 + 22	$(133)_{10}$	$(85)_{16}$	$(97)_s$	$(151)_{10}$	151 + 128	279	ė

Byte Substitution [key= 4]

Plaintext	Decimal Value (Plaintext)	Decimal Value + key	Decimal Value	Hex Value	S-Box Value	S-Box to Decimal Value	Decimal Value + 128	Decimal Value	Ciphertext
r	114	114 + 4	$(118)_{10}$	$(76)_{16}$	$(38)_s$	$(56)_{10}$	56 + 128	184	¸

Byte Substitution [key= 105]

Plaintext	Decimal Value (Plaintext)	Decimal Value + key	Decimal Value	Hex Value	S-Box Value	S-Box to Decimal Value	Decimal Value + 128	Decimal Value	Ciphertext
i	105	105 + 105	$(210)_{10}$	$(D2)_{16}$	$(B5)_s$	$(181)_{10}$	181 + 128	309	ĵ

Byte Injection [key= 17]

Plaintext	Decimal Value (Plaintext)	Plaintext, (2 * key) + 128	Decimal Values (Ciphertext)	Ciphertext
t	116	116, (2 * 17) + 128	116, 162	t

Shift Cipher [key= 7]

Plaintext	Decimal Value (Plaintext)	Decimal Value $\oplus$ key	Decimal Value (Ciphertext)	Ciphertext
h	104	$104 \oplus 25$	113	q

Byte Substitution [key= 22]

Plaintext	Decimal Value (Plaintext)	Decimal Value + key	Decimal Value	Hex Value	S-Box Value	S-Box to Decimal Value	Decimal Value + 128	Decimal Value	Ciphertext
m	109	109 + 22	$(131)_{10}$	$(83)_{16}$	$(EC)_s$	$(236)_{10}$	236 + 128	364	Ŭ

Ciphertext: **[hŽ¡¡ńi®—˜ŽuŎk™ÊlŠŬė¸ĵtqŬ**

Decryption

Secret Random Number array(n) = [5, 12, 19, 26, 31, 99, 81, 76, 31, 94, 66, 54, 43, 20, 86, 51, 10, 62, 61, 49, 66, 40. . . .]

Secret Step count array(key) = [7, 7, 18, 8, 104, 23, 20, 20, 104, 103, 25, 110, 27, 5, 28, 22, 4, 105, 17, 25, 22. . . ..]

Ciphertext: **[hŽ¡¡ńi®—˜ŽuŎk™ÊlŠŬė¸ĵtqŬ**

Now to decrypt each character in the ciphertext.

Shift Cipher [key= 7]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
[	91	$(91\text{-}7)_{10}$	84	T

- Bob will compare the values of n in pairs to determine their mathematical characteristics.
  - The first two numbers in the key set are 7 and 7.
  - 7 and 7 are double prime numbers. Thus Bob would decrypt the next cipher text value using shift cipher.

Byte Injection [key = 7]

Ciphertext	Decimal Values (Ciphertext)	Decimal Value (Plaintext)	Plaintext
hŽ	104 ,142	104	h

- Similarly, Bob will compare the next two values which are 7 and 18, and since the first value is a prime number, Bob will remove the character next to the current index in the ciphertext. In this case, "Ž" will be removed from the ciphertext.

Byte Substitution [key = 18]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - 128-bit	Decimal Value	S-Box Value	Hex Value	Hex Value to Decimal Value	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
¡	**161**	161 - 128	$(33)_{10}$	$(21)_s$	$(7B)_{16}$	$(123)_{10}$	123 - 18	**105**	i

- Since in the secret key sequence the next key is 18, therefore values that are simply neither prime nor a perfect square, Bob will carry out byte-substitution.
  - The ciphertext value is taken, and 128 is subtracted from it, which is then converted to its hex value.
  - Using an S-box and the hex value, Bob will obtain the value which will be used to replace the ciphertext.
  - As the value from the S-box is in hexadecimal, its value will be converted to decimal.
  - Lastly, the key will be deducted from the value obtained from the S-box to get the equivalent Ascii/UTF char which will give Bob the ciphertext.
  - The steps as mentioned earlier are followed each time byte-substitution is carried out.

Byte Substitution [key = 8]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - 128-bit	Decimal Value	S-Box Value	Hex Value	Hex Value to Decimal Value	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
¡	161	161 - 128	$(33)_{10}$	$(21)_s$	$(7B)_{16}$	$(123)_{10}$	123 - 8	115	s

Byte Substitution [key = 104]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - 128-bit	Decimal Value	S-Box Value	Hex Value	Hex Value to Decimal Value	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
ń	324	324 - 128	$(196)_{10}$	$(C4)_s$	$(88)_{16}$	$(136)_{10}$	136 - 104	32	[space]

Byte Injection [key = 23]

Ciphertext	Decimal Values (Ciphertext)	Decimal Value (Plaintext)	Plaintext
i®	105, 174	105	i

Byte Substitution [key = 20]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - 128-bit	Decimal Value	S-Box Value	Hex Value	Hex Value to Decimal Value	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
—	161	161 - 128	$(33)_{10}$	$(21)_s$	$(7B)_{16}$	$(123)_{10}$	123 - 18	115	s

Byte Substitution [key = 20]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - 128-bit	Decimal Value	S-Box Value	Hex Value	Hex Value to Decimal Value	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
˜	324	324 - 128	$(196)_{10}$	$(C4)_s$	$(88)_{16}$	$(136)_{10}$	136 - 104	32	[space]

Byte Substitution [key = 104]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - 128-bit	Decimal Value	S-Box Value	Hex Value	Hex Value to Decimal Value	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
Ž	142	142 - 128	$(14)_{10}$	$(OE)_s$	$(D7)_{16}$	$(215)_{10}$	215 - 104	111	o

Byte Injection [key = 103]

Ciphertext	Decimal Values (Ciphertext)	Decimal Value (Plaintext)	Plaintext
uŎ	117, 334	117	u

Byte XOR [key = 25]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value $\oplus$ key	Decimal Value (Plaintext)	Plaintext
k	107	$107 \oplus 25$	114	r

- Since for the above decryption, the corresponding key value is a perfect square(25), thus Bob will carry out byte X-OR between 25 and the corresponding ciphertext char Ascii/UTF-8 value, as mentioned in the algorithm.

Byte Substitution [key = 110]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - 128-bit	Decimal Value	S-Box Value	Hex Value	Hex Value to Decimal Value	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
™	153	153 - 128	$(25)_{10}$	$(19)_s$	$(8E)_{16}$	$(142)_{10}$	142 - 110	32	[space]

Byte Substitution [key = 27]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - 128-bit	Decimal Value	S-Box Value	Hex Value	Hex Value to Decimal Value	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
Ê	**202**	202 - 128	$(74)_{10}$	$(4A)_s$	$(5C)_{16}$	$(92)_{10}$	92 - 27	**65**	A

Byte Injection [key = 5]

Ciphertext	Decimal Values (Ciphertext)	Decimal Value (Plaintext)	Plaintext
lŠ	108, 138	108	l

Byte Substitution [key = 28]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - 128-bit	Decimal Value	S-Box Value	Hex Value	Hex Value to Decimal Value	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
Ŭ	364	364 - 128	$(236)_{10}$	$(EC)_s$	$(83)_{16}$	$(131)_{10}$	131 - 28	103	g

Byte Substitution [key = 22]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - 128-bit	Decimal Value	S-Box Value	Hex Value	Hex Value to Decimal Value	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
ė	279	279 - 128	$(151)_{10}$	$(97)_s$	$(85)_{16}$	$(133)_{10}$	133 - 22	111	o

Byte Substitution [key = 4]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - 128-bit	Decimal Value	S-Box Value	Hex Value	Hex Value to Decimal Value	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
¸	184	184 - 128	$(56)_{10}$	$(38)_s$	$(76)_{16}$	$(118)_{10}$	118 - 4	114	r

Byte Substitution [key = 105]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - 128-bit	Decimal Value	S-Box Value	Hex Value	Hex Value to Decimal Value	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
ĵ	309	309 - 128	$(181)_{10}$	$(B5)_s$	$(D2)_{16}$	$(210)_{10}$	210 - 105	105	i

Byte Injection [key = 17]

Ciphertext	Decimal Values (Ciphertext)	Decimal Value (Plaintext)	Plaintext
t	116, 162	116	t

Byte XOR [key = 25]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value $\oplus$ key	Decimal Value (Plaintext)	Plaintext
q	113	$113 \oplus 25$	104	h

Byte Substitution [key = 22]

Ciphertext	Decimal Value (Ciphertext)	Decimal Value - 128-bit	Decimal Value	S-Box Value	Hex Value	Hex Value to Decimal Value	Decimal Value - key	Decimal Value (Plaintext)	Plaintext
Ŭ	364	364 - 128	$(236)_{10}$	$(EC)_s$	$(83)_{16}$	$(131)_{10}$	131 - 22	109	m

Plaintext: **This is our Algorithm**

Chapter 5

Results and Analysis

In this section of our research, we will analyse our proposed model by setting parameters such as message size and key size. We will be highlighting the major attributes of our model and the impact it has on the strength and computation of the encryption algorithm.

5.1 Implementation and Computational cost

Now, we will be looking into how our algorithm performs for different specifications to map out a general idea of how it would perform practically. We have used Intel(R) Core(TM) i5-8265U processor with a maximum clock speed of 1.60GHz. Our operating system is Windows 10 Home Single Language version 21H1. We have written all our code in Google Collab Notebook using the python programming language.

To create a general algorithm analysis, we have implemented our code by setting a few limitations for experimental purposes. To explain our algorithm, we have followed the same S-box design used in AES, and used a static dictionary to store the S-box values in terms of item-value. The only drawback is that for large-scale dynamic scenarios, the S-box may not be able to give optimal results. This can be solved by creating a dynamic S-box that optimises the algorithm further.

We can take several approaches to determine the computation cost and complexity. The most feasible one for us with our resources is to find the execution time and python autotime to measure the computational time for our algorithm.

Colab notebook includes a package known as the ipython-autotime, which, once loaded, can determine the execution time for each cell in the notebook. We have written our code for encryption and decryption in different cells along with the other necessary variables and obtained the autotime for each cell. This way, we obtained the average autotime for encryption and decryption after running the code several times with different parameters. Additionally, we also took into account the run-time execution time for each cell to get another perspective of the computational cost.

As mentioned in the previous chapter, the encryption or decryption key would comprise an array with 256 numbers, giving it a variable length in terms of bits. We have taken different lengths of the key, which included 100 and 200 numbers, respectably, to make a comparative analysis and run our algorithm for different text sizes. This allowed us to compare how our algorithm works in terms of speed for different scenarios.

In Table 5.1, figure 5.1 and figure 5.2, we can see that the ipython autotime gives us a better picture of how each parameter is creating a difference in the time it takes to encrypt and decrypt a text for various text sizes. We can observe that, as the message size increases from 100 characters to 1000 characters, the time for encryption and decryption increases. However, the key size does not clearly indicate how it affects the execution time.

Performance in terms of speed is not the only major criterion for any cryptographic algorithm. How it ensures security is also crucial. The security measures of any cryptographic algorithm do not only depend on the integrity it provides to the data but also on how it is designed. Most of the time, a failed cryptographic scheme is often due to the exploitations made with any architectural flaw. We will further discuss how our model creates confusion in the ciphertext, which would act as an indicator of algorithm strength in terms of security.

Message Size (length)	**Key size (array length)**	**Average Execution Time(s)**		**Average ipython-autotime (ms)**	
		Encryption	**Decryption**	**Encryption**	**Decryption**
100	100	0.516	0.495	10.26	13.49
	200	0.520	0.528	9.18	13.18
	256	0.589	0.496	9.03	14.35
500	100	0.523	0.558	16.29	40.23
	200	0.606	0.581	17.18	40.62
	256	0.477	0.516	16.52	39.73
1000	100	0.503	0.512	24.59	71.44
	200	0.550	0.531	25.83	72.33
	256	0.494	0.497	24.4	70.89

Table 5.1: Execution and Autotime for different Message and key Size

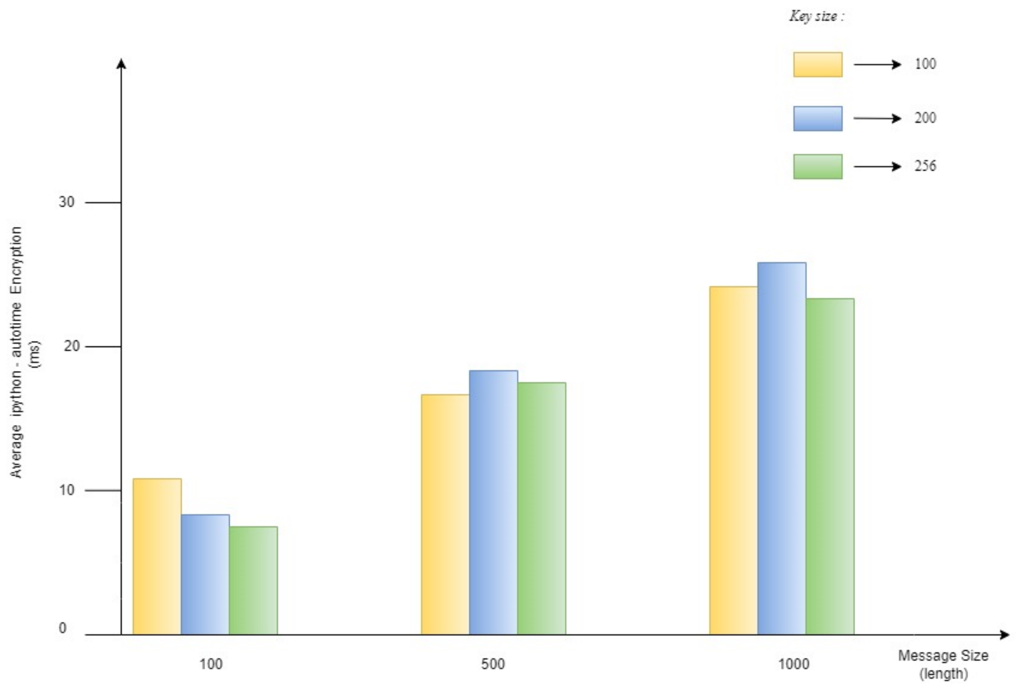


Figure 5.1: Autotime for Encryption

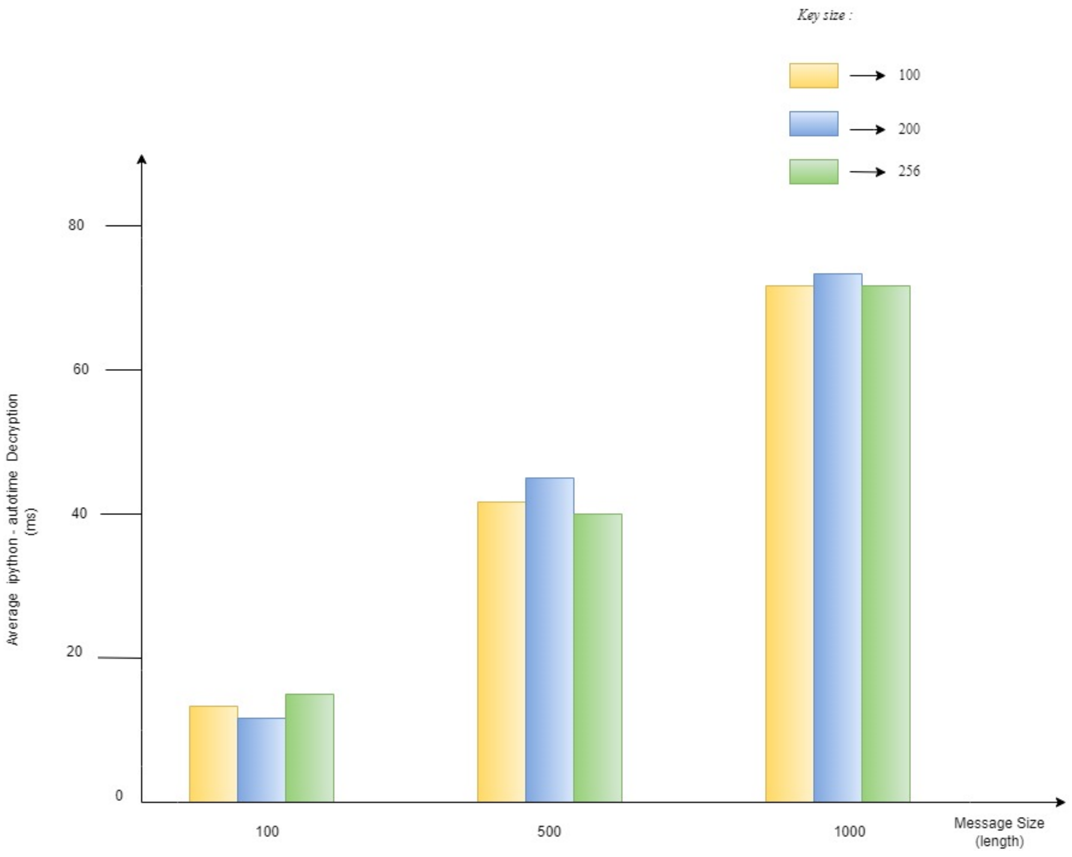


Figure 5.2: Autotime for Decryption

5.2 Randomness and confusion of Ciphertext

In this part, we will attempt to analyse our model further. The analysis of any cryptographic algorithm depends on major factors such as key length and the type of cryptography used, i.e. symmetric or asymmetric. It also depends on how the ciphertext generated creates confusion and diffusion. Confusion essentially means having a ciphertext that is not readable, and diffusion refers to how a ciphertext does not have any recognisable pattern. Any algorithm possessing these two characteristics ensures that it can be used as an encryption method. As mentioned earlier, we have considered the stream cipher technique for our proposed model, which means encryption is done byte to byte or bit to bit. Generally, stream ciphers are faster and are specifically designed for complexity. However, one disadvantage of the stream cipher method is the lack of handling diffusion of the ciphertext. And as explained in the model, we use a big percentage of key, i.e. 78%, to encrypt using byte substitution, which could create a pattern. Below, we try to demonstrate how confusion is maintained for different ciphertexts of the same plaintext.

Plaintext: This is our Algorithm

Ciphertext: [hŽ¡¡ńi®—˜ŽuŎk™ÊlŠŬė¸ĵtqŬ

Ciphertext: ōíďÖŏ®Ŝñ@ĕŜñ]íŝł+}*m

Ciphertext: Ź¤äÒiœŷúkůA¾ŮíĕË$yh·

Ciphertext: Qh—ús¾puĶůśčl)A®Ŭˆä%hÊP

Ciphertext: ÊĴ%*áŮ¡Nl oĒčiêæÁmÊ

The above ciphertexts are achieved from the same plaintext but using different key sets. It is not feasible to calculate or measure the randomness of any text stream other than through statistical tests, but that is not the main focus of our research. However, we can observe the difference between plaintext and ciphertext to determine the randomness. For any given cryptographic system, the more randomised it is, the more confusion it creates, eventually creating a chaotic system which indicates the strength of the model. In the above example, we can observe that for different keys, we get a different cipher text for the same plaintext. Well, that is quite obvious. However, the difference between the ciphertext and its unreadability is what we are aiming at. The strength of symmetric key cryptography heavily depends on its key. Generally, the larger the key size, the tougher it is to guess the key. Moreover, applying brute force to find all the possible keys would be quite impossible. As discussed in our model, we have implemented D-H and collatz conjecture for our key negotiation and generation. These layers determine the key's strength and, eventually, the algorithm's strength.

5.3 Parametric Comparison and Secret Key

Factors	**AES**	**RSA**	**Proposed Model**
Type of Cryptography	Symmetric	Asymmetric	Symmetric
No. of key	1	2	1
Key length	128,192,256	1024, 2048, 3072	Variable
Rounds	10,12,14	1	1
Types of Cipher	Block	Block	Stream
Limitation	Same encryption method for each block	High Computational power required	Only creates confusion

Table 5.2: Comparison between RSA, AES and Proposed Model

Chapter 6

Conclusion

After extensive research and analysis, we have successfully established an algorithm for encrypting data. To restate our core goal, we aimed to create a new cryptographic algorithm incorporating Collatz conjecture. The algorithm has a fast execution speed and successfully creates a random ciphertext, eventually leading to a chaotic system. Apart from a few minor limitations, we have successfully established an algorithm for encrypting data. These limitation can be easily overcome by optimising our algorithm and model for future application. However, we must have definite proof of work to put our work into practice. Expert cryptanalysis requires extensive testing for any new cryptography to be applied in a real-world scenario.

To summarise our report, there is no doubt that there is a high demand for improved data security and its advancements. With new inventions every day in this technology-based world, it will only increase exponentially. Almost all daily work is done over the internet, including banking, shopping, ordering food, ride-sharing, etc. These simple and basic applications involve sensitive data and require security. But, there are not enough safeguards to protect these valuable data being transmitted over the internet. The existing ones are unable to keep up with the increasing demand for safe data communication. Therefore, this research is a stepping stone toward creating a safer online environment using unconventional methods that could withstand the various threats

References

[1] T. J. Murray, "Cryptographic protection of computer-based data files," *MIS Quarterly*, vol. 3, no. 1, pp. 21–28, 1979, ISSN: 02767783. [Online]. Available: http://www.jstor.org/stable/249145 (visited on 09/18/2022).

[2] M. Chamberland, "The collatz chameleon," *Math Horizons*, vol. 14, no. 2, pp. 5–8, 2006, ISSN: 10724117, 19476213. [Online]. Available: http://www.jstor.org/stable/25678649 (visited on 09/18/2022).

[3] P.-J. Kang, S.-K. Lee, and H.-Y. Kim, "Study on the design of mds-m2 twofish cryptographic algorithm adapted to wireless communication," in *2006 8th International Conference Advanced Communication Technology*, vol. 1, 2006, 4 pp.–695. DOI: 10.1109/ICACT.2006.206060.

[4] A. A. Hasib and A. A. M. M. Haque, "A comparative study of the performance and security issues of aes and rsa cryptography," in *2008 Third International Conference on Convergence and Hybrid Information Technology*, vol. 2, 2008, pp. 505–510. DOI: 10.1109/ICCIT.2008.179.

[5] M. Holler, "The zimmermann telegram: How to make use of secrets?" *HOMO OECONOMICUS*, vol. 26, pp. 23–39, Jan. 2009.

[6] A. MacGibbon, "Cyber security: Threats and responses in the information age," Australian Strategic Policy Institute, Tech. Rep., 2009. [Online]. Available: http://www.jstor.org/stable/resrep03941 (visited on 09/18/2022).

[7] N. Li, "Research on diffie-hellman key exchange protocol," in *2010 2nd International Conference on Computer Engineering and Technology*, vol. 4, 2010, pp. V4-634-V4–637. DOI: 10.1109/ICCET.2010.5485276.

[8] C. A. Feinstein, "The collatz 3n+1 conjecture is unprovable," *International Journal of Forecasting*, p. 2, May 2011. DOI: 10.1016/j.ijforecast.2011.08.005.

[9] S. Som and S. Ghosh, "A simple algebraic model based polyalphabetic substitution cipher," *International Journal of Computer Applications*, vol. 39, pp. 53–56, Feb. 2012. DOI: 10.5120/4844-7111.

[10] S. Verma and B. Ojha, "A discussion on elliptic curve cryptography and its applications," *International Journal of Computer Science Issues*, vol. 9, Jan. 2012.

[11] L. Xiaoqin, L. Wei, C. Xiuxin, Z. Xiaoli, and D. Zhengang, "Application of the advanced encryption standard and dm642 in the image transmission system," in *2012 7th International Conference on Computer Science Education (ICCSE)*, 2012, pp. 444–447. DOI: 10.1109/ICCSE.2012.6295110.

[12] T. K. Jishamol and K. Rahimunnisa, “Low power and low area design for advanced encryption standard and fault detection scheme for secret communications,” in *2013 International Conference on Communication and Signal Processing*, 2013, pp. 743–747. DOI: 10.1109/iccsp.2013.6577155.

[13] B. Bhat, A. W. Ali, and A. Gupta, “Des and aes performance evaluation,” in *International Conference on Computing, Communication Automation*, 2015, pp. 887–890. DOI: 10.1109/CCAA.2015.7148500.

[14] P. F. Yeh, “The role of the zimmermann telegram in spurring america’s entry into the first world war,” *American Intelligence Journal*, vol. 32, no. 1, pp. 61–64, 2015, ISSN: 0883072X. [Online]. Available: http://www.jstor.org/stable/26202105 (visited on 09/18/2022).

[15] A. Kumar Paul, “A Highly Secured Mathematical Model for Data Encryption Using Fingerprint Data,” *International Journal on Data Science and Technology*, vol. 2, no. 4, p. 46, 2016.

[16] M. Enriquez, D. W. Garcia, and E. Arboleda, “Enhanced hybrid algorithm of secure and fast chaos-based, aes, rsa and elgamal cryptosystems,” *Indian Journal of Science and Technology*, vol. 10, pp. 1–14, Jun. 2017. DOI: 10.17485/ijst/2017/v10i27/105001.

[17] D. Gautam, C. Agrawal, P. Sharma, M. Mehta, and P. Saini, “An enhanced cipher technique using vigenere and modified caesar cipher,” in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, 2018, pp. 1–9. DOI: 10.1109/ICOEI.2018.8553910.

[18] W. Patterson and C. Winston-Proctor, “Origins of cryptography,” in Nov. 2020, pp. 59–68, ISBN: 9781003052029. DOI: 10.1201/9781003052029-8.

[19] Y. Lin, X. Xia, and J. Yang, “Document encryption method with mechanism of enigma machine,” in *2021 International Conference on Artificial Intelligence, Big Data and Algorithms (CAIBDA)*, 2021, pp. 259–262. DOI: 10.1109/CAIBDA53561.2021.00061.

[20] *M. London(2021). “5 Cybersecurity Trends in 2021.* https://staysafeonline.org/blog/5-cybersecurity-trends-in-2021/, [Online; accessed 2022-09-18], 2021.

[21] S. Sharma, K. N. Patel, and A. Siddhath Jha, “Cryptography using blowfish algorithm,” in *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 2021, pp. 1375–1377. DOI: 10.1109/ICAC3N53548.2021.9725661.

[22] Z. Hu, B. Liu, X. Ren, and Y. Tang, “Analysis and implementation of the enigma machine,” in *2022 International Conference on Big Data, Information and Computer Network (BDICN)*, 2022, pp. 475–480. DOI: 10.1109/BDICN55575.2022.00093.