HASHING & DOUBLE ENCRYPTION TECHNIQUE FOR INFORMATION STORAGE IN CLOUD

Report submitted to the SASTRA Deemed to be University
in partial fulfillment of the requirements
for the award of the degree of

BCSCCS801: **PROJECT WORK**

Submitted by

NAME: SOORYA KUMAR S
(Reg. No.: 222003091, CSE)

July 2022

SASTRA
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION
DEEMED TO BE UNIVERSITY
(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

SRINIVASA RAMANUJAN CENTRE

KUMBAKONAM, TAMIL NADU, INDIA – 612 001

SRINIVASA RAMANUJAN CENTRE

KUMBAKONAM, TAMIL NADU, INDIA – 612 001

Bonafide Certificate

This is to certify that the report titled "**Hashing & Double Encryption Technique for Information Storage in Cloud"** submitted in partial fulfillment of the requirements for the award of the degree of B. Tech. Computer Science and Engineering to the SASTRA Deemed to be University, is a bonafide record of the work done by **Mr. SOORYA KUMAR S** (Reg. No. 222003091) during the final semester of the academic year 2021- 22, in the **Srinivasa Ramanujan Centre** under my supervision. This report has not formed the basis for the award of any degree, diploma, associateship, fellowship or other similar title to any candidate of any University.

Signature of Project Supervisor :
Name with Affiliation : Mrs. **Meenakshi P, AP-II, Department of CSE**
Date :

Project *Vivavoc*e held on _____

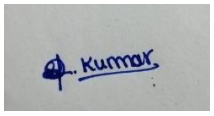
Examiner 1 **Examiner 2**

SRINIVASA RAMANUJAN CENTRE

KUMBAKONAM, TAMIL NADU, INDIA – 612 001

Declaration

I declare that the report titled "**Hashing & Double Encryption Technique for Information Storage in Cloud"** submitted by me is an original work done by me under the guidance of Mrs. **Meenakshi P, AP-II, Srinivasa Ramanujan Centre, SASTRA Deemed to be University** during the final semester of the academic year 2021-22, in the **Srinivasa Ramanujan Centre**. The work is original and whether I have used materials from other sources, I have given due credit and cited them in the text of the report. This report has not formed the basis for the award of any degree, diploma, associate-ship, fellowship or other similar title to any candidate of any University.

Signature of the candidate(s) : 

Name of the candidate(s) : **Soorya Kumar S**

Date :

ACKNOWLEDGEMENT

I pay our sincere obeisance to the God Almighty for his grace and infinite mercy and for showing on us his choicest blessings.

I would like to express our thanks to our Chancellor **Prof. R. Sethuraman, Vice Chancellor Dr. S. Vaidhyasubramaniam** and Registrar **Dr. R. Chandramouli** for having given us an opportunity to be a student of this esteemed institution.

I express our deepest thanks **to Dr. V. Ramaswamy,** Dean and **Dr. A. Alli Rani, Associate Dean,** Srinivasa Ramanujan Centre for their constant support and suggestions when required without any reservations.

I express our gratitude to **HOD incharge Dr. S. Meganathan, ACP, Head of Department, Computer Science Engineering**, Srinivasa Ramanujan Centre for his constant support and valuable suggestion for the completion of the project.

I exhibit our pleasure in expressing our thanks to Mrs. **Meenakshi P, AP-II, Department of CSE**, our guide for her ever-encouraging spirit and meticulous guidance for the completion of the project.

I would like to place on record the benevolent approach and pain taking efforts of guidance and correction of **Dr. J. Sangeetha, SAP, Department of CSE** and **Dr. R. Thanuja, AP-II, Department of CSE**, the project coordinators and all department staff to whom we owe our hearty thanks for ever.

Without the support of our parents and friends this project would never have become reality.
I dedicate this work to our well-wishers, with love and affection.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES:

ABBREVIATIONS

ECC – Elliptical Curve Cryptography.

IAAS - Infrastructure as a Service

PAAS - Platform as a Service

SAAS - Software as a Service

DOS – Denial of Services

SHA – Secure Hash Algorithm

DB – Database

SMTP – Simple Mail Transfer Protocol

AWS – Amazon Web Services

ABSTRACT

Information storage in the cloud is very popular in recent days. The Cloud platform provides various services to its users and one of the important zones of service is the security of the information stored. This project proposes an enhanced approach of a complex Hybrid encryption system involving a Symmetric and an Asymmetric key encryption technique to secure the information on the cloud platform from potential attacks like encryption failure, man-in-middle attack and Insider attack etc along with a hashing technique for the key. The proposed hybrid encryption system involves a newly designed Symmetric key technique "Dehex Algorithm" which encrypts the information with a random key generated from the input size. The Dehex algorithm is followed by the ECC algorithm which is an Asymmetric encryption method. The output formed will be double encrypted information, which then reaches the cloud storage. The keys used for encryption need to be securely stored and so the keys are subjected to the SHA hashing algorithm. Hence this new approach can be one of the ways which can be used for securing information stored in the cloud.

Key Words: Dehex algorithm, Elliptical Curve Cryptography (ECC), SHA, SMTP Protocol, Cloud, data transfer, security.

CHAPTER 1

SUMMARY OF BASE PAPER

Title: A new lightweight cryptographic algorithm for enhancing data security in cloud computing

Journal Name: Elsevier

Publisher: Fursan Thabit, Sharaf Alhomdy, Abdulrazzaq H.A. Al-Ahdal, Prof Dr Sudhir Jagtapa

Year: 2021

Indexed In: Science direct

The Base-Paper Outline:

There has been massive growth of data and information in this techo-world. Data plays a major role in each of our human lives. As the data grows larger, the risk factors associated with it also increase. To secure the data, to prevent it from attackers, and store them safely for later access by the user this paper proposed a new lightweight cryptographic algorithm for enhancing data security and stored in the cloud. The algorithm is a 16 bytes block cipher known as Feistel cipher and followed by some logical operations such as XOR, Shift and swap operations using Shannon's diffusion and confusion concepts. To make them more secure, the resultant encrypted content is stored in the cloud. The cloud adds more security to the user data and decreases the possibility of creeping the information out.

Contribution:

The Base paper includes:

- It uses a 16-byte block cipher called the Feistel Cipher as first encryption.
- The algorithm uses logical operations such as XOR, XNOR, Shift, Swap.
- It is then followed by Shannon's diffusion and confusion process.
- It proposes a strong level of security and enhancement in the Cipher execution process.
- It is then compared with existing algorithms such as DES and so on.

The Proposed Algorithm:

- It is designed to work with Shannon and Fiestal algorithms

- The existing algorithms takes more round of encryption and deploy them into the cloud. But the proposed model takes 4 to 5 round of encryption and deploy them into the cloud.
- Decryption process is just the reverse of the encryption process.
- It uses logical and mathematical concepts to encrypt the data.

Advantages and Disadvantages:

Advantages:

- The output is highly secured and difficult to crack since there involves more mathematical calculations and logical concepts.
- The Number of rounds of encryption and decryption will be less.
- The time complexity for the proposed algorithm is said to be considerable.

Disadvantages:

- Fails to explain about the process of key management.
- The size of the key must be equal to the block size.

CHAPTER 2

INTRODUCTION

Communication has been a lot in today's world and yet it grows more and more in the future. There exists a lot of information in this world which has some purpose to do. As the information grows more and more, the task of saving them securely is also very high. They need to be maintained with special care so as to satisfy the needs in the future. Till then, the information must be stored in local (or) online directory such as cloud until the information is taken by the user. Storing the information safely is the utmost concern.

The most important condition in communication of data and messages in security. Needing security is a tedious task. There exist many false people called Hackers (or) attackers whose job is to seek (or) take information without their knowledge. There may be a possibility of information getting leaked. Since the path cannot be secured as it is huge, the message is encrypted and stored which has been done by the users for the past few years.

As information and technology grows higher and higher, the risk also grows higher. Then the intruders (or) Hackers also develop themselves to some sort of new ways to get the information from the other's user. So new algorithms must be developed to ensure that avoiding the same algorithms which would be easy for the intruders to peek through.

Even though some algorithms that are existing are strong, adding new algorithms and using them may develop some new type of encryption which would create some toughness of breaking it. At present days, using double encryption is more popular as it creates more security than using a single algorithm.

Thus, protecting the information would be fair enough to make sure that it is safe. It is very difficult to secure the path as it is huge. As we create, it is difficult to maintain the security as updating them requires a lot of work.

To maintain privacy over the unsecured network, the messages are encrypted and sent to the receiver. Messages are modified first, then sent to the receiver and remodified later to see the original message.

CHAPTER 3

METHODOLOGIES

Information Security: The process of securing the information from various attacks and protecting it from various threats is called information security.

Cryptography: The branch of science which is used to hide the sensitive information through some of the existing cryptographic algorithms. These proposed cryptographic algorithms are used to change the message into some irregular format called an encrypted message and the process of doing so is called an encryption. This process will allow only authorized persons to enter and will now allow unauthorized persons to enter.

Original information: It is also known as a plain text (or) user's original message. This is used as input for the cryptographic algorithms.

Cryptographic Key: An element which provides encryption to the user's data based on some functions based on the cryptographic algorithms. The input and the key are the important parameters.

Encrypted information: The output of the encryption process where it is obtained by taking the user's message and key as the parameters. The functions of encryption depend upon the algorithms that users use.

Encryption: The process of obtaining the encrypted message is called as encryption

Decryption: The process of obtaining the original message from the encrypted message is called decryption.

Key Management: It is a process of putting some security in the cryptographic keys. Key plays a major role in both encryption and decryption process. Key also ensures safe transfer of data through the internet. With the key, only valid users can know about the key information and the message that is shared. The only task of the user is that the key must be transferred to the receiver safely for the decryption process. Key management depends upon the used cryptographic algorithms and level of security.

Authentication: It is the action done to ensure the identity of a user. i.e, to make sure that the user is valid.

Integrity: It helps to ensure that only authorized persons are allowed to access and change the information. It also ensures that message are received by authorized persons only.

Availability: Contents are available to authorized users at any time when they require.

Access Control: Prevent the use of unauthorized access. It provides the feature that the resources are available for access at the target computers.

Privacy: User has rights to taken control of the information.
Authorization: Ability to access the resources by the receiver through mail sharing and cloud.

Types of Cryptographic algorithms:

Based on key type:

- **Symmetric Cryptographic algorithm:** The algorithm is said to be symmetric if the same key is used for both encryption and decryption. The inverse of the key is used to decrypt the encrypted data. Example: DES, AES.
- **Asymmetric Cryptographic algorithm:** The algorithm is said to be asymmetric if the key used for encryption and decryption are different. These pairs of keys are called public key and private key. Example: RSA, ECC.

Based on the encrypting length:

- **Block Cipher:** It is a technique used to encrypt the information on specific block size. The encrypted information is divided into blocks of specific size. This encryption technique is used to encrypt the specified block size. Padding is needed if the block size is incomplete. Sometimes padding may cause the file size.
- **Stream Cipher:** It is a technique by which each byte of information is encrypted using cryptographic algorithms. No padding option is required in this case. There will be no difference in terms of size in the original and encrypted message.

Based on Text Conversion:

- **Substitution algorithm:** It replaces (or) substitutes the plain text with alternate contents by performing some defined functions with the key. Example: Poly Monoalphabetic Ciphers.
- **Transposition algorithm:** It changes the position of the original message which in turn produces a distorted text and is considered as an encrypted text. The contents in the message are not altered but only the positions of the text are changed.

Cloud: A platform which contains some features such as storage, online applications and computing data in a server. The cloud charges using its resources for its speed usage and efficient usage, memory required and data processing, security.

Cloud processing: It is a process of connecting a remote server to a host to process (or) store data that is required. It provides various services such as:

- Iaas (Infrastructure as a Service)
- Paas (Platform as a Service)
- Saas (Software as a Service)

Types of Cloud:

- **Private Cloud.**
- **Public Cloud.**
- **Hybrid Cloud.**
- **Community Cloud.**

- **Private Cloud:** It is designed to satisfy their small users such as single users. These types of clouds are highly secured and its infrastructure is only known to the user only. Sharing option is disabled in this cloud. The user (or) the enterprise must take care of the cloud. Each individual is responsible for the storage of data.
- **Public Cloud:** It is designed to satisfy large scale industries for its commercial use. Data can be shared and computed among the public and its database settings are in public. This cloud is publicly visible and less protected than private clouds.
- **Hybrid Cloud:** It is defined as the combination of both public and private. The advantage of using hybrid cloud is low cost. It is available to internal users. The disadvantage of using hybrid cloud is less secure and sometimes issues.
- **Community Cloud:** A cloud platform which is cost effective, scalable and flexible. It allows several companies to work on the same platform, since they have similar needs and same concerns. Community cloud is also known as **"Hybrid form of Private Cloud"**.

Services offered by Cloud:

- **Iaas (Infrastructure as a Service)** - It provides basic facilities such as Storage, Computation, networking resources and so on. It provides services based on "Pay-as-you-Service". Some of the platforms include Amazon S3, Amazon EC2, MYSQL instance and so on.
- **Paas (Platform as a Service)** - A platform where users can develop and deploy their applications. A third-party provider provides hardware and software tools over the internet. This reduces the burden of management for the customer. It provides a restricted environment like Iaas. The tools in Paas are simple and convenient to use. Some of the examples include Google App Engine and so on.
- **Saas (Software as a Service)** - It delivers applications over the internet. There is no need for the user to download to the local directory and then install it. Users can access the applications through the internet. The maintenance of the software will be done automatically so as to enable the users to use the software easily. Some of the examples include YouTube, Google Docs, Netflix and so on.

Data Attacks: It is a process of changing the original information by unwanted users (or) non valid users. It is changed when stored in a database (or) during the communication between two users. It is done such that the attacker may sometimes act as a valid user and misuse the information.

Attacks on the information may lead to:

- Misuse of data.
- Information leak.
- Loss of important (or) sensitive information.
- Change in the original information.

Some of the data attacks include:

- Man-in-the-Middle Attack,
- Insider Attack.
- Malware Attack.
- Denial of Service Attack. (DOS)
- SQL Injection Attack.

Types of Attacks:

- **Active Attack:** If the original information is subjected to change, then the attack is known as active attack. It is easy to detect the changes by the user. The main motive of this attack is to share false information.
- **Passive Attack:** If the original information is not subjected to any of the changes, then the attack is called a passive attack. This type of attack is done to steal the information without the permission of users and very difficult to find if there were any changes.

Table 1: Difference between private and public cloud.

Private Cloud	**Public Cloud**
Designed for small users	Designed for large scale industries
Highly Secured	Not highly secured as that of a private cloud.
Sharing option is disabled.	Sharing option is enabled.
Not publicly visible	Publicly visible.

Table 2: Difference between symmetric and asymmetric key cryptosystem.

Symmetric Key Cryptosystem	**Asymmetric Key Cryptosystem**
Same key is used for both the encryption and decryption process.	Multiple keys are used for both encryption and decryption.
Encryption process is fast	Encryption process is slow
It is used when transferring large amount of data	It is used to Transfer small amount of data
EG: AES, DES	EG: RSA, ECC

Table 3: Difference between stream and block cipher.

Stream Cipher	**Block Cipher**
Each byte of information is encrypted	Encryption occurs based on a specific block size that is specified by the user.
Padding is not required; file size remains same	Padding is required, file size may increase.
Faster when compared with block cipher	Slow when compared with stream cipher
It uses the concept of confusion	It uses the concept of both confusion and diffusion.

Table 4: Difference between substitution and transposition cryptographic algorithms.

Substitution algorithms	**Transposition algorithms**
It replaces the plain text with alternative contents.	It changes the position of the text, which in turn produces distorted text.
The contents in the messages are changed based on defined cryptographic algorithms.	The contents in the messages are not changed, only the positions are changed.
EG: Caesar Cipher	EG: Rail Fence Cipher

Table 5: Difference between active and passive attacks.

Active Attacks	**Passive Attacks**
If the messages are changed due to some attack, then it is termed as active attacks.	If the messages are not changed, then it is called as passive attacks.
If an attack is done, it can be found easily	If an attack is done, it cannot be found easily.
The main motive of this attack is to send the false information	The main motive of this attack is to steal the information without the permission of the user.

CHAPTER 4

THE DESIGN APPROACH

The growth of information is rapidly increasing. The rate of production of data at each rate is increasing a lot. Since information is huge, the task of securing them also remains high. It is hectic for the designer to provide security to each data and store them where there is no threat of attack. In today's world, each of the databases has the threat of being attacked. Even though the database is secured, there would be some minor disadvantage in the database and with that point, hackers peek through the database and see the information.

As data can be vulnerable to attacks, the information has to be secured. Even though it is hectic, this is considered a better way so that the contents are not seen by others. There are various cryptographic algorithms which provide the concept of security. New algorithms have to be developed so that the chances of attack will be minimized. As hackers try seeking out through new methods, developers must also develop new methods so as to keep the contents secured.

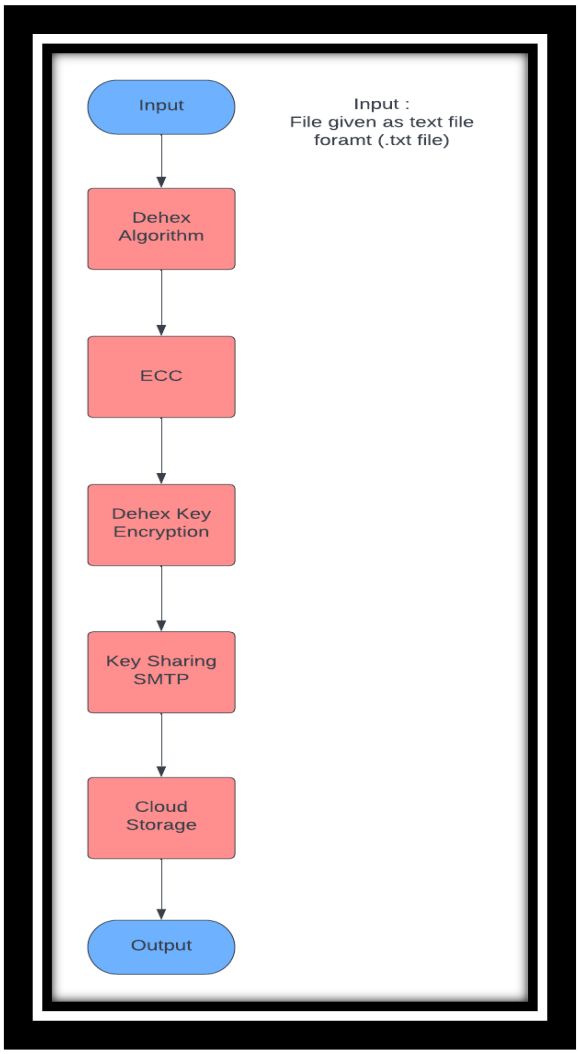


Fig 1: Architecture diagram of Encryption Model.

4.1 The proposed algorithm works in the following approach:

- Dehex Algorithm,
- ECC,
- Dehex key encryption,
- Key sharing,
- Cloud Storage,
- Data Decryption.

4.2 Working of Encryption module:

The algorithm requires an input to be started. The input is given as a text file format (.txt file). The first algorithm is called the “Dehex Algorithm”. This algorithm is considered the first level of encryption. The output of the dehex algorithm is given to the second level of encryption called “ECC” (Elliptical Curve Cryptography. The output from ECC will be doubly encrypted.

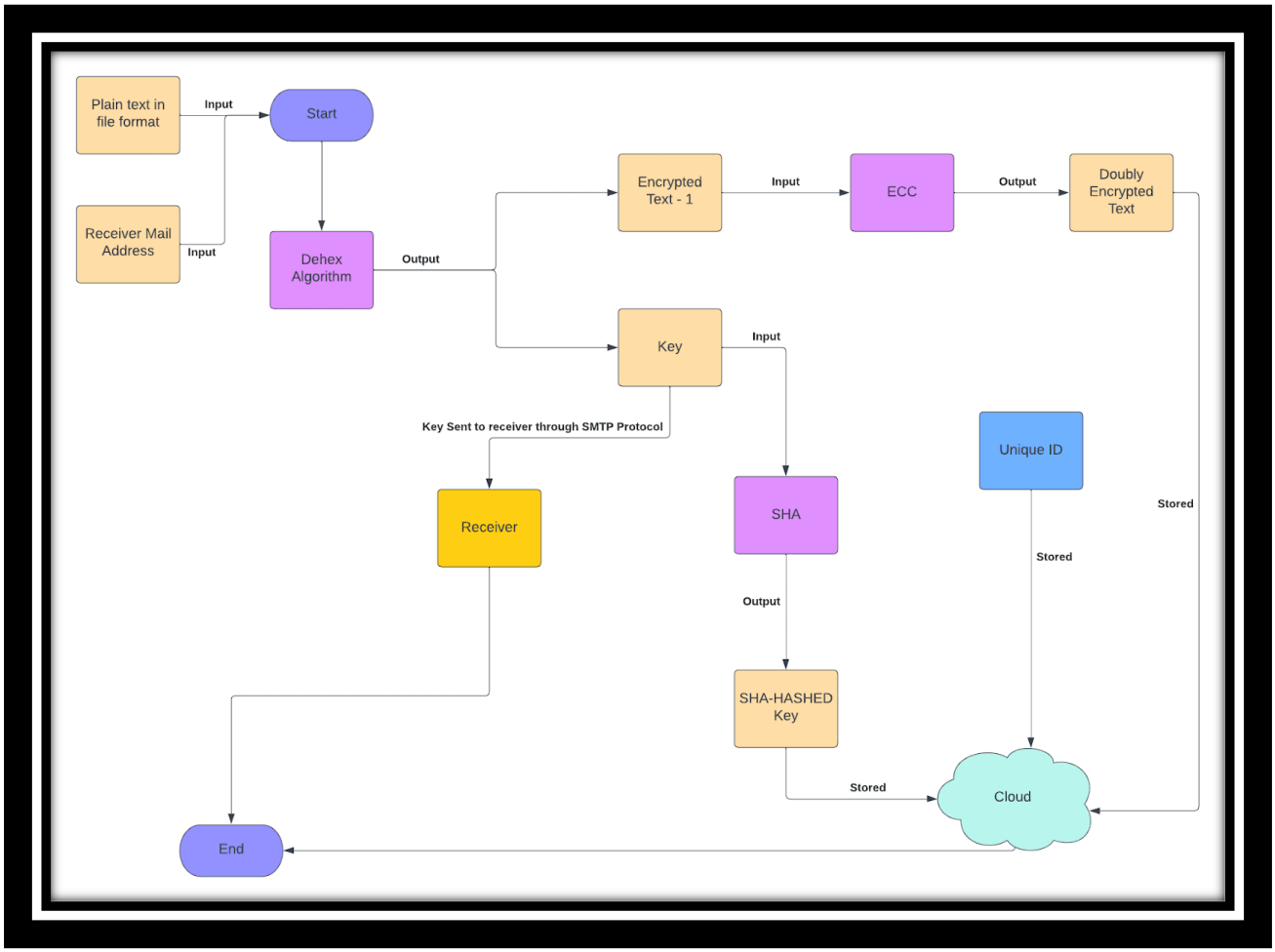


Fig 2: Work flow diagram of Encryption module

The key which is generated from Dehex algorithm is subjected to SHA algorithm so that key won’t be visible to others. The main advantage of using SHA is that it cannot be decrypted.

The cloud connection is established from the python shell. An unique id is created at random to identify the contents that are stored in the cloud. Then the doubly encrypted text along with the hashed key is stored in the cloud.

Then the key file along with the unique id is sent to the receiver through SMTP (Simple Mail Transfer Protocol). This mail is sent for the decryption purpose.

4.3 Working of Decryption module:

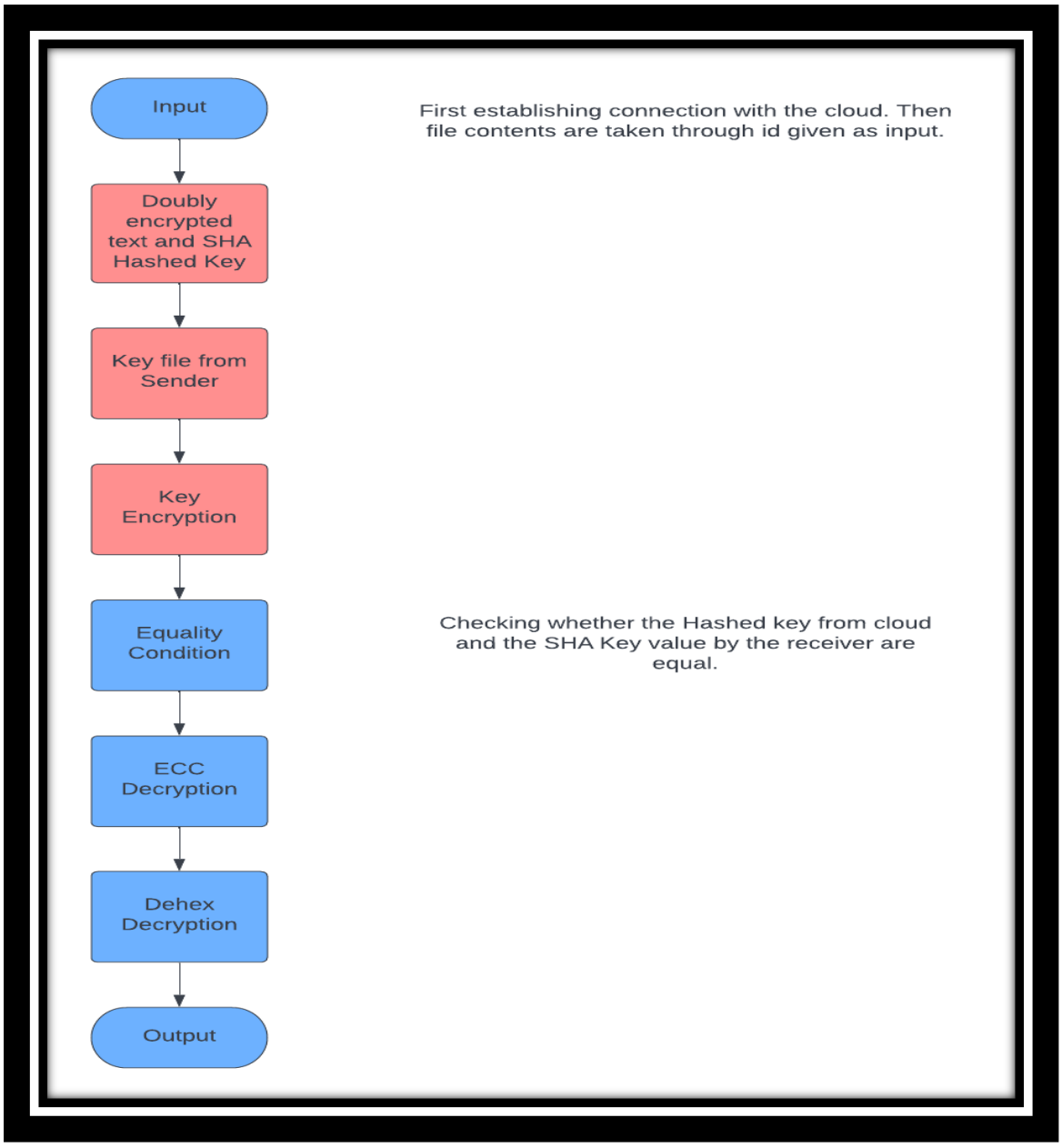


Fig 3: Architecture diagram of Decryption module

First, Cloud connection is established between the python terminal. Then the doubly encrypted information along with key hashed value are taken. The contents from the cloud is

taken through the unique id. Then the key file which is sent through SMTP protocol (mail) is taken and subjected to SHA again.

If the SHA Value taken from the cloud is equal to SHA value which is obtained from the SMTP protocol, then the decryption starts. Else the decryption process would not occur since key mismatch. This is done so that only authorized users must have access to the message sent.

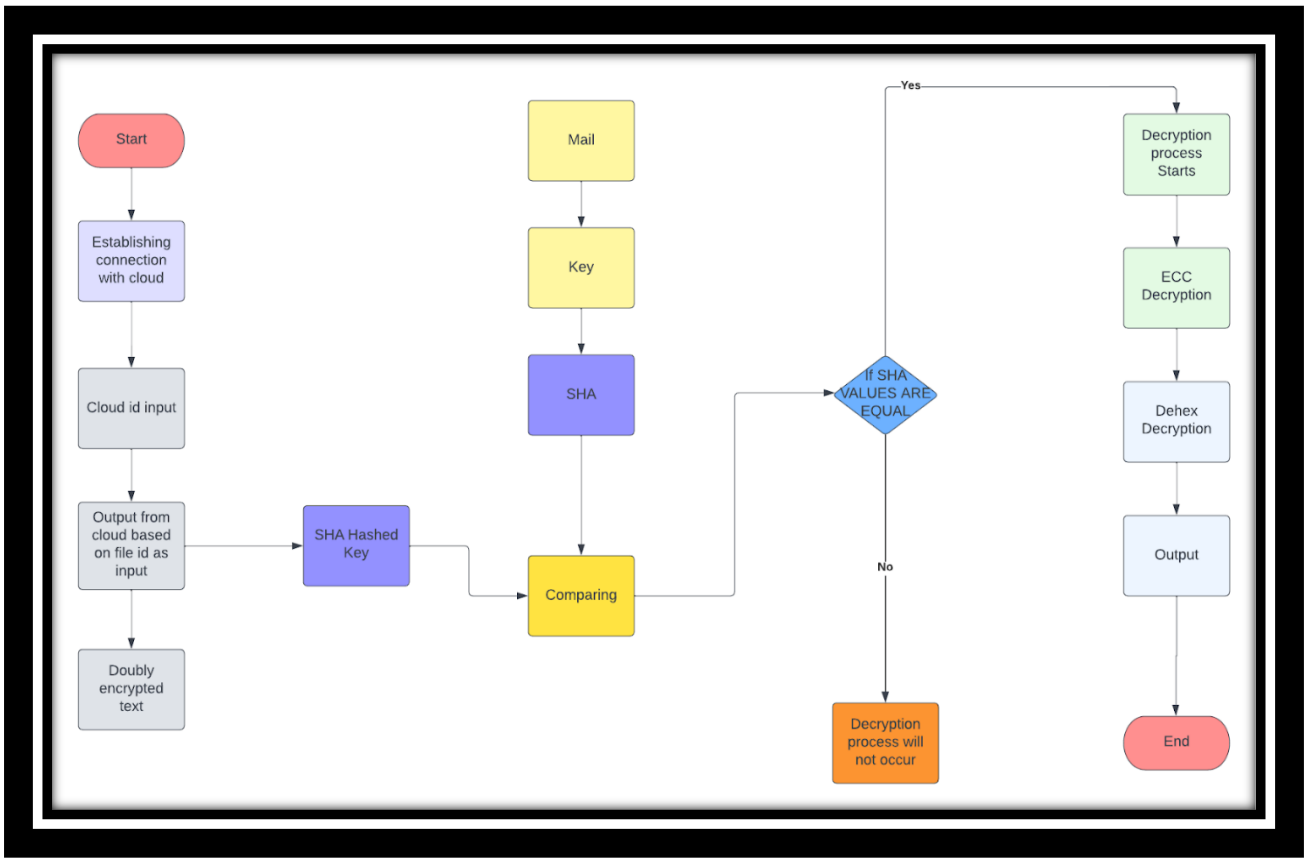


Fig 4: Work flow diagram of Decryption module.

The first stage of decryption is called ECC (Elliptical Curve Cryptography), where the double encrypted text is given as input. The Output from ECC is sent as input to the Dehex algorithm which is called the second round of decryption. The output will be the plain text (or) original message that users want to see.

4.4 Advantages of using Key:

- Key Generated by randomness which cannot be predicted.
- At each encryption, different keys are generated.
- Key size cannot be predicted which comes from the input size.
- The percentage that the key could be predicted is minimal.

4.5 Working of MongoDB Cloud:

It is defined as Highly flexible non-relational database which handles structured, unstructured, semi-structured data. It has a capacity to handle and store large amounts of data. It uses a different form storage format called BSON, which is called the Binary form of JSON format. It is used so that it can handle various data types.

MongoDB stores in the form of collections and documents. Collections are nothing but a set of documents. Documents consist of Key-value pairs, which are used for identification. The structure of the documents is not constant, as it changes when new data are added and deleted.

4.5.1 Parts in MongoDB:

- **Drivers:** They are used to communicate with MongoDB, which supports various programming languages.
- **MongoDB Shell:** Act as a java interface for MongoDB database. It is used during data taking and updating through queries.
- **Storage Engine:** It is used to store data in the memory and on the disk. It can have more than one Storage Engine.

4.5.2 MongoDB has 2 Layers:

- **Application Layer:** It consists of a user interface (or) Front-end and the server (or) Back-end is connected. Front-end is where a user interacts through his devices such as mobile. Back-end is used to interact with servers and drivers with the queries that users want.
- **Data Layer:** User queries are sent and are received in Storage Engine. Then the storage engine is responsible for reading (or) writing data in the file (or) in memory.

4.5.3 Advantages of using MongoDB:

- Flexibility,
- Greater performance.

4.5.4 Real Time applications of MongoDB:

- Real-Time Analysis,
- Content Management.

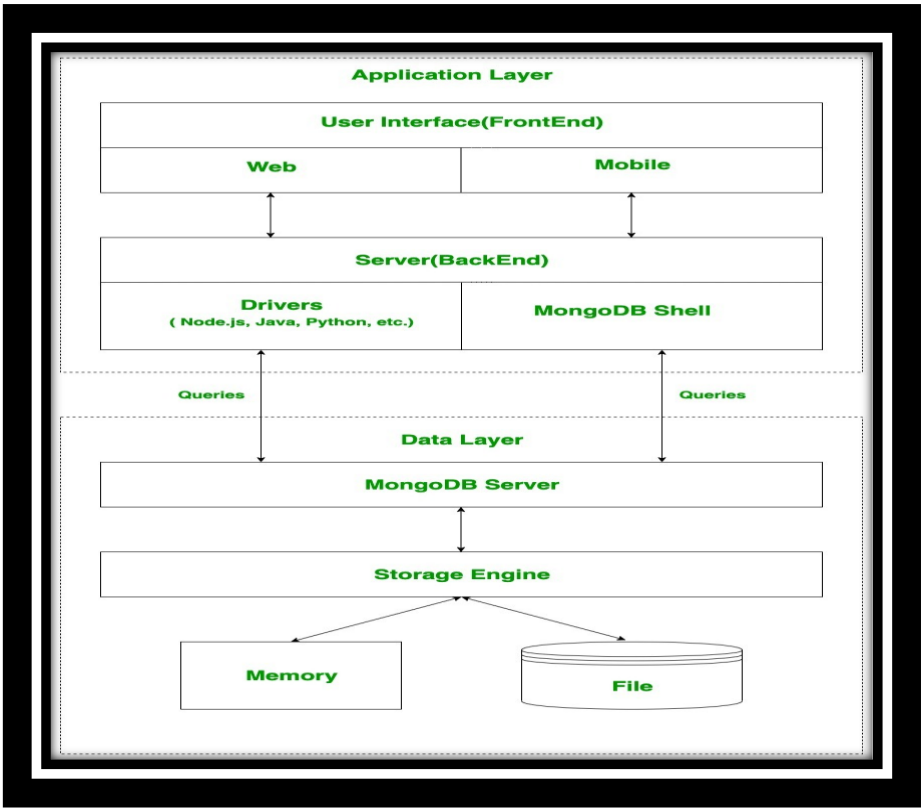


Fig 5: MongoDB Working module.

4.6 SMTP protocol:

It allows for the transfer of information between the sender and the receiver. It helps to transfer messages between different devices (or) on the same device. It is also considered as one of the best and emerging activities that are carried out through the internet. It first establishes connection with the receiver through TCP protocol. The connection is established through port 25. After establishing the connection, the mail is sent to the receiver successfully.

4.6.1 Working of SMTP Protocol:

- Electronic mail is sent using Mail User Agent (MUA). Messages consist of 2 parts: body and header. Body is where the user types the sender's and the receiver's address. Header contains some information such as the subject.
- Then the mail is sent to the SMTP server through port 25.
- Email addresses consist of 2 parts: Username and Domain name. EG: If email id is: sooryakumar@gmail.com, Here Username: sooryakumar and Domain name: gmail.com.
- Once the message (or) mail is received, the mail is stored in the server until the user sees it.
- User can then use his login credentials to access the mail.

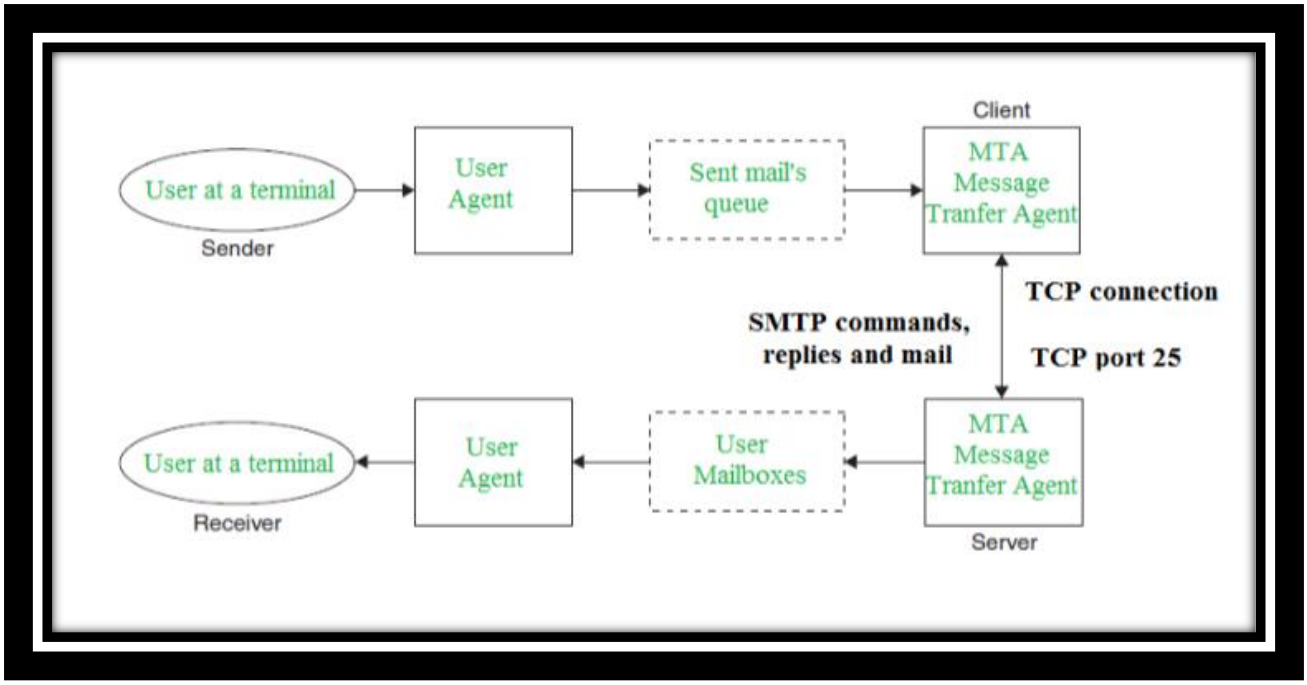


Fig 6: Working of Simple Mail Transfer Protocol (SMTP).

CHAPTER 5

ALGORITHMS AND WORKING PRINCIPLES

5.1 Dehex Algorithm:

It is a newly created Symmetric Key Cryptosystem (or) Secret Key Cryptography, which is easy and straightforward to use. it's used for faster encryption. This algorithm is employed to encrypt input files of text format only. This algorithm encompasses a key which is to be used for both encryption and decryption. it's also ready to transfer large amounts of information with minimal time and value.

It comes under Substitution Technique. It works under the Stream Cipher Model. It creates a key supported the computer file given by the user at run time. Since the secret's generated dynamically, its key size and key contents are unpredictable by the user. The generated secret is then added with the text input successively giving another text, which forms the primary level of encryption.

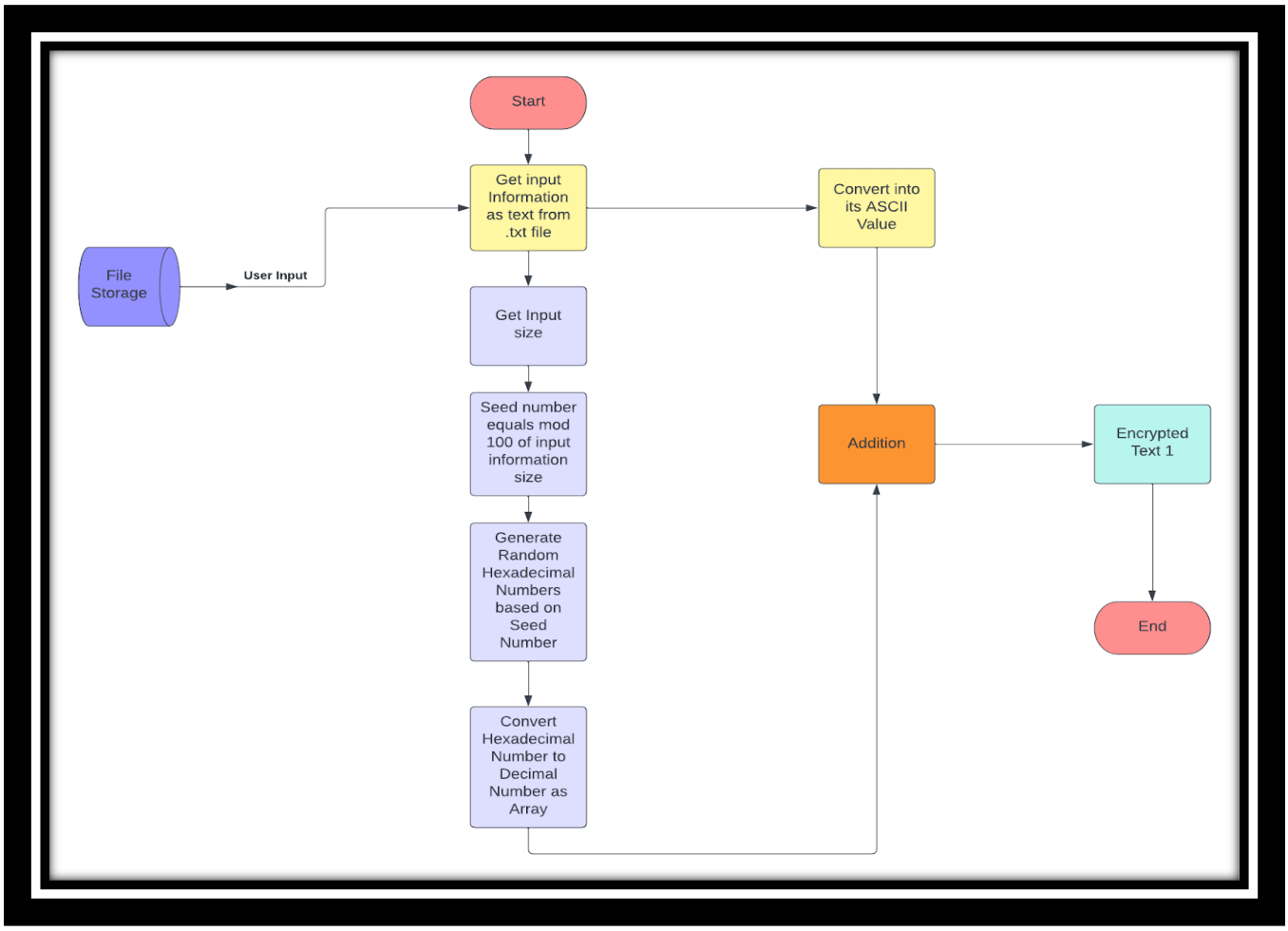


Fig 7: Working of Dehex algorithm

5.1.1 Advantages of Dehex algorithm:

- The key formed (or) created by using randomness technique where the system itself generates equivalent hexadecimal numbers from the dimensions of the computer file.
- Then equivalent decimals are calculated supported the hexadecimal numbers generated. because the given seed number is different from the obtained key, this creates a bonus of making variety for the user.
- The length of the key's unpredictable because it is generated indiscriminately. At every time in key, the numbers that are created indiscriminately which in together creates some security for the user.
- Key size depends upon the computer file which changes supported the user's needs. supported the file selected, the key's generated.
- Using a Stream Cipher technique, which quickens the computation process. there's no have to perform some operations like splitting.
- Since it works under Substitution technique, the plain text is replaced with other alternative texts in order that the initial contents don't seem to be read by anyone except the sender and receiver.

5.2 Elliptical Curve Cryptography (ECC):

- It is an Asymmetric Key (or) public key cryptosystem which is taken into account as a robust cryptographic approach. ECC maintains a high level of performance and security.
- In recent years, since industry has grown, this has been adopted by hundreds of companies as an innovative tool for security technology.
- It generates security between key pairs for public key encryption by using the mathematics of elliptic curves.
- This algorithm is popular due to smaller key size than the other algorithms.
- They are very difficult to crack mathematically.
- They are widely utilized in many web applications since key length is small and effective for usage.
- It can be satisfied through the following equation:

$$\mathbf{y^2=x^3 + ax + b.}$$

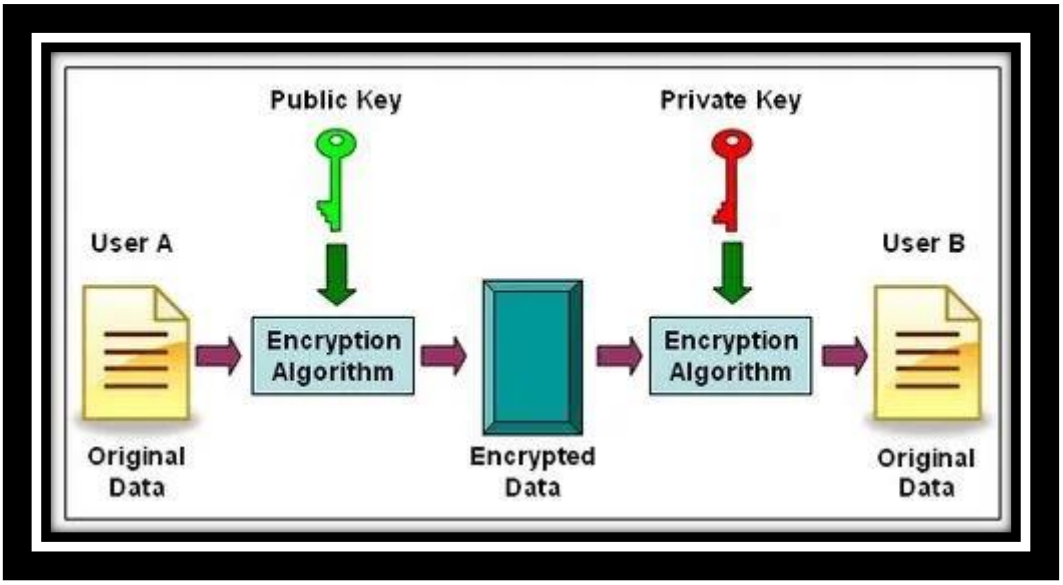


Fig 8: Working of ECC

5.2.1 Advantages of ECC:

- Shorter key and effective usage.
- Simple and easy to use.
- Encrypt the information with less energy.
- Fast encryption and decryption.
- Less time
- Faster key generation.

5.3 SHA:

- SHA Stands for Secure Hash Algorithm which is employed for hashing data. It shortens the input file into smaller forms which can't be understood.
- SHA isn't considered as an encryption algorithm, rather is taken into account as “One Way Hash Function”.
- Data is transformed into a secured format that's unreadable. But it will be readable providing the receiver encompasses a key that matches it.
- SHA Provides a feature of “One-Way-Encryption”, which suggests that there's no possibility of decryption.
- Even if one character is modified from the entire document, the SHA value are going to be different than obtained before. Since using it, whether or not the database is stolen (or) hacked then he would know only the hashed passwords and not the initial passwords.
- The family of SHA are classified into 6 Functions:
    - SHA-0.
    - SHA-1.
    - SHA-224.
    - SHA-256.

- SHA-384.
- SHA-512.

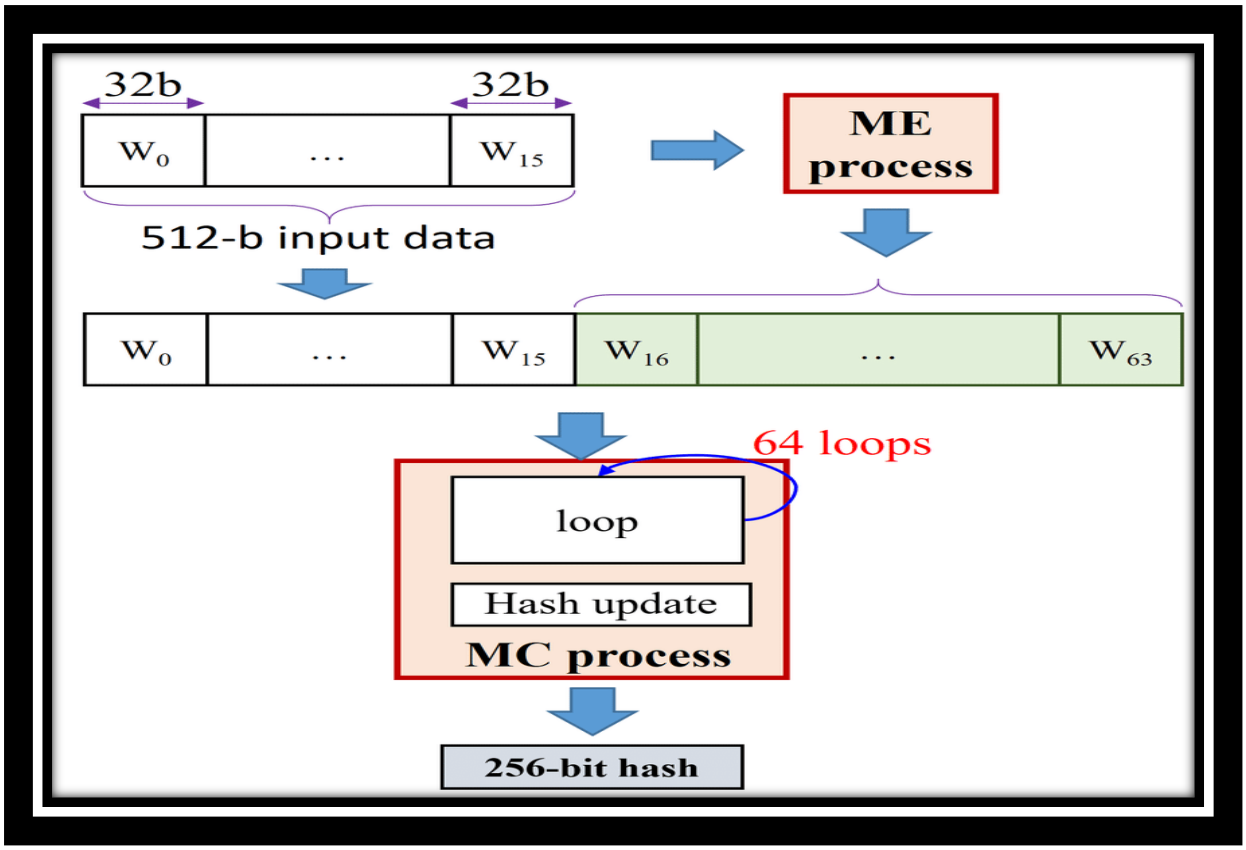


Fig 9: Working of SHA

CHAPTER 6

FORMULAS AND TIME COMPLEXITY

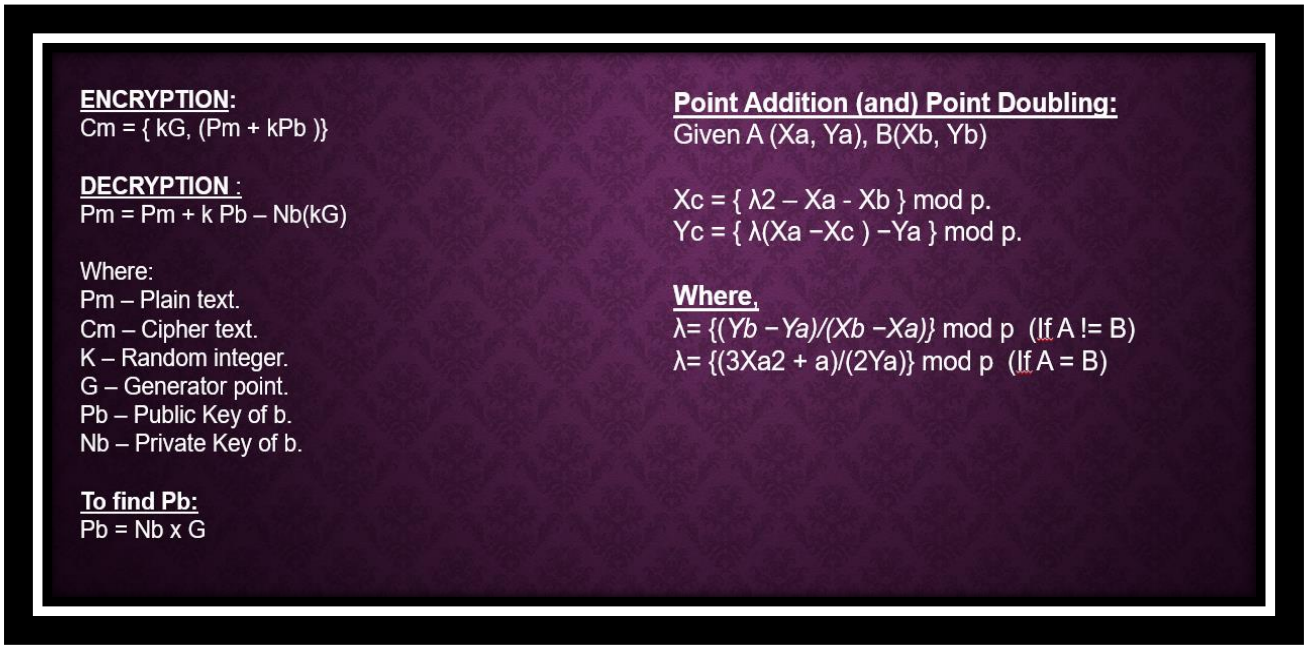


Fig 10: Formulas in Elliptical Curve Cryptography (ECC)

- The size of the input is made into mod 100, which act as a seed value for the key generation.
- After generating the key, simple addition is made along the text, so as to form the first round of encryption.

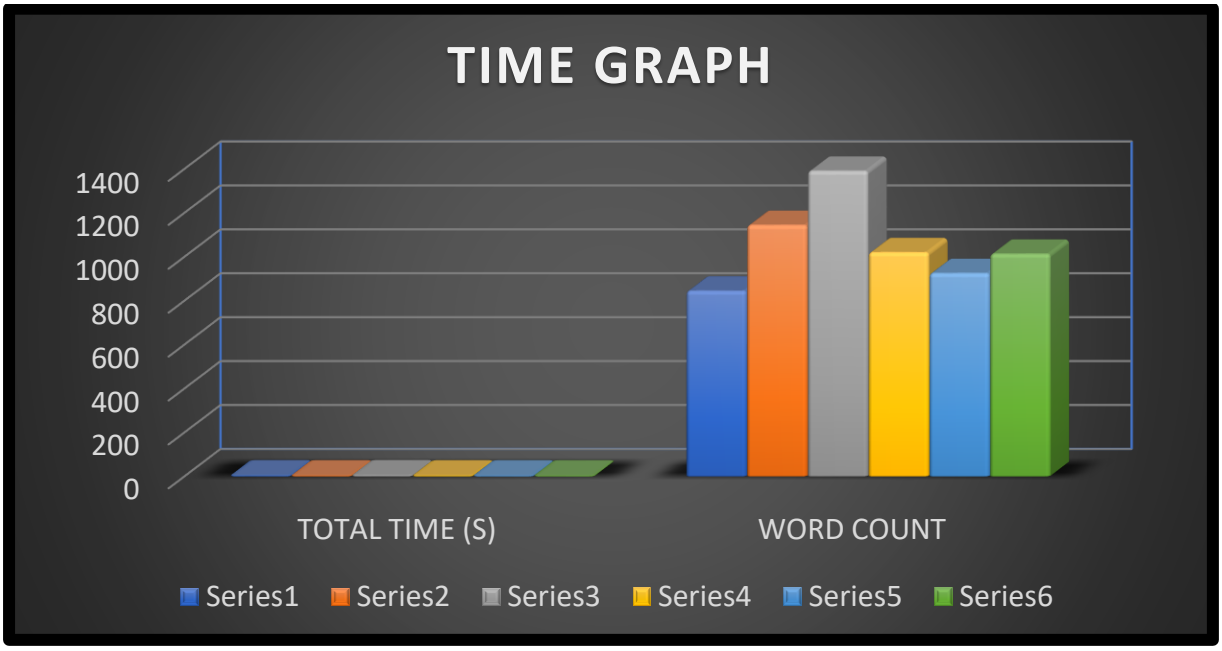


Fig 11: Graph plotted for Word Count and Total time taken

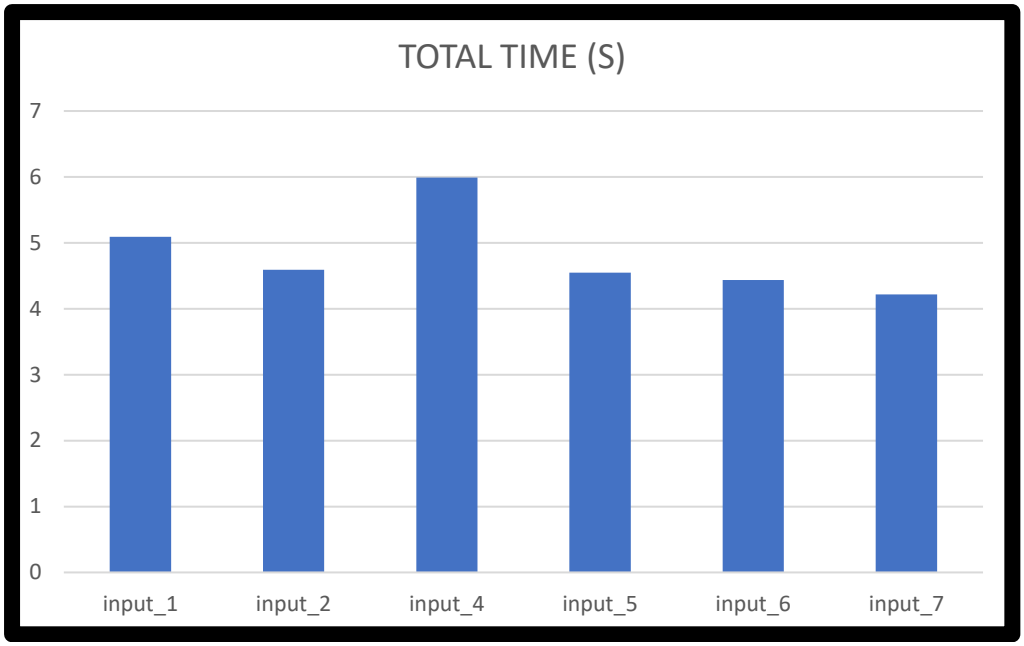


Fig 12: Graph Plotted for inputs taken and total time taken

Time Complexity:

N - number of characters in input file.
M - key size (1-100)

- **Best case:** Case 1: N<M. Then time complexity is O(M). Padding will be done to make N to M. (Say 100 is key size, and 45 is Input text, then 55 characters are to be appended and encryption for 100 characters which is equal to M is done.)
- **Average case:** N=M. Then time complexity is O(N) or O(M). [Best case and average case are same in this case].
- **Worst case:** N>M and when N%M=1, padding must be done in such a way that, (N+(M-1)) is done. So, the time complexity becomes O(M+N-1) ≈ O(M+N)

CHAPTER 7

LEARNING OBSERVATIONS

- Time complexity depends upon the time taken for generating the random number and padding of empty spaces in the text if required.

- When the size of input is large, then time taken for both encryption and decryption will also be larger.

- The message which is sent as input during the encryption process is brought again after the decryption process.

- The original contents in the message are not lost during the process of storing them in cloud and while retrieving them from cloud.

- From the time complexity graph, we can say that encryption and decryption process are at faster rate for small inputs and increase in time for large inputs.

- The key is made not to seen by others by the process of SHA. Since SHA cannot be decrypted.

- It provides good quality of encryption and decryption through symmetric and asymmetric algorithms.

CHAPTER 8

ADVANTAGES AND INFERENCES

8.1 Advantages:

- In Dehex Algorithm, the key generated is of randomness whose length cannot be predicted. The key used is stored and hence used in both encryption and decryption process.
- Even though Dehex algorithm can protect the text in secure way, to add more security it is then made to pass through the Second encryption Algorithm called as “ECC” which also protects the text in secure way. Thus, the input text is double encrypted.
- Instead of taking the local databases which could be a threat to attack, the encrypted information and SHA-Hashed Key is stored in large databases called as “Cloud”. Since Cloud is huge, the probability of finding the information is decreased.
- The key generated in Dehex is subjected to SHA where it is converted into alternative form. It is then used in the decryption process for key authentication and validation.
- The key stored in file format is then sent to receiver through Mail Transfer protocol for the decryption purpose.
- If encryption is done and stored in cloud, the user can decrypt them at their required time. There is no time bound that decryption must occur once encryption is done.

8.2 Inferences:

- There are minimum methods for file encryption. This new approach provides a pathway for encrypting the information at easier rate.
- Since the newly invented symmetric and existing asymmetric algorithms would form a hybrid cryptography, it provides a new way for encrypting a file.
- This can be a good method (or) alternate approach to various complex algorithms.
- Storing the information act as another advantage that file stored in cloud would be safe since cloud is huge and vast.

8.3 Concepts Achieved:

- **Confidentiality:** It ensures the documents are seen by the receiver only.
- **Authentication:** It checks whether if the receiver is the known and valid for the sender.
- **Integrity:** It ensures that only the sender and receiver have access to change the information.
- **Availability:** Sender and receiver have access to the information all the time.
- **Access Control:** Prevent the use of unauthorized access. It provides the feature that the resources are available for access at the target computers.
- **Privacy:** User has rights to taken control of the information.
- **Authorization:** Ability to access the resources by the receiver through mail sharing and cloud.

CHAPTER 9

RESULTS AND FUTURE ENHANCEMENTS

9.1 Results:

- The chances of unauthorized access are decreased by using the concept of double encryption and decryption.
- Using Symmetric and Asymmetric cryptosystem, the complexity of encryption will be higher and chances of stealing them would be minimal.
- The encrypted text along with key is stored in cloud, since the local storage might be under the threat of attack.
- The message which is sent as input during encryption process is same as message after decryption process is done.
- As cloud is vast and huge, the files are identified by the authorized user through the id which is generated at random.
- During each encryption process, different keys are generated which ensures that humans cannot predict.

9.2 Future Enhancements:

- Looking forward to create a web-based application (or) mobile application, where users need to upload their file and by clicking on encrypt button, the contents are encrypted and stored in cloud.
- Since this model would work only for text documents, I would extend this approach to encrypt all pdfs, documents and files which includes text.

CHAPTER 10

SOURCE CODE

10.1 Encryption Module:

```
from collections import namedtuple
Point = namedtuple("Point", "x y")
from tkinter import *
import datetime
import hashlib
import secrets
import smtplib
import ssl
import time
from datetime import timedelta
from collections import namedtuple
from email import encoders
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from random import *
import pymongo
import numpy

def encryption():
  input_file = inputfile.get()
  mail_id = mailaddr.get()
  start_time = time.time()
  def dehex():
    # DEHEX ALGORITHM - Round 1 Encryption
    # Reading input file from user
    from future.backports.email._encoded_words import len_b
    print(" Round 1 Encryption  : Dehex Algorithm ")
    message = ""
    inputFile = open(input_file, 'r')
    for line in inputFile:
      message = message + line
    def Convert(message):
      list1 = []
      list1[:0] = message
      return list1
    message_list = Convert(message)
    new_list = []
    for elem in message_list:
      temp = elem.split(', ')
      new_list.append((temp))
    len_input = len(new_list)
```

```
    result = []
    for elem in new_list:
        result.extend(ord(num) for num in elem)
    new_list_ascii = numpy.array(result)
    seed_number = (int(len(new_list) % 100))
    print(seed_number)
    if seed_number % 2 == 1:
        seed_number += 1
    else:
        seed_number
    hex_string = '0123456789abcdef'
    hexadecimal_number = ''.join([secrets.choice(hex_string) for x in range(seed_number)])
    res = int(hexadecimal_number, 16)
    first_part = hexadecimal_number[0:len(hexadecimal_number) // 2]
    second_part = hexadecimal_number[len(hexadecimal_number) // 2 if
len(hexadecimal_number) % 2 == 0
                                else ((len(hexadecimal_number) // 2) + 1):]
    hex_string1 = first_part
    hex_string2 = second_part
    first_part1 = int(hex_string1, 16)
    second_part1 = int(hex_string2, 16)
    hex_value1 = hex(first_part1)
    hex_value2 = hex(second_part1)
    hex_value1 = numpy.array(hex(first_part1))
    hex_value2 = numpy.array((hex(second_part1)))
    hex_value1 = hex_value1.tolist()
    hex_value2 = hex_value2.tolist()
    a = hex(first_part1 ^ second_part1)
    b = a[2:]
    c = (hexadecimal_number + b)
    res1 = int(c, 16)
    final_number = res ^ res1
    final_number_str = str(final_number)
    final_number_list = [int(x) for x in final_number_str]
    final_number_arr = numpy.array(final_number_list)
    print(final_number_arr)
    size_final_number_array = len(final_number_arr)
    remainder = int((len_input % size_final_number_array))
    pad_size = size_final_number_array - remainder
    message = message + (' ' * pad_size)
    message_list = Convert(message)
    new_list_1 = []
    for elem in message_list:
        temp = elem.split(', ')
        new_list_1.append((temp))
    len_input_1 = len(new_list_1)
    result = []
    for elem in new_list_1:
        result.extend(ord(num) for num in elem)
    new_list_ascii_1 = numpy.array(result)
```

```
        final_pad = []
        quotient = int(len_input_1 / size_final_number_array)
        new_list_ascii_1 = new_list_ascii_1.reshape(quotient, size_final_number_array)
        # final_number_arr = final_number_arr.reshape(1, (size_final_number_array*quotient))
        final_pad = new_list_ascii_1 + final_number_arr
        final_pad = numpy.array(final_pad).flatten()
        Round_1_enc = "".join([chr(c) for c in final_pad])
        print("Encrypted text after Dehex is:", Round_1_enc)
        xx = randint(1, 1000)
        newfilename = "key_".__add__(str(xx)).__add__(".txt")
        print(xx)
        with open(newfilename, 'w') as f:
            s = str(final_number_arr)
            f.write(s)

        dehex_dict = dict()
        dehex_dict['Round_one_res'] = Round_1_enc
        dehex_dict['key_file'] = newfilename
        dehex_dict['id'] = xx
        return dehex_dict
    res_dehex = dehex()

    encrypted_output = res_dehex.get('Round_one_res')
    encryption_key = res_dehex.get('key_file')
    with open('dehex_enc.txt', 'w') as f:
        s = str(encrypted_output)
        f.write(s)

 # ECC – Second Round of Encryption
f = open('dehex_enc.txt', 'r')
    original = f.read()
    f.close()
    print('Entered Text=', original)
    eO = 'ORIGIN'
    ep = 71
    ea = 1
    eb = 3
    ena = 47
    enb = 8
    k = 30
    eg = '35,10'
    epa = epb = ekpb = nbkg = ''
    bina = ''
    pt = []
    radix = []
    for i in range(65, 91):
        radix.append(chr(i))
    for i in range(97, 123):
        radix.append(chr(i))
    for i in range(10):
```

```
    radix.append(str(i))
radix.append('+')
radix.append(',')
points = []
lhs = rhs = 0
for i in range(ep):
    for j in range(ep):
        lhs = (j * j) % ep
        rhs = ((i * i * i) + (ea * i) + eb) % ep
        if lhs == rhs:
            points.append(str(i) + ',' + str(j))
points.append(eO)
def valid(P):
    if P == eO:
        return True
    else:
        return (
                (P.y ** 2 - (P.x ** 3 + ea * P.x + eb)) % ep == 0 and
                0 <= P.x < ep and 0 <= P.y < ep)
def inv_mod_p(x):
    if x % ep == 0:
        raise ZeroDivisionError("Impossible inverse")
    return pow(x, ep - 2, ep)
def ec_inv(P):
    if P == eO:
        return P
    return Point(P.x, (-P.y) % ep)
def ec_add(P, Q):
    if not (valid(P) and valid(Q)):
        raise ValueError("Invalid inputs")
    if P == eO:
        result = Q
    elif Q == eO:
        result = P
    elif Q == ec_inv(P):
        result = eO
    else:
        if P == Q:
            dydx = (3 * P.x ** 2 + ea) * inv_mod_p(2 * P.y)
        else:
            dydx = (Q.y - P.y) * inv_mod_p(Q.x - P.x)
        x = (dydx ** 2 - P.x - Q.x) % ep
        y = (dydx * (P.x - x) - P.y) % ep
        result = Point(x, y)
    assert valid(result)
    return result
nat = ena
xp = xq = int(eg.split(',')[0])
yp = yq = int(eg.split(',')[1])
P = Point(xp, yp)
Q = Point(xq, yq)
```

```
if ena == 1:
   epa = eg
else:
   while nat != 1:
      r = ec_add(P, Q)
      if r == eO:
         P = r
      else:
         xp = r.x
         yp = r.y
         P = Point(xp, yp)
      nat = nat - 1
   if r == eO:
      epa = r
   else:
      epa = str(xp) + ',' + str(yp)
nat = enb
xp = xq = int(eg.split(',')[0])
yp = yq = int(eg.split(',')[1])
P = Point(xp, yp)
Q = Point(xq, yq)
if enb == 1:
   epb = eg
else:
   while nat != 1:
      r = ec_add(P, Q)
      if r == eO:
         P = r
      else:
         xp = r.x
         yp = r.y
         P = Point(xp, yp)
      nat = nat - 1
   if r == eO:
      epb = r
   else:
      epb = str(xp) + ',' + str(yp)
nat = k
if k == 1:
   ekpb = epb
else:
   if epb == eO:
      P = Q = eO
      while nat != 1:
         r = ec_add(P, Q)
         if r == eO:
            P = r
         else:
            xp = r.x
            yp = r.y
            P = Point(xp, yp)
```

```
            nat = nat - 1
        if r == eO:
            ekpb = r
        else:
            ekpb = str(xp) + ',' + str(yp)
    else:
        xp = xq = int(epb.split(',')[0])
        yp = yq = int(epb.split(',')[1])
        P = Point(xp, yp)
        Q = Point(xq, yq)
        while nat != 1:
            r = ec_add(P, Q)
            if r == eO:
                P = r
            else:
                xp = r.x
                yp = r.y
                P = Point(xp, yp)
            nat = nat - 1
        if r == eO:
            ekpb = r
        else:
            ekpb = str(xp) + ',' + str(yp)
temp = original
for i in temp:
    bina = bina + format(ord(i), '08b')
if len(bina) % 6 != 0:
    for i in range((6 - (len(bina) % 6))):
        bina = bina + str(0)
bina.replace('0b', '')
i = 0
while (i != len(bina)):
    t = bina[i:i + 6]
    t = '0b' + t
    t = int(t, 2)
    pt.append(points[t])
    i = i + 6
for ok in pt:
    if ok == eO:
        P = eO
    else:
        t = ok.split(',')
        P = Point(int(t[0]), int(t[1]))
    if ekpb == eO:
        Q = eO
    else:
        Q = Point(int(ekpb.split(',')[0]), int(ekpb.split(',')[1]))
    r = ec_add(P, Q)
    if r == eO:
        pt[pt.index(ok)] = radix[points.index(eO)]
    else:
```

```
            xp = r.x
            yp = r.y
            pt[pt.index(ok)] = radix[points.index(str(xp) + ',' + str(yp))]

    ct = ''.join(pt)
    ct1 = ct
    print('Encrypted Text=', ct1)
    f = open('ECC1_enc.txt', 'w')
    f.write(ct1)
    f.close()

# SHA – Key Value
    newfilename1 = res_dehex.get('key_file')
    result = hashlib.sha256(newfilename1.encode())
    print("The hexadecimal equivalent of SHA256 is : ")
    i = result.hexdigest()
    print(i)

    # Establishing Connection with the cloud using MongoDB
    cloud_password = ""
    cloud_pswd = open('cloud_password.txt', 'r')
    for line in cloud_pswd:
        cloud_password = cloud_password + line
    client = pymongo.MongoClient(
        "mongodb+srv://soorya:" + cloud_password +
"@cluster0.hl7ws.mongodb.net/Main?retryWrites=true&w=majority")
    # db = client.test
    rid = res_dehex.get('id')
    db = client["Main"]
    collections = db["Main"]
    current_time = datetime.datetime.now()
    post = {"_id": rid, "name": "dehex_ecc", "Encrypted information": ct1, "Key_Dehex": i,
            "Date and Time": current_time}
    collections.insert_one(post)
    print("Cloud Connection Established")

    # Creating Key Management through SMTP protocol - Mail Transfer
    pswd = ""
    input_pswd = open('Mail_password.txt', 'r')
    for line in input_pswd:
        pswd = pswd + line
    subject = "Security key for file stored in cloud ID: " + str(rid) + "."
    body = "This is an email with attachment sent from Python which has a security key file for
encrypted file stored in Cloud whose ID is: " + str(
        rid) + ", which is uploaded on " + str(current_time) + "."
    sender_email = "222003091@sastra.ac.in"
    receiver_email = mail_id
    password = pswd
    message = MIMEMultipart()
    message["From"] = sender_email
    message["To"] = receiver_email
```

```
  message["Subject"] = subject
  # message["Bcc"] = receiver_email  # Recommended for mass emails
  message.attach(MIMEText(body, "plain"))
  filename = newfilename1  # In same directory as script
  with open(filename, "rb") as attachment:
     part = MIMEBase("application", "octet-stream")
     part.set_payload(attachment.read())
  encoders.encode_base64(part)
  part.add_header(
     "Content-Disposition",
     f"attachment; filename= {filename}",
  )
  message.attach(part)
  text = message.as_string()
  # Log in to server using secure context and send email
  context = ssl.create_default_context()
  with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
     server.login(sender_email, password)
     server.sendmail(sender_email, receiver_email, text)
  emptylabel.config(text='Encryption successful! Data stored in cloud with ID: '+str(rid)+'.',
fg='red', font=('Arial', 20))
  # Getting the time taken by the algorithm
  elapsed_time_secs = time.time() - start_time
  msg_time = "Execution took: %s secs (Wall clock time)" %
timedelta(seconds=round(elapsed_time_secs))
  print(elapsed_time_secs)

# Simple GUI using Tkinter
window=Tk()
window.title("Encryption")
window.geometry('1000x500')
label1 = Label(window,text='Enter the input file : ',fg = 'blue', font =('Arial',14))
label1.grid(row=1,column=1,padx=5,pady=10)
inputfile=StringVar()
mailaddr=StringVar()
textbox1 = Entry(window,textvariable=inputfile,fg = 'blue', font =('Arial',14))
textbox1.grid(row=1,column=2)
label2 = Label(window,text='Enter your mail id : ',fg = 'red', font =('Arial',14))
label2.grid(row=2,column=1,padx=5,pady=10)
textbox2 = Entry(window,textvariable=mailaddr,fg = 'red', font =('Arial',14))
textbox2.grid(row=2,column=2)
button1 = Button(window, command=encryption, text='Encrypt',fg = 'black', font
=('Arial',14))
button1.grid(row=4,column=3,padx=10)

emptylabel=Label(window,fg='green',font=('Arial',20))
emptylabel.grid(row=10,column=10,pady=10)
window.mainloop()
```

10.2 Decryption Module:

```
from datetime import timedelta
from tkinter import *
import hashlib
from random import *
from collections import namedtuple
import numpy
import points as points
Point = namedtuple("Point", "x y")
import pymongo
import time

def decryption():
   # To Establish Connection with Cloud
   start_time = time.time()
   file_id = inputfileID.get()
   file_id= int(file_id)
   input_key_file = inputfilepath.get()
   client = pymongo.MongoClient(
      "mongodb+srv://soorya:soorya@cluster0.hl7ws.mongodb.net/Main?retryWrites=true&w
=majority")
   db = client["Main"]
   collections = db["Main"]
   cloud_data = collections.find_one({"_id": file_id})
   print(cloud_data)
   # Getting the encrypted information from cloud
   encrypted_output = cloud_data.get('Encrypted information')
   print(encrypted_output)
   # Getting the key of Dehex From the Cloud
   encrypt_key_from_cloud = cloud_data.get('Key_Dehex')
   print(encrypt_key_from_cloud)
   # Getting the key_id.txt from Local Directory
   message1 = ""
   # Applying SHA to the Key_id.txt
   result = hashlib.sha256(input_key_file.encode())
   print("The hexadecimal equivalent of SHA256 is : ")
   sha_decryption = result.hexdigest()
   print(sha_decryption)
   if (encrypt_key_from_cloud == sha_decryption):
      print("Key matched! Decryption starts.")

      # Decryption for ECC
      ct1 = encrypted_output
      print('TEXT READ FROM FILE:', ct1)
      eO = 'ORIGIN'
      ep = 71
      ea = 1
      eb = 3
      ena = 47
      enb = 8
```

```
k = 30
eg = '35,10'
epa = epb = ekpb = nbkg = ''
radix = []
for i in range(65, 91):
    radix.append(chr(i))
for i in range(97, 123):
    radix.append(chr(i))
for i in range(10):
    radix.append(str(i))
radix.append('+')
radix.append(',')
points = []
lhs = rhs = 0
for i in range(ep):
    for j in range(ep):
        lhs = (j * j) % ep
        rhs = ((i * i * i) + (ea * i) + eb) % ep
        if lhs == rhs:
            points.append(str(i) + ',' + str(j))
points.append(eO)
def valid(P):
    if P == eO:
        return True
    else:
        return (
                (P.y ** 2 - (P.x ** 3 + ea * P.x + eb)) % ep == 0 and
                0 <= P.x < ep and 0 <= P.y < ep)
def inv_mod_p(x):
    if x % ep == 0:
        raise ZeroDivisionError("Impossible inverse")
    return pow(x, ep - 2, ep)
def ec_inv(P):
    if P == eO:
        return P
    return Point(P.x, (-P.y) % ep)
def ec_add(P, Q):
    if not (valid(P) and valid(Q)):
        raise ValueError("Invalid inputs")
    if P == eO:
        result = Q
    elif Q == eO:
        result = P
    elif Q == ec_inv(P):
        result = eO
    else:
        if P == Q:
            dydx = (3 * P.x ** 2 + ea) * inv_mod_p(2 * P.y)
        else:
            dydx = (Q.y - P.y) * inv_mod_p(Q.x - P.x)
        x = (dydx ** 2 - P.x - Q.x) % ep
```

```
        y = (dydx * (P.x - x) - P.y) % ep
        result = Point(x, y)
    assert valid(result)
    return result
nat = k * enb
if epb == eO:
    P = Q = eO
else:
    xp = xq = int(eg.split(',')[0])
    yp = yq = ep - int(eg.split(',')[1])
    P = Point(xp, yp)
    Q = Point(xq, yq)
if nat == 1:
    nbkg = eg
else:
    while nat != 1:
        r = ec_add(P, Q)
        if r == eO:
            P = r
        else:
            xp = r.x
            yp = r.y
            P = Point(xp, yp)
        nat = nat - 1
    if r == eO:
        nbkg = r
    else:
        nbkg = str(xp) + ',' + str(yp)
bina = ''
ct = []
for i in ct1:
    ct.append(points[radix.index(i)])
for ok in ct:
    if ok == eO:
        P = eO
    else:
        t = ok.split(',')
        P = Point(int(t[0]), int(t[1]))
    if nbkg == eO:
        Q = eO
    else:
        Q = Point(int(nbkg.split(',')[0]), int(nbkg.split(',')[1]))
    r = ec_add(P, Q)
    if r == eO:
        ct[ct.index(ok)] = points.index(eO)
    else:
        xp = r.x
        yp = r.y
        ct[ct.index(ok)] = points.index(str(xp) + ',' + str(yp))
for i in ct:
    bina = bina + format(i, '06b')
```

```
if len(bina) % 8 != 0:
    for i in range((8 - (len(bina) % 8))):
        bina = bina + str(0)
bina.replace('0b', '')
i = 0
ct = []
while (i != len(bina)):
    t = bina[i:i + 8]
    t = '0b' + t
    t = int(t, 2)
    ct.append(chr(t))
    i = i + 8
pt1 = ''.join(ct)
pt1 = pt1.rstrip('\x00')
print('Decrypted Text=', pt1)

# Dehex decryption:
final_number_list = [x for x in pt1]
final_number_arr = numpy.array(final_number_list)
print(" Array format of Encrypted Text : ", final_number_arr)
result = []
for elem in final_number_arr:
    result.extend(ord(num) for num in elem)
    new_list_ascii = numpy.array(result)
print(" ASCII Value of Encrypted array : ", new_list_ascii)
# encrypt_key1 = str(encrypt_key)
message1 = ""
inputFile = open(input_key_file, 'r')
for line in inputFile:
    message1 = message1 + line
    message1 = message1.strip()
    message1 = message1.rstrip(']')
    message1 = message1.lstrip('[')
    message1 = message1.replace(" ", "")
    message1 = "".join(message1.splitlines())
print(message1)
encrypt_key_list = [x for x in message1]
print(encrypt_key_list)
key_array = [int(x) for x in encrypt_key_list]
encrypt_key_arr = numpy.array(key_array)
print(encrypt_key_arr)
# key_array = [int(x) for x in encrypt_key_arr]
print(" Key as Array format ", encrypt_key_arr)
encrypt_key_length = len(key_array)
print(" The key size is : ", encrypt_key_length)
encrypt_text_length = len(pt1)
print("Text Length is : ", encrypt_text_length)
quotient = int(encrypt_text_length / encrypt_key_length)
reshaped_ASCII_array = new_list_ascii.reshape(quotient, encrypt_key_length)
final_dec = []
final_dec = reshaped_ASCII_array - encrypt_key_arr
```

```
      final_dec = numpy.array(final_dec).flatten()
      print(final_dec)
      final_1_enc = "".join([chr(c) for c in final_dec])
      print(final_1_enc)
      emptylabel.config(text=final_1_enc, fg='red', font=('Arial', 14), wraplength=1000, justify='left')

      with open('output.txt','w')as f:
         s = str(final_1_enc)
         f.write(s)
  else:
     print("Key not matching! Decryption can't be done.")
     emptylabel.config(text="Key not matching! Decryption can't be done.", fg='red', font=('Arial', 20), wraplength=500, justify='left')
  elapsed_time_secs = time.time() - start_time
  msg_time = "Execution took: %s secs (Wall clock time)" % timedelta(seconds=round(elapsed_time_secs))
  print(elapsed_time_secs)

# GUI Using Tkinter
window=Tk()
window.title("Decryption")
window.geometry('1000x500')
label1 = Label(window,text='Enter the cloud file ID : ',fg = 'blue', font =('Arial',14))
label1.grid(row=1,column=1,padx=5,pady=10)
inputfileID=StringVar()
inputfilepath=StringVar()
textbox1 = Entry(window,textvariable=inputfileID,fg = 'blue', font =('Arial',14))
textbox1.grid(row=1,column=2)
label2 = Label(window,text='Enter key File name or filepath : ',fg = 'red', font =('Arial',14))
label2.grid(row=2,column=1,padx=5,pady=10)
textbox2 = Entry(window,textvariable=inputfilepath,fg = 'red', font =('Arial',14))
textbox2.grid(row=2,column=2)
button1 = Button(window, command=decryption, text='Decrypt',fg = 'black', font =('Arial',14))
button1.grid(row=4,column=3,padx=10)
window1=Tk()
window1.title("Output")
window1.geometry('500x500')
emptylabel=Label(window1,fg='green',font=('Arial',20))
emptylabel.grid(row=1,column=1)
window.mainloop()
```

CHAPTER 11

OUTPUT SCREENSHOTS

Fig 13: Encryption module

Fig 14: Python terminal for encryption

Fig 15: Key file along with id is sent to receiver through SMTP protocol

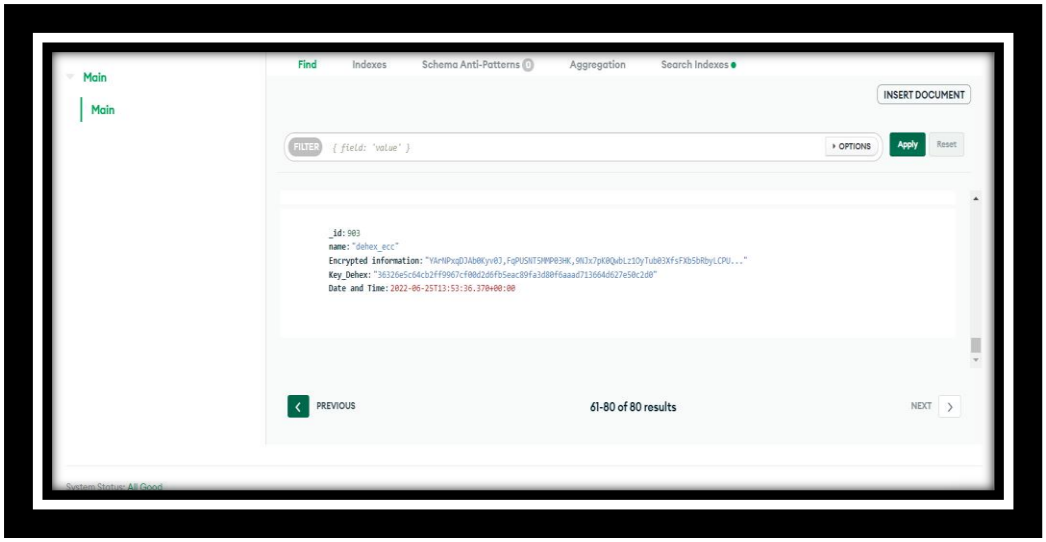


Fig 16: Doubly encrypted text, key hashed, id is stored in cloud.

Fig 17: Decryption module.

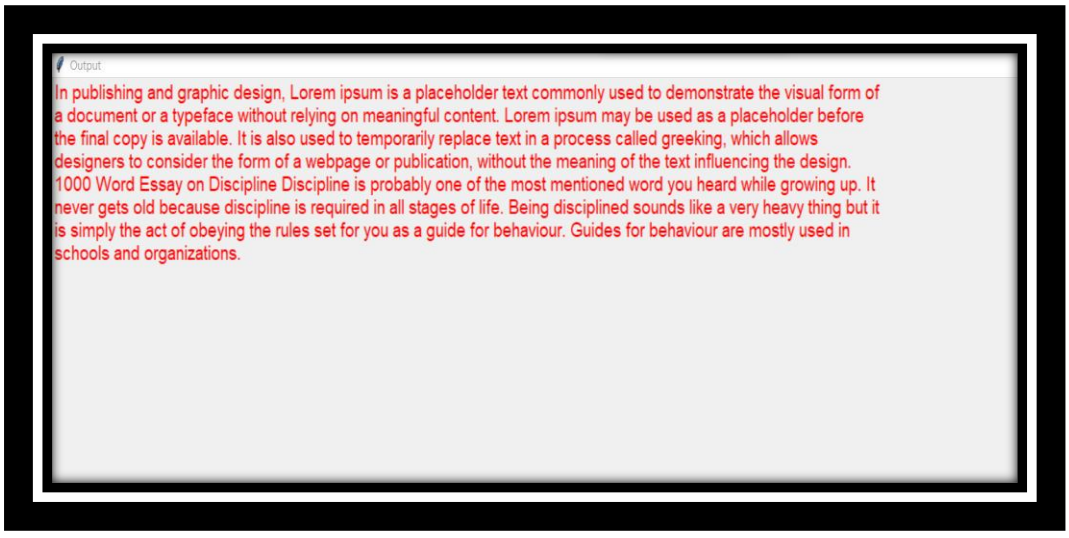


Fig 18: Output window

CHAPTER 12

REFERENCES

1. Data Security and Privacy Protection for Cloud Storage: A Survey by Pan Yang ,Naixue Xiong ,Jingli Ren – 2020

2. Dynamic multi-client searchable symmetric encryption with support for Boolean queries by Leilei Du, Kenli Li, Qin Liu, Zhiqiang Wu, Shaobo Zhang - 2020

3. Privacy-Preserving Tensor Decomposition Over Encrypted Data in a Federated Cloud Environment by Jun Feng, Laurence T. Yang, Qing Zhu, Kim-Kwang Raymond Choo - 2020

4. Secure Data Sharing in Cloud Computing Using Revocable-Storage Identity-Based Encryption by Kwangsu Lee - 2020

5. Cryptographic Solution-Based Secure Elliptic Curve Cryptography Enabled Radio Frequency Identification Mutual Authentication Protocol for Internet of Vehicles by Surbhi Sharma, Baijnath Kaushik, Mohammad Khalid Iman Rahmani, Mohammad Ezaz Ahmed - 2021

6. Security Threats and Challenges in Public Cloud Storage by G. Nagarajan, K. Sampath Kumar - 2021

7. A detailed review of Cloud Security: Issues, Threats & Attacks by Aditi Patel, Nisarg Shah, Dipak Ramoliya, Amit Nayak – 2020

8. Cloud Cryptography: User End Encryption by Sameer A. Nooh - 2020