

# **SECURING MEDICAL IMAGING**

**ST. TERESA'S COLLEGE(AUTONOMOUS)  
AFFILIATED TO MAHATMA GANDHI UNIVERSITY**



## **PROJECT REPORT**

*In partial fulfillment of the requirements for the award of the degree of*  
**BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY  
MANAGEMENT)**

*By*  
**GAYATHRI G- SB21BCA015**  
*&*  
**NANDANA P S- SB21BCA027**

**III DC BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY  
MANAGEMENT)**

*Under the guidance of*  
**VEENA ANTONY**

**DEPARTMENT OF BCA (CLOUD TECHNOLOGY AND  
INFORMATION SECURITY MANAGEMENT)  
MARCH 2024**

# **SECURING MEDICAL IMAGING**

**ST. TERESA'S COLLEGE(AUTONOMOUS)  
AFFILIATED TO MAHATMA GANDHI UNIVERSITY**



## **PROJECT REPORT**

*In partial fulfillment of the requirements for the award of the degree of*  
**BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY  
MANAGEMENT)**

*By*  
**GAYATHRI G- SB21BCA015**  
*&*  
**NANDANA P S- SB21BCA027**

**III DC BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY  
MANAGEMENT)**

*Under the guidance of*  
**VEENA ANTONY**

**DEPARTMENT OF BCA (CLOUD TECHNOLOGY AND  
INFORMATION SECURITY MANAGEMENT)  
MARCH 2024**

## **DECLARATION**

We, undersigned, hereby declare that the project report, **‘SECURING MEDICAL IMAGING’**, submitted for partial fulfillment of the requirements for the award of degree of BCA (Cloud Technology and Information Security Management) at St. Teresa’s College (Autonomous), Ernakulam (Affiliated to Mahatma Gandhi University), Kerala, is a bonafide work done by us under the supervision of Veena Antony. This submission represents our ideas in our own words where ideas or words of others have not been included. We have adequately and accurately cited and referenced the sources. We also declare that we have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any image idea, fact, or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma, or similar title of any other University.

Ernakulam

March 2024

GAYATHRI G –SB21BCA015

NANDANA P S – SB21BCA027

**ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM**  
**BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY**  
**MANAGEMENT)**



**CERTIFICATE**

This is to certify that the report entitled “**SECURING MEDICAL IMAGING**”, submitted by Gayathri G and Nandana P S to the Mahatma Gandhi University in partial fulfillment of the requirements for the award of the Degree of BCA (Cloud Technology and Information Security Management) is a bonafide record of the project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.



**VEENA ANTONY**  
**Internal Supervisor**

**ARCHANA P MENON**  
**Head of the department**

**External Supervisor**

## ACKNOWLEDGEMENT

First and foremost, we thank God Almighty for his blessings. We take this opportunity to express our gratitude to all those who helped us in completing this project successfully. I wish to express our sincere gratitude to the **Manager Rev. Dr. Sr. Vinitha CSST** and the Principal **Dr. Alphonsa Vijaya Joseph** for providing all the facilities.

We express our sincere gratitude towards the Head of the department **Mrs. Archana Menon P.** We deeply express sincere thanks to our project guide **Mrs. Veena Antony** for her proper guidance and support throughout the project work.

We are indebted to our beloved teachers whose cooperation and suggestions throughout the project helped us a lot. We thank all our friends and classmates for their support.

We convey our hearty thanks to our parents for their moral support, suggestions, and encouragement.

## **ABSTRACT**

Medical images play an essential role in e-healthcare applications for providing quick and better remote diagnosis and treatment to patients. Medical images generally comprise secret details about the patients and are therefore prone to various security threats during their transmission over public networks. The recent COVID-19 has highlighted the fact that medical images are consistently created and disseminated online, leading to a need for protection from unauthorized utilization. Thus, it is required to secure these images before their communication over public networks. However, due to the distinctive properties of medical images, like higher correlation and redundancy among the pixels, and larger size, it is required to design an efficient encryption model to resist various security threats. This project proposes a solution to address these challenges by implementing a hybrid encryption scheme for securing medical images. The proposed system combines the strengths of symmetric and asymmetric encryption algorithms to achieve a balance between efficiency and security. Symmetric encryption is utilized for bulk image encryption due to its speed, while asymmetric encryption is employed for key exchange and ensuring confidentiality. Additionally, SHA2 is applied to verify the integrity of the encrypted images.

Furthermore, the project explores the integration of access control mechanisms to manage the permissions of healthcare professionals accessing encrypted medical images. Role-based access control (RBAC) is considered to enforce fine-grained access policies based on the roles and responsibilities of individuals within the healthcare organization.

## TABLE OF CONTENTS

<b>Chapter 1 INTRODUCTION .....</b>	<b>1</b>
<b>1.1 General Background .....</b>	<b>1</b>
<b>1.2 Information security .....</b>	<b>1</b>
<b>1.3 Principles of Information Security.....</b>	<b>1</b>
<b>1.3.1 Confidentiality .....</b>	<b>2</b>
<b>1.3.2 Integrity.....</b>	<b>2</b>
<b>1.3.3 Availability.....</b>	<b>3</b>
<b>1.4 Cryptography.....</b>	<b>3</b>
<b>1.4.1 Types of Cryptography.....</b>	<b>4</b>
<b>1.4.2 Client-side Cryptography.....</b>	<b>5</b>
<b>1.5 Server-side Encryption.....</b>	<b>6</b>
<b>1.6 AES algorithm .....</b>	<b>7</b>
<b>1.7 Triple DES Algorithm.....</b>	<b>10</b>
<b>1.8 RSA Algorithm.....</b>	<b>11</b>
<b>1.9 Objectives.....</b>	<b>13</b>
<b>Chapter 2 LITERATURE SURVEY.....</b>	<b>14</b>
<b>Chapter 3 EXISTING SYSTEM.....</b>	<b>16</b>
<b>3.1 Drawbacks of Existing System.....</b>	<b>16</b>
<b>3.2 Functional Workflow of Existing System.....</b>	<b>17</b>
<b>Chapter 4 PROPOSED SYSTEM.....</b>	<b>18</b>
<b>4.1 Merits of the Proposed System .....</b>	<b>18</b>
<b>Chapter 5 SYSTEM DESIGN AND ARCHITECTURE.....</b>	<b>21</b>
<b>5.1 Architecture Design .....</b>	<b>21</b>
<b>5.2 Use-Case Diagram .....</b>	<b>23</b>
<b>5.3 Admin Level Function.....</b>	<b>24</b>
<b>5.4 Doctor Level Function.....</b>	<b>25</b>
<b>5.5 Patient Level Function.....</b>	<b>25</b>
<b>5.6 Patient or Doctor Registration.....</b>	<b>26</b>

5.7 User Login.....	26
5.8 Uploading files into the system.....	26
5.9 View files from the system .....	26
5.10 Table Design.....	27
Chapter 6 SYSTEM REQUIREMENTS.....	30
Chapter 7 MODULE DESCRIPTION.....	31
7.1 Module 1: Registration & Login .....	31
7.2 Module 2: Client-side Encryption using Hybrid algorithm.....	31
7.3 Module 3: Downloading & Decrypting.....	32
Chapter 8 IMPLEMENTATION.....	33
8.1 Steps to use the system.....	34
Chapter 9 CONCLUSION .....	39
APPENDIX.....	40
REFERENCES.....	46



## LIST OF FIGURES

SL NO	FIGURE NO	FIGURE DESCRIPTION	PAGE NO
1	1.1	CIA Triad	1
2	1.2	Key Schedule Algorithm	8
3	1.3	AES Algorithm	9
4	1.4	3DES Algorithm	10
5	1.5	RSA Algorithm	13
6	3.1	Existing System Encryption Phase	17
7	3.2	Existing System Decryption Phase	17
8	5.1	Encryption	21
9	5.2	Decryption	22
10	5.3	Actor and use-case diagram	23
11	5.4	Use-case diagram of the proposed project	24
12	5.5	Level 1 DFD – Admin Function	24
13	5.6	Level 1 DFD – Doctor Function	25
14	5.7	Level 1 DFD – Patient Function	25
15	5.8	Tables	27-29

## CHAPTER 1

### INTRODUCTION

#### 1.1 GENERAL BACKGROUND

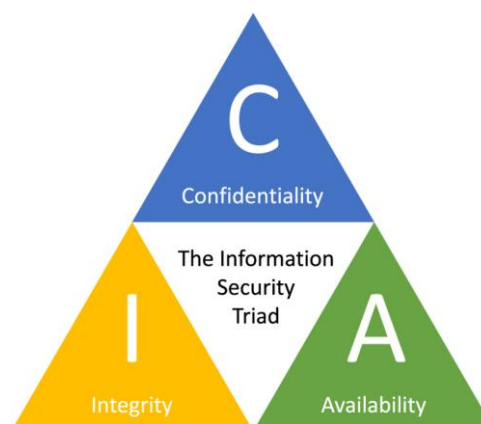
In the realm of e-healthcare services, the ability to offer remote medical assistance has been greatly enhanced, enabling quicker provision of first aid and medical care to patients. Central to these advancements are medical images, pivotal for facilitating prompt and accurate remote diagnoses and treatments. However, the transmission of these images over public networks poses significant security risks due to their sensitive nature. To mitigate these risks, it's imperative to secure medical images before their transmission over public networks

#### 1.2 INFORMATION SECURITY

Information security refers to the practice of protecting information from unauthorized access, disclosure, alteration, or destruction. It encompasses various strategies, policies, and technologies designed to safeguard sensitive data and ensure its confidentiality, integrity, and availability.

Ensuring the confidentiality and integrity of medical data, particularly when transmitting sensitive images over public networks, is critical. Measures like encryption and access controls safeguard patient information, upholding trust and compliance with regulations.

#### 1.3 PRINCIPLES OF INFORMATION SECURITY



**Fig:1.1 CIA Triad**

The CIA Triad, comprising Confidentiality, Integrity, and Availability, forms the cornerstone of information security. Confidentiality ensures data is accessible only to authorized users, Integrity maintains data accuracy and trustworthiness, while Availability ensures data remains accessible to authorized parties when needed. These principles guide the implementation of security measures, safeguarding sensitive information, mitigating risks, and ensuring compliance with regulations.

### **1.3.1 CONFIDENTIALITY**

Ensures that information is accessible only to authorized individuals or entities. Confidentiality measures prevent unauthorized access, disclosure, or exposure of sensitive data, safeguarding its privacy and confidentiality.

Confidentiality in the context of securing medical images is essential to protect patient's privacy and sensitive health information. Medical images often contain personal identifiers, such as names, dates of birth, and medical history, making them highly sensitive. Ensuring confidentiality prevents unauthorized access, disclosure, or exposure of these images to individuals or entities who are not involved in the patient's care. Unauthorized access could lead to breaches of patient confidentiality, identity theft, or other privacy violations, undermining trust in healthcare providers and potentially causing emotional distress or harm to patients.

### **1.3.2 INTEGRITY**

Focuses on maintaining the accuracy, consistency, and trustworthiness of data. Integrity controls ensure that data remains unchanged and uncorrupted throughout its lifecycle, preventing unauthorized alterations, tampering, or modifications. Integrity is crucial for securing medical images to maintain their accuracy, reliability, and trustworthiness. Medical images serve as critical diagnostic tools, guiding healthcare professionals in making informed decisions about patient care. Any unauthorized alteration, tampering, or corruption of these images could lead to misdiagnoses, incorrect treatments, or compromised patient safety. Ensuring integrity safeguards the authenticity and consistency of medical images throughout their lifecycle, from acquisition to transmission and storage. It helps prevent data manipulation or

unauthorized modifications, preserving the reliability of diagnostic information and upholding the quality of patient care

### 1.3.3 AVAILABILITY

Ensures that information is accessible and usable by authorized users whenever needed. Availability measures protect against disruptions, such as cyberattacks, natural disasters, or technical failures, which could otherwise result in data unavailability or downtime.

Availability is vital for securing medical images to ensure they are accessible to authorized users whenever needed for patient care or clinical decision-making. Delays or disruptions in accessing medical images could impede timely diagnoses, treatment planning, or emergency interventions, potentially jeopardizing patient outcomes. Guaranteeing availability involves implementing resilient infrastructure, redundant systems, and disaster recovery measures to mitigate the risk of downtime or data loss. By ensuring the continuous availability of medical images, healthcare providers can uphold the efficiency, effectiveness, and responsiveness of patient care services, ultimately improving patient outcomes and satisfaction.

## 1.4 CRYPTOGRAPHY

Cryptography is the science and practice of securing communication and information by transforming it into an unreadable format, known as ciphertext, using mathematical algorithms and techniques. Its primary goal is to ensure the confidentiality, integrity, and authenticity of data in various communication channels and storage systems. Cryptography plays a crucial role in modern information security, protecting sensitive information from unauthorized access, tampering, and interception

### Features of cryptography

1. Confidentiality: Cryptography ensures that only authorized parties can access and decipher encrypted data. By transforming plaintext into ciphertext using encryption algorithms and keys, sensitive information remains confidential and protected from eavesdroppers or unauthorized entities.
2. Integrity: Cryptography verifies the integrity of data by detecting any unauthorized modifications or alterations. Hash functions and digital signatures are cryptographic

techniques used to generate checksums or unique identifiers for data, enabling recipients to verify its authenticity and integrity.

3. **Authentication:** Cryptography provides mechanisms for verifying the identities of communicating parties. Digital signatures, certificates, and cryptographic protocols authenticate the origin and integrity of messages, ensuring that they come from trusted sources and have not been tampered with during transmission.
4. **Non-repudiation:** Cryptographic techniques offer non-repudiation, which prevents individuals from denying their involvement in a transaction or communication. Digital signatures provide proof of the origin and authenticity of messages, making it difficult for senders to disown their actions or deny responsibility.
5. **Key management:** Cryptography relies on secure key management practices to protect encryption keys and ensure their confidentiality and integrity. Key distribution, rotation, and revocation are essential aspects of key management, safeguarding encrypted data from unauthorized access or decryption.
6. **Versatility:** Cryptography offers a versatile range of algorithms and techniques tailored to specific security requirements and applications. From symmetric encryption for fast, efficient data protection to asymmetric encryption for secure key exchange and digital signatures, cryptography provides diverse tools to address various security challenges.

#### **1.4.1 TYPES OF CRYPTOGRAPHY**

Each type of cryptography serves distinct purposes and has its strengths and weaknesses, depending on the specific security requirements and use cases. Understanding the characteristics and applications of symmetric key cryptography, asymmetric key cryptography, and hash functions is essential for designing secure cryptographic systems and protocols.

##### **SYMMETRIC KEY CRYPTOGRAPHY**

Symmetric key cryptography, also known as secret key cryptography, involves the use of a single shared key for both encryption and decryption of data. The same key is used by both the sender and the recipient to encrypt and decrypt messages. Symmetric key algorithms are typically faster and more efficient than asymmetric key algorithms, making them suitable for encrypting large volumes of data. However, the challenge lies in securely distributing the shared key to authorized parties without interception or compromise.

##### **ASYMMETRIC KEY CRYPTOGRAPHY**

Asymmetric key cryptography, also known as public-key cryptography, utilizes a pair of mathematically related keys: a public key and a private key. The public key is widely distributed and used for encrypting messages, while the private key is kept secret and used for decrypting the encrypted data. Asymmetric key algorithms offer advantages in key distribution and digital signatures, enabling secure communication and authentication without the need for a pre-shared secret key. However, asymmetric key operations are computationally intensive, making them less efficient for bulk encryption compared to symmetric key algorithms.

### **HASH FUNCTIONS**

Hash functions are cryptographic algorithms that transform input data of arbitrary size into a fixed-size output, known as a hash value or digest. Hash functions are one-way functions, meaning that it is computationally infeasible to reverse the process and derive the original input data from the hash value. Hash functions are commonly used for data integrity verification, digital signatures, and password hashing. They provide a compact and unique representation of data, enabling efficient verification of data integrity and authenticity. Additionally, hash functions are resistant to collisions, where two different inputs produce the same hash value, ensuring the reliability of cryptographic operations.

#### **1.4.2 CLIENT-SIDE CRYPTOGRAPHY**

Client-side cryptography refers to cryptographic techniques and protocols that are implemented on the client side of a communication or data processing system. This approach enables end-to-end encryption and ensures that data is encrypted and decrypted locally on the client device, rather than relying solely on server-side encryption.

Client-side encryption involves encrypting data on the user's end before transmitting it to the server. In this model, the encryption and decryption processes occur locally, typically within the user's device or application. The encrypted data is then sent over the network to the server, where it remains encrypted until it's decrypted by the authorized user.

- **Enhanced Data Privacy:** One of the key benefits of client-side encryption is heightened data privacy. Since data is encrypted before leaving the user's device, it remains confidential even during transit and storage on the server. This minimizes the risk of unauthorized access, data breaches, or surveillance.
- **Reduced Trust Dependency:** With client-side encryption, users have greater control over their data since they manage the encryption keys. This reduces

reliance on third-party service providers and mitigates concerns about data exposure due to potential vulnerabilities or breaches on the server side.

- **Compliance and Regulatory Alignment:** Client-side encryption aligns well with stringent data protection regulations such as GDPR (General Data Protection Regulation) and HIPAA (Health Insurance Portability and Accountability Act), where organizations are mandated to implement robust data security measures to protect users' privacy.

### 1.5 Server-side Encryption:

Contrary to client-side encryption, server-side encryption involves encrypting data within the server environment. Encryption and decryption processes are managed by the server, and data is stored in encrypted form. Users typically interact with the server in plaintext, and encryption and decryption occur transparently without user involvement.

- **Simplified User Experience:** Server-side encryption offers a seamless user experience as encryption and decryption processes are abstracted away from the user. Users interact with data in its unencrypted form, and encryption operations are handled transparently by the server, minimizing complexity and potential user errors.
- **Centralized Management:** With server-side encryption, encryption keys and management are centralized within the server infrastructure. This facilitates key management and ensures consistency in encryption policies across the organization, making it easier to enforce security protocols and access controls.
- **Scalability and Performance:** Server-side encryption may offer better scalability and performance, particularly in scenarios involving large volumes of data. Since encryption operations are performed within the server environment, optimized hardware and software configurations can be employed to handle encryption tasks efficiently.

## 1.6 AES Algorithm

The Advanced Encryption Standard (AES) stands as a cornerstone in modern cryptographic protocols, providing a robust and efficient method for encrypting sensitive data. Adopted by the U.S. government in 2001, AES has become the de facto standard for securing data across various applications, including communication, storage, and authentication. Let's delve into how AES works and its key features, accompanied by illustrative figures.

AES operates on the principles of symmetric-key cryptography, where the same secret key is used for both encryption and decryption. It employs a block cipher algorithm, meaning it encrypts data in fixed-size blocks. The AES algorithm supports key sizes of 128, 192, or 256 bits, with 128-bit keys being the most commonly used.

Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.

Points to remember

- AES is a block cipher.
- The key size can be 128/192/256 bits.
- Encrypts data in blocks of 128 bits each.

That means it takes 128 bits as input and outputs 128 bits of encrypted cipher text as output. AES relies on substitution-permutation network principle which means it is performed using a series of linked operations which involve replacing and shuffling of the input data.

Working of the cipher :

AES performs operations on bytes of data rather than in bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of the input data at a time.

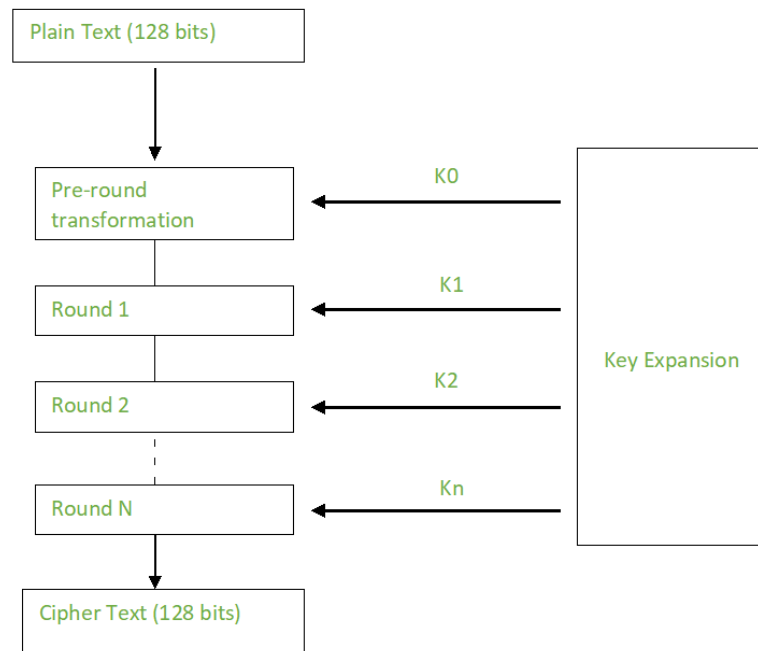
The number of rounds depends on the key length as follows :

- 128-bit key – 10 rounds
- 192-bit key – 12 rounds
- 256-bit key – 14 rounds

Creation of Round keys :



A Key Schedule algorithm is used to calculate all the round keys from the key. So the initial key is used to create many different round keys which will be used in the corresponding round of the encryption.



**Fig 1.2 Key Schedule Algorithm**

### Encryption:

**Key Expansion:** The process begins with key expansion, where the original encryption key is transformed into a series of round keys. These round keys are used in each round of the encryption and decryption processes.

**Initial Round:** In the initial round, the input data (plaintext) is XORed with the first round key.

**Rounds:** AES operates through a series of rounds, with the number of rounds depending on the key size. For AES-128, there are 10 rounds; for AES-192, there are 12 rounds; and for AES-256, there are 14 rounds. Each round consists of several transformation steps:

**SubBytes:** SubBytes involves substituting each byte of the state (input data or intermediate data) with a corresponding byte from the S-box (substitution box), which is a predefined lookup table.

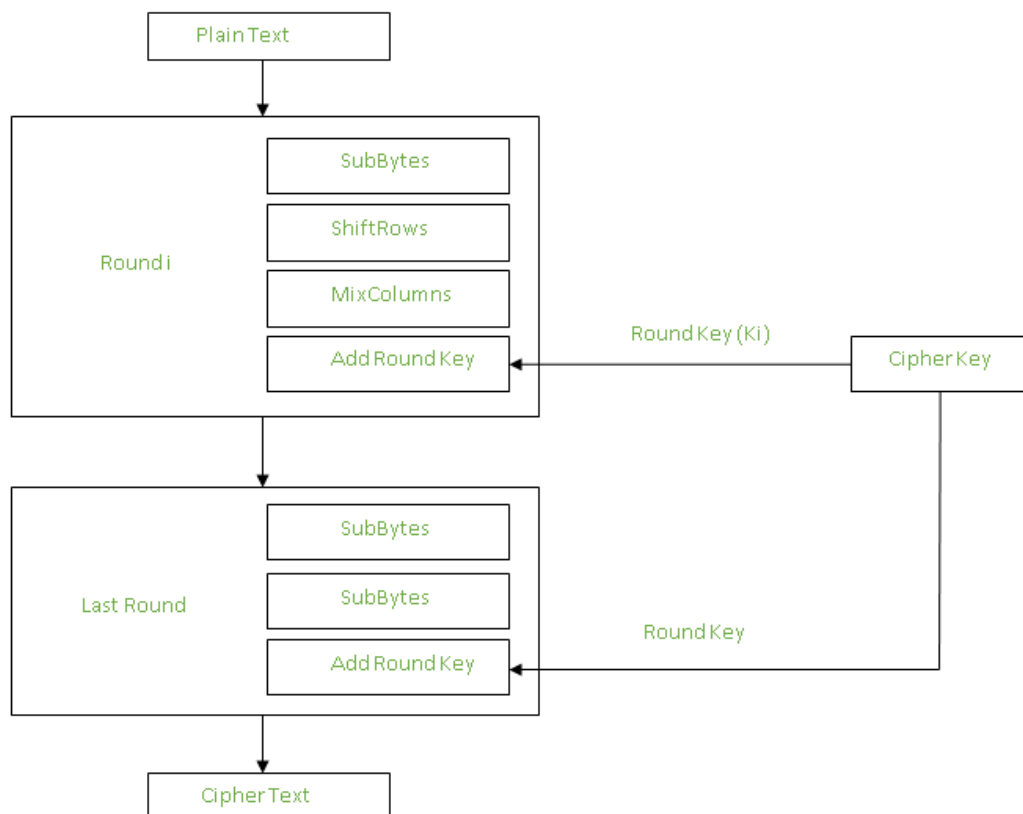
**ShiftRows:** ShiftRows involves shifting the rows of the state matrix to the left by varying offsets. This step ensures diffusion and confusion within the data.

**MixColumns:** MixColumns operates on each column of the state matrix, using a matrix multiplication operation that further enhances diffusion and confusion.

**AddRoundKey:** AddRoundKey XORs the state with the round key for that specific round, introducing the key's influence into the encryption process.

**Final Round:** The final round excludes the MixColumns step, focusing on SubBytes, ShiftRows, and AddRoundKey.

**Output:** After completing all rounds (including the final round), the resulting data is the ciphertext (encrypted data).



**Fig:1.3 AES Algorithm**

After all these rounds 128 bits of encrypted data are given back as output. This process is repeated until all the data to be encrypted undergoes this process.

Decryption :

The stages in the rounds can be easily undone as these stages have an opposite to it which when performed reverts the changes. Each 128 blocks goes through the 10,12 or 14 rounds depending on the key size.

The stages of each round of decryption are as follows :

- Add round key

- Inverse MixColumns
- ShiftRows
- Inverse SubByte

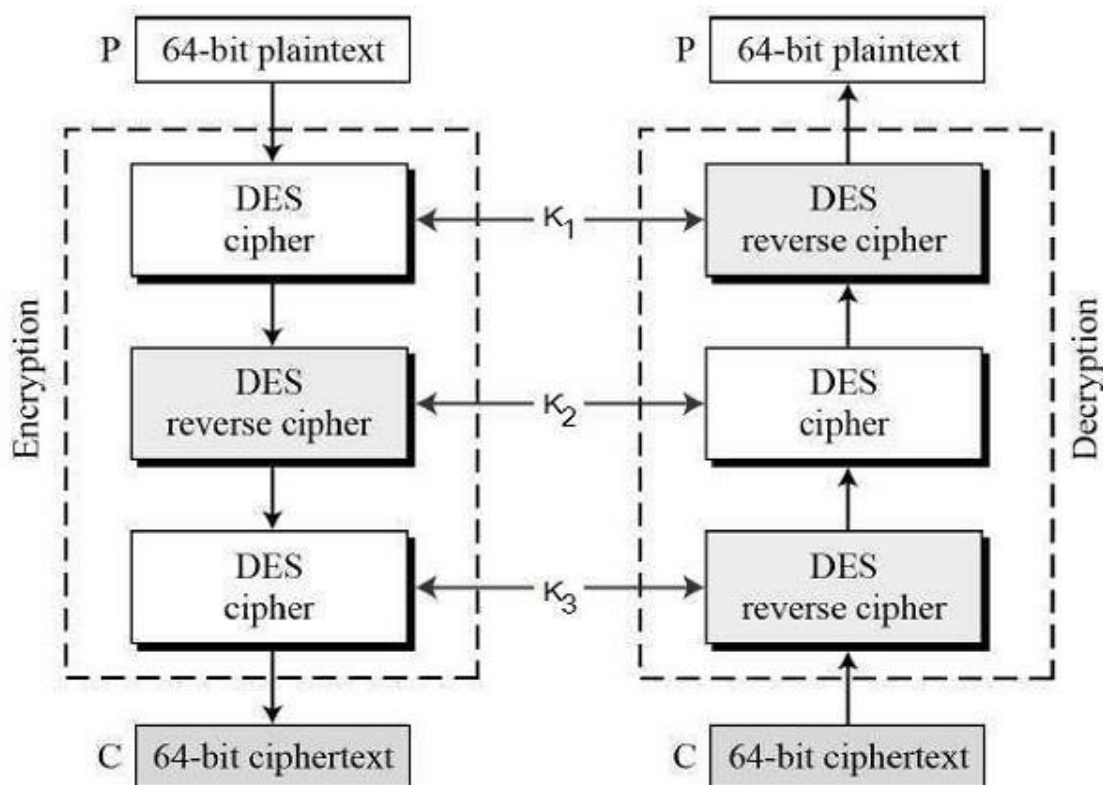
### 1.7 Triple DES Algorithm

The speed of exhaustive key searches against DES after 1990 began to cause discomfort amongst users of DES. However, users did not want to replace DES as it takes an enormous amount of time and money to change encryption algorithms that are widely adopted and embedded in large security architectures.

The pragmatic approach was not to abandon the DES completely but to change how DES is used. This led to the modified schemes of Triple DES (sometimes known as 3DES). Incidentally, there are two variants of Triple DES known as 3-key Triple DES (3TDES) and 2-key Triple DES (2TDES).

#### 3-KEY Triple DES

Before using 3TDES, the user first generates and distributes a 3TDES key  $K$ , which consists of three different DES keys  $K_1$ ,  $K_2$ , and  $K_3$ . This means that the actual 3TDES key has a length of  $3 \times 56 = 168$  bits. The encryption scheme is illustrated as follows –



**Fig: 3DES Algorithm**

The encryption-decryption process is as follows –

- Encrypt the plaintext blocks using a single DES with key  $K_1$ .
- Now decrypt the output of step 1 using a single DES with key  $K_2$ .
- Finally, encrypt the output of step 2 using a single DES with key  $K_3$ .
- The output of step 3 is the ciphertext.
- Decryption of a ciphertext is a reverse process. The user first decrypts using  $K_3$ , then encrypts with  $K_2$ , and finally decrypts with  $K_1$ .

Due to this design of Triple-DES as an encrypt–decrypt–encrypt process, it is possible to use a 3TDES (hardware) implementation for a single DES by setting  $K_1$ ,  $K_2$ , and  $K_3$  to be the same value. This provides backward compatibility with DES.

The second variant of Triple DES (2TDES) is identical to 3TDES except that  $K_3$  is replaced by  $K_1$ . In other words, the user encrypts plaintext blocks with key  $K_1$ , then decrypts with key  $K_2$ , and finally encrypts with  $K_1$  again. Therefore, 2TDES has a key length of 112 bits.

Triple DES systems are significantly more secure than single DES, but these are a much slower process than encryption using single DES.

## 1.8 RSA Algorithm

RSA algorithm is an asymmetric cryptography algorithm. Asymmetric means that it works on two different keys i.e. Public Key and Private Key. As the name describes the Public Key is given to everyone and the Private key is kept private.

Since this is asymmetric, nobody else except the browser can decrypt the data even if a third party has the public key of the browser. The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is a multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So, if somebody can factorize the large number, the private key is compromised. Therefore, encryption strength lies in the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024-bit keys could be broken soon. But till now it seems to be an infeasible task.

When using RSA for encryption and decryption of general data, it reverses the key set usage. Unlike signature verification, it uses the receiver's public key to encrypt the data, and it uses the receiver's private key to decrypt the data. Thus, there is no need to exchange any keys in this scenario.

There are two broad components when it comes to RSA cryptography, they are:

**Key Generation:** Generating the keys to be used for encrypting and decrypting the data to be exchanged.

**Encryption/Decryption Function:** The steps that need to be run when scrambling and recovering the data.

### Key Generation

You need to generate public and private keys before running the functions to generate your ciphertext and plaintext. They use certain variables and parameters, all of which are explained below:

Choose two large prime numbers ( $p$  and  $q$ )

Calculate  $n = p * q$  and  $z = (p-1)(q-1)$

Choose a number  $e$  where  $1 < e < z$

Calculate  $d = e^{-1} \bmod (p-1)(q-1)$

You can bundle private key pair as  $(n, d)$

You can bundle public key pair as  $(n, e)$

### Encryption/Decryption Function

Once you generate the keys, you pass the parameters to the functions that calculate your ciphertext and plaintext using the respective key.

If the plaintext is  $m$ , ciphertext =  $me \bmod n$ .

If the ciphertext is  $c$ , plaintext =  $cd \bmod n$

To understand the above steps better, you can take an example where  $p = 17$  and  $q = 13$ . The value of  $e$  can be 5 as it satisfies the condition  $1 < e < (p-1)(q-1)$ .

$N = p * q = 221$

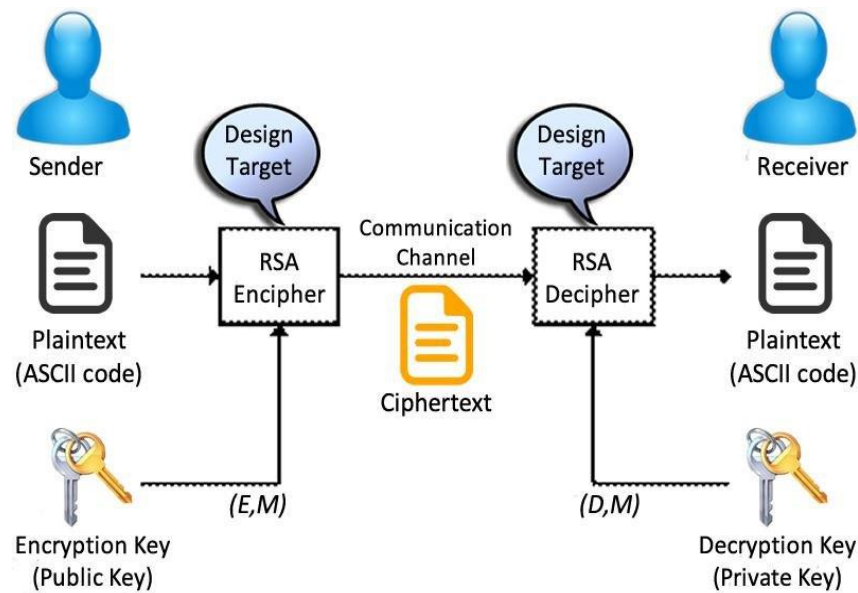
$d = e^{-1} \bmod (p-1)(q-1) = 29$

Public Key pair =  $(221, 5)$

Private Key pair =  $(221, 29)$

If the plaintext( $m$ ) value is 10, you can encrypt it using the formula  $me \bmod n = 82$ .

To decrypt this ciphertext( $c$ ) back to the original data, you must use the formula  $cd \bmod n = 29$ .



**Fig:1.5 RSA Algorithm**

## 1.9 OBJECTIVES

The main objective of this project is to enhance the security of remote medical assistance by addressing the vulnerability of transmitting medical images over public networks. These images are crucial for accurate diagnoses and treatments but are susceptible to security risks due to their sensitive nature. To mitigate these risks, the project proposes a hybrid encryption scheme tailored specifically for securing medical images. By combining the strengths of symmetric and asymmetric encryption algorithms, the system aims to achieve a balance between efficiency and security. Symmetric encryption is utilized for bulk data encryption, ensuring speed, while asymmetric encryption is employed for key exchange and confidentiality. Additionally, the system incorporates SHA2 to verify the integrity of encrypted images, adding an extra layer of security. Given the unique characteristics of medical images, such as high correlation and redundancy among pixels and larger file sizes, the project aims to develop an efficient encryption model capable of withstanding various security threats. Overall, the objective is to safeguard patient privacy and confidentiality while enabling the swift and secure transmission of medical images for remote diagnoses and treatments in e-healthcare services.

## CHAPTER 2

### LITERATURE SURVEY

Cryptography refers to techniques of securing information and communication derived from mathematical perception to convert messages in ways that are tough to interpret. Cryptography is firmly associated with the department of cryptology along with cryptanalysis. It consists of techniques such as blending words with images, microdots, and alternative ways to mask data during storage or else transit.

This paper introduces a novel hybrid cryptography technique combining AES and Blowfish algorithms, blending symmetric and symmetric encryption for heightened security. Each user possesses a private key for decryption, ensuring simultaneous access for multiple individuals. Additionally, the method employs MD5 hashing for key hashing during encryption and decryption, enhancing key security and overall hybrid cryptography security. The resulting ciphertext is virtually indecipherable without the intended recipient's decryption capabilities, strengthening data confidentiality [\[1\]](#).

This paper explores the realm of cryptography and its connection to modern cloud computing challenges. Traditional cryptography techniques are discussed, highlighting their evolution and association with securing data in cloud environments. Previous research focused on symmetric algorithms for cloud data security, but their simplicity led to performance issues. Our research introduces a robust symmetric key encryption approach, enhancing security by encrypting files locally on the client side before uploading them to the cloud. Upon download, decryption occurs using the encryption-generated key. This method not only bolsters security but also improves performance, especially for large files. By adding this extra layer of security, we aim to mitigate potential attacks on sensitive information and address standardization concerns in cloud security protocols [\[2\]](#).

This paper describes an approach and implementation issues of client-side data encryption for thin client–web browsers. Since the service provider may be a competitor of the service user, sensitive parts of data are encrypted and decrypted by the client. Cipher key generation and distribution issues are discussed: although the cipher key is kept private, the paper shows the way for the key validation by the server side to maintain data consistency [\[3\]](#).

This thesis introduces an approach for designing secure web applications using client-side encryption to protect user data from web server compromises. It also presents CryptFrame, a set of tools facilitating the development of such applications. CryptFrame enables encryption and decryption of sensitive data within the user's browser, with keys securely stored in a browser extension to prevent unauthorized access. It employs templated verification for controlled access to plaintext data by trusted client-side code, ensuring security. Additionally, CryptFrame includes a principal graph feature for safe management of shared data permissions, even in the presence of active adversaries [\[4\]](#).

Another paper introduces a novel approach to client-side encryption in web servers using hybrid cryptography. The scheme aims to enhance data security and access control by combining

symmetric and asymmetric encryption techniques, providing a robust solution for protecting sensitive data [\[5\]](#).

Focusing on key management challenges, this work proposes strategies for efficiently handling encryption keys in web server environments. By leveraging hybrid cryptography techniques, the paper aims to reduce computational overhead and streamline key management processes [\[6\]](#).

This paper presents a hybrid cryptography-based solution to improve data confidentiality in web servers. By combining encryption methods, the proposed approach ensures enhanced data protection, contributing to a more secure web server environment [\[7\]](#).

Addressing the need for secure data sharing, this paper discusses the application of hybrid encryption techniques in web servers. The proposed approach emphasizes data privacy and integrity, facilitating the secure sharing of sensitive information [\[8\]](#).

Focusing on client-side security, this paper introduces a hybrid encryption approach for web servers. By addressing key management challenges, the proposed scheme aims to enhance security while ensuring secure data access for authorized users [\[9\]](#).

This study evaluates the performance of hybrid cryptography techniques for client-side encryption in web servers. Considering factors such as computational overhead and data security, the paper provides insights into the effectiveness of hybrid encryption in web server environments [\[10\]](#).



## CHAPTER 3

### EXISTING SYSTEM

The existing research work introduced a system using a symmetric key algorithm that would help encrypt sensitive information. In this paper, they introduce a hybrid technique for Cryptography using two algorithms (AES and Blowfish). This technique combines symmetric and another symmetric encryption which would give a high security and it also includes a private key where everyone can use the key for the decryption process at the same time. This technique also benefits making hashing for the Key by using the MD5 hashing function that will hash the key in the encryption process and make the same process in decryption. This will increase the security of the key plus increasing security using hybrid cryptography. Finally, they will have cipher text that cannot be decrypted except by the recipient.

In the encryption phase, plaintext is divided into blocks and encrypted using both AES and Blowfish algorithms simultaneously. The results are concatenated to create encrypted blocks, while the encryption key undergoes MD5 hashing for added security.

During decryption, ciphertext blocks are decrypted using AES and Blowfish algorithms with the hashed key. This integration of AES, Blowfish, and MD5 hashing ensures robust encryption and decryption processes, offering enhanced security against unauthorized access. It provides constancy and flexibility that allows users to store as well as retrieve images at any moment and from any place using the web.

#### 3.1 Drawbacks of Existing System

- The encryption key is stored in the device's memory, making it vulnerable to theft.
- The existing system is only applicable to textual data
- It cannot handle or process files other than text files
- Performance is low for large files.
- Symmetric key cryptography carries a high risk around key transmission
- Every time the key gets shared, the risk of interception by an unintended third party exists
- It is not implemented in the cloud environment.

### 3.2 functional workflow of existing system

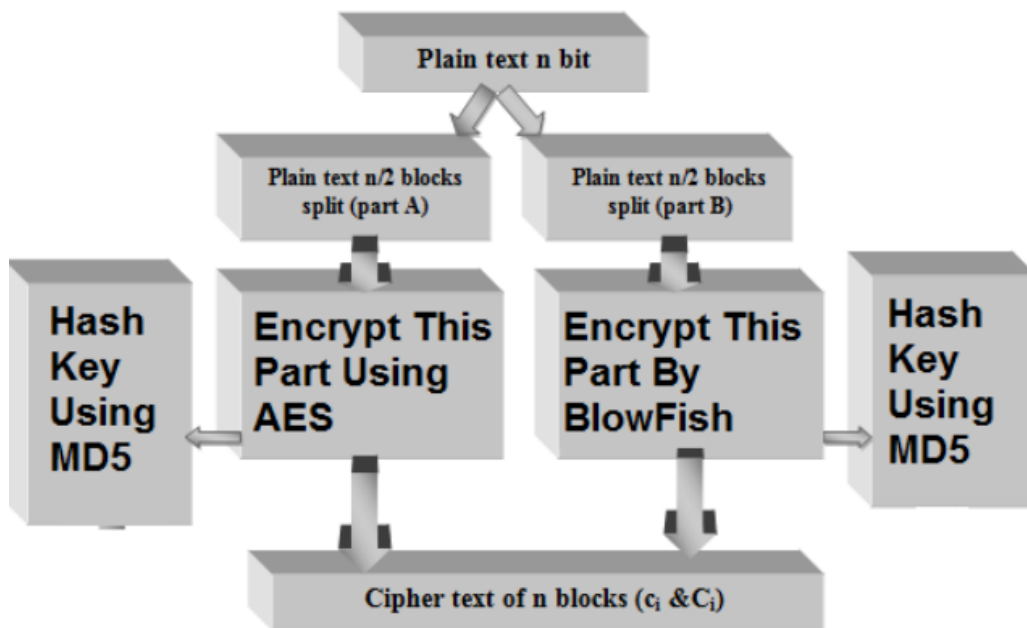


Fig:3.1 Existing System Encryption Phase

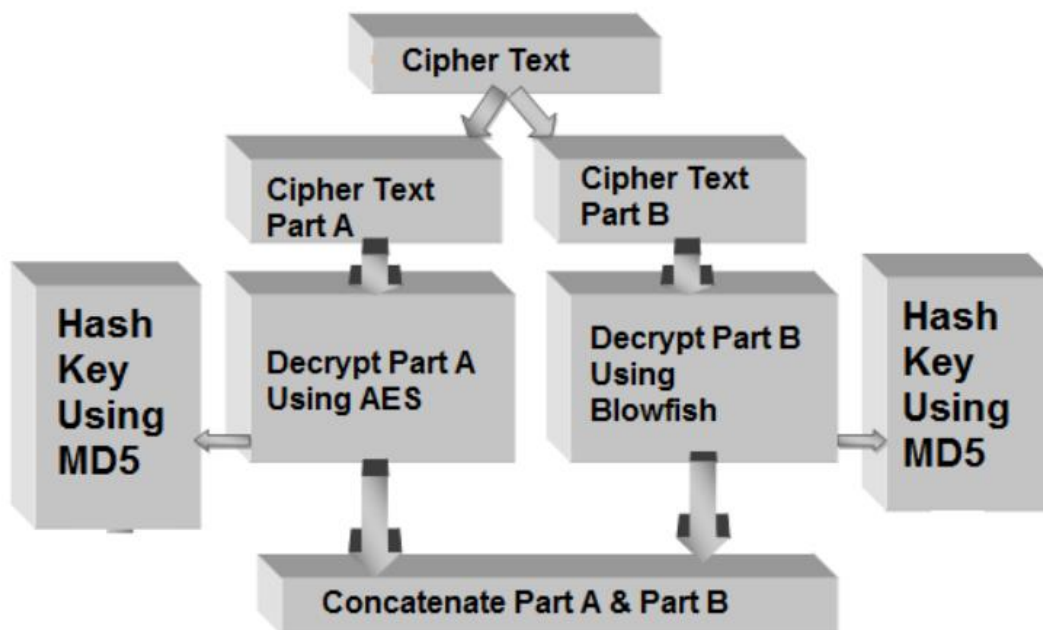


Fig:3.2 Existing System Decryption phase

## Chapter 4

### PROPOSED SYSTEM

The existing system is implemented to secure data before uploading to the web server using a client-side cryptographic method. This uses a symmetric cryptographic algorithm. Though this system provides security to images stored in the web server, hybrid cryptography proves to be more effective and difficult to crack. While Asymmetric Encryption can be used to encrypt secrets to a public key. Hybrid Encryption can be used to encrypt large amounts of images to a public key. So here the proposed system uses a combination of symmetric and asymmetric cryptographic algorithms to encrypt images on the client side using the public key before uploading to the web server.

The encryption is done by dividing the picture into two equal parts, the first part is encrypted by the AES algorithm and the second part is done using Triple DES. These parts are further encrypted using RSA to provide further security. This technique also benefits making hashing for the document by using the SHA256 hashing function that will hash the image value in the encryption process and make the same process in decryption. These can be used to ensure the integrity of data.

When encryption is done, after completing the combination of symmetric and asymmetric algorithms, the encrypted file is uploaded to the Web server. The uploaded encrypted file has to go through web server-side encryption once again. Then we download the file from the web server to the storage and decrypt the image file by applying the private keys. When decryption is done, we get back our desired image which was secure in the web server as well as during transmission. The system introduces a portal through which the admin can register patients and doctors followed by an OTP verification, which is an authentication technique to provide extra security. Then after logging in the admin can upload the file to the server which will be encrypted on the client side. Then the patient and doctor can decrypt the file by receiving another mail containing the OTP that is sent to the user's mail-id for decryption. By providing the OTP, the user completes the authentication, and the decrypted file will get downloaded.

AES (Advanced Encryption Standard) is widely recognized for its strong security and efficiency. Its benefits lie in its ability to provide high levels of encryption while remaining computationally efficient, making it suitable for encrypting large amounts of data quickly.

AES has been extensively studied and analyzed by cryptographers, and its adoption as the standard encryption algorithm by governments and organizations worldwide speaks to its reliability and trustworthiness.

3DES (Triple Data Encryption Standard) is a symmetric encryption algorithm that applies the DES algorithm three times to each data block. Despite being older than AES and having a slower encryption speed, 3DES is still widely used due to its proven security and backward compatibility with legacy systems. Its benefit lies in its resilience against cryptanalysis and its continued support in various applications and industries, making it a dependable choice for encrypting sensitive data.

RSA (Rivest-Shamir-Adleman) is an asymmetric encryption algorithm that relies on the mathematical properties of prime numbers. Its primary benefit lies in its ability to provide secure key exchange and digital signatures without requiring a shared secret key between communicating parties. RSA is widely used in public-key cryptography systems, such as SSL/TLS for securing internet communications and digital signatures for verifying the authenticity of messages or documents. Its security is based on the computational difficulty of factoring large prime numbers, making it highly resistant to attacks when implemented correctly.

Combining these encryption algorithms enhances security by leveraging the strengths of each. AES provides efficient encryption of data, 3DES offers backward compatibility and proven security, and RSA facilitates secure key exchange and digital signatures. By employing a combination of AES-3DES-RSA for client-side encryption, organizations can benefit from a robust and versatile approach to data security, ensuring strong protection against various threats while accommodating diverse encryption needs.

Utilizing AES, 3DES, and RSA for client-side encryption on web servers presents distinct advantages tailored to diverse security requirements. AES, renowned for its efficiency and robustness, ensures rapid encryption and decryption processes, making it ideal for large-scale data encryption. Its ability to handle varying key sizes provides flexibility in addressing different security needs. On the other hand, 3DES, despite being slower than AES, offers backward compatibility and proven resistance to cryptanalysis, making it suitable for legacy systems or environments requiring stringent security standards.

Additionally, RSA excels in asymmetric encryption, enabling secure key exchange and digital signatures. Its reliance on public and private key pairs enhances security without sacrificing performance. By leveraging these cryptographic algorithms for client-side encryption, web servers can uphold data confidentiality, integrity, and authenticity, safeguarding sensitive information from unauthorized access and cyber threats.

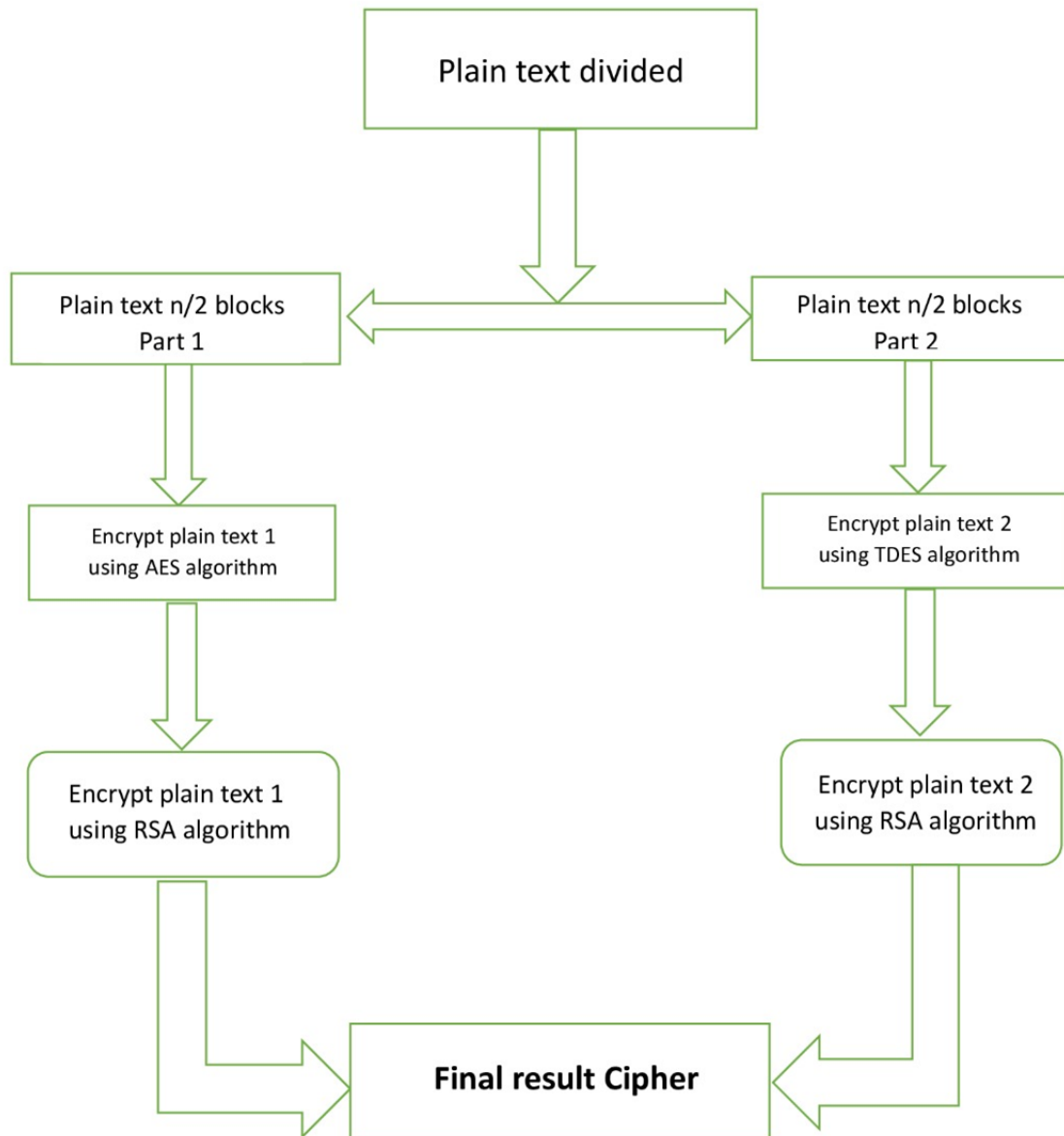
#### **4.1 Merits of the proposed system**

- It overcomes the biggest flaw of symmetric key cryptography since it doesn't need to exchange any keys that can decrypt data.
- Enhanced the encryption algorithm to support data types like images
- Optimize the encryption algorithm and underlying implementation to improve performance, especially for handling large files.
- Integrating asymmetric key cryptography eliminates the need for sharing symmetric encryption keys directly and mitigates the risk of interception by third parties
- Hackers can't modify data during transmission since doing so will prevent the receiver's private key from decrypting the message, thus informing the receiver that the message has been meddled with.

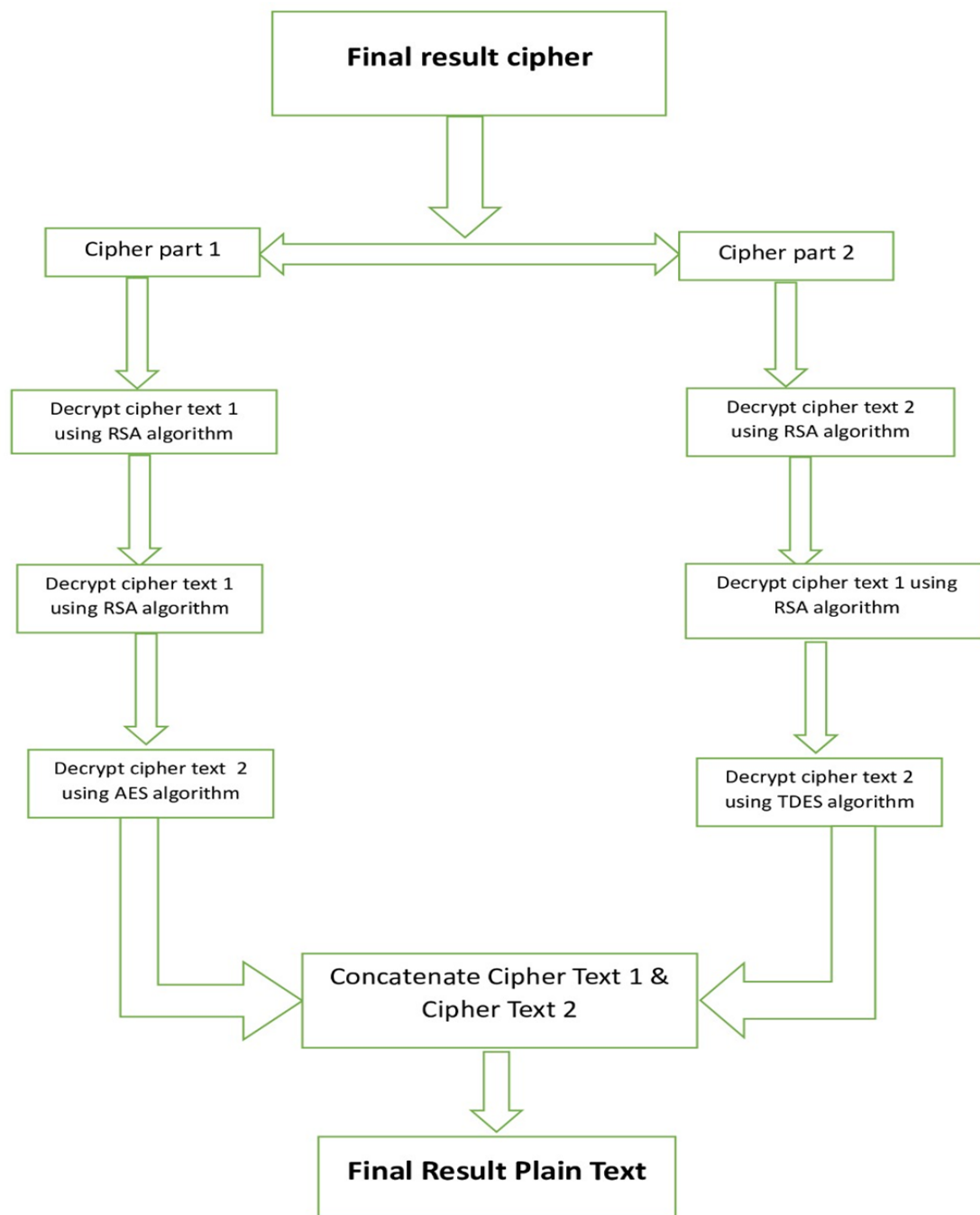
## Chapter 5

### SYSTEM DESIGN AND ARCHITECTURE

#### 5.1 Architecture Diagram



**Fig 5.1 Encryption**

**Fig:5.2 Decryption**

## 5.2 USE CASE DIAGRAM

Use cases help to determine the functionality and features of the software from user's perspective. A use case describes how a user interacts with the system by defining the steps required to accomplish a specific goal. Variations in the sequence of steps describe various scenarios. In the diagram, the stick figure represents an actor that is associated with one category of user. In the use-case diagram, the use cases are displayed as ovals. A use case is a set of scenarios that describe an interaction between a user and a system. A use case diagram displays the relationship between actors and use cases. The two main components of a use case diagram are use cases and actors.

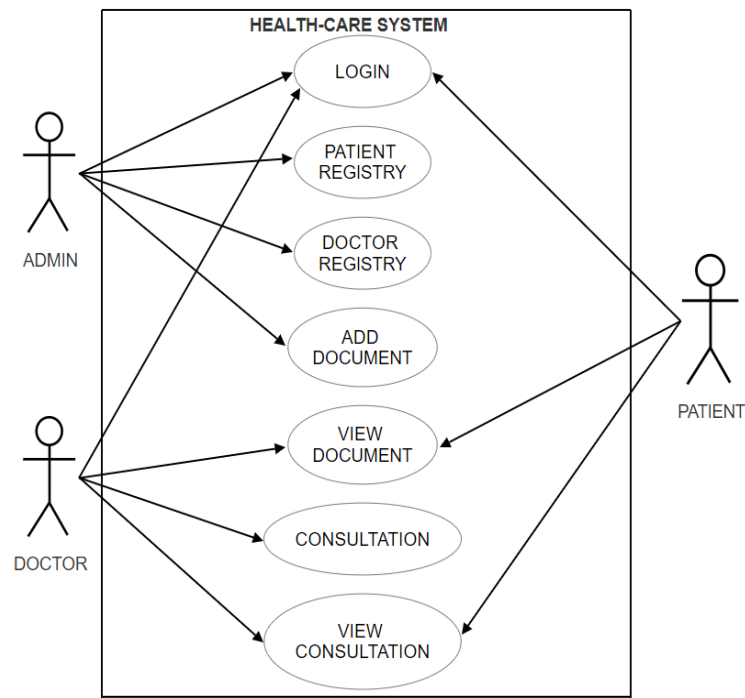


**Fig:5.3 Actor and use case diagram**

The actors are connected by lines to the use cases that they carry out. The use cases are placed in a rectangle but the actors are not. This rectangle is a visual reminder of the system boundaries and the actors are outside the system. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified. When the initial task is complete, use case diagrams are modelled to present the outside view. In brief, the purposes of use case diagrams can be said to be as follows

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements is actor.



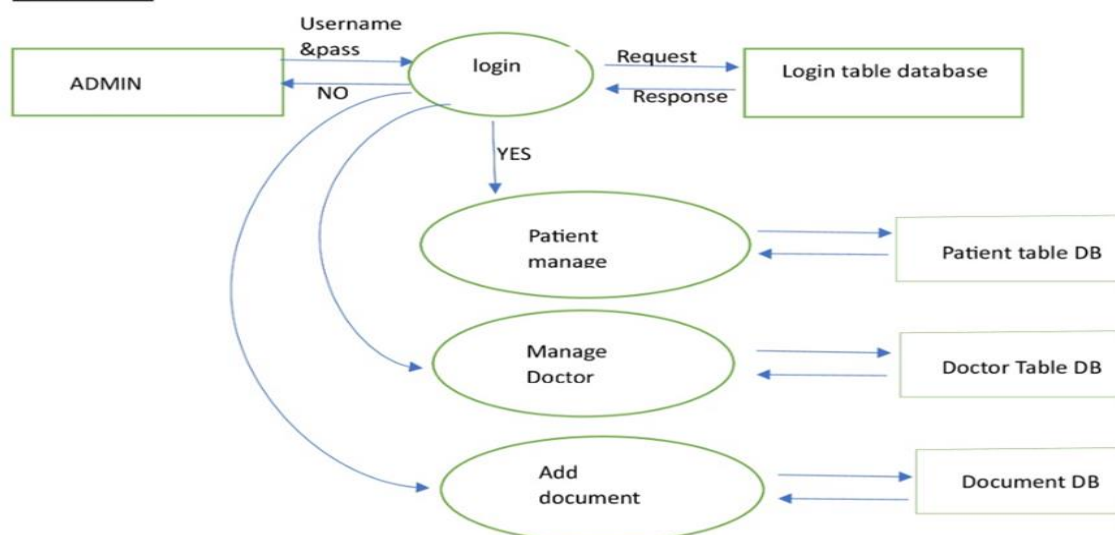


**Fig:5.4 Use-Case Diagram**

### 5.3 Admin Level Function

The system provides several functions for the admin to perform. Each component is connected to the main database of the system. The functions are:

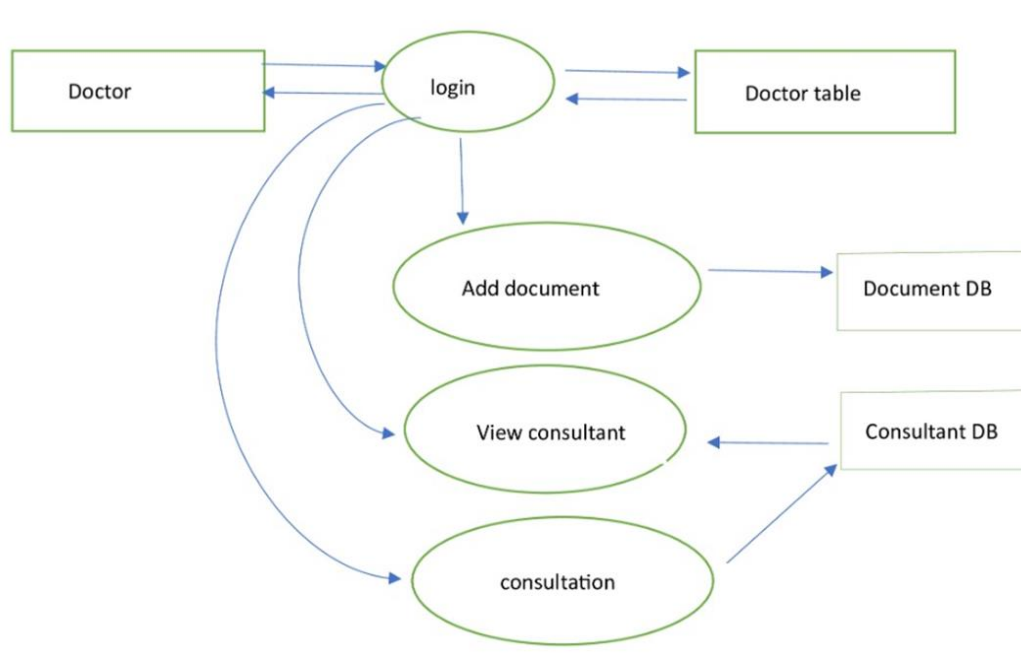
#### **LEVEL 1**



**Fig:5.5 Level 1 DFD – Admin Function**

## 5.4 Doctor Level Function

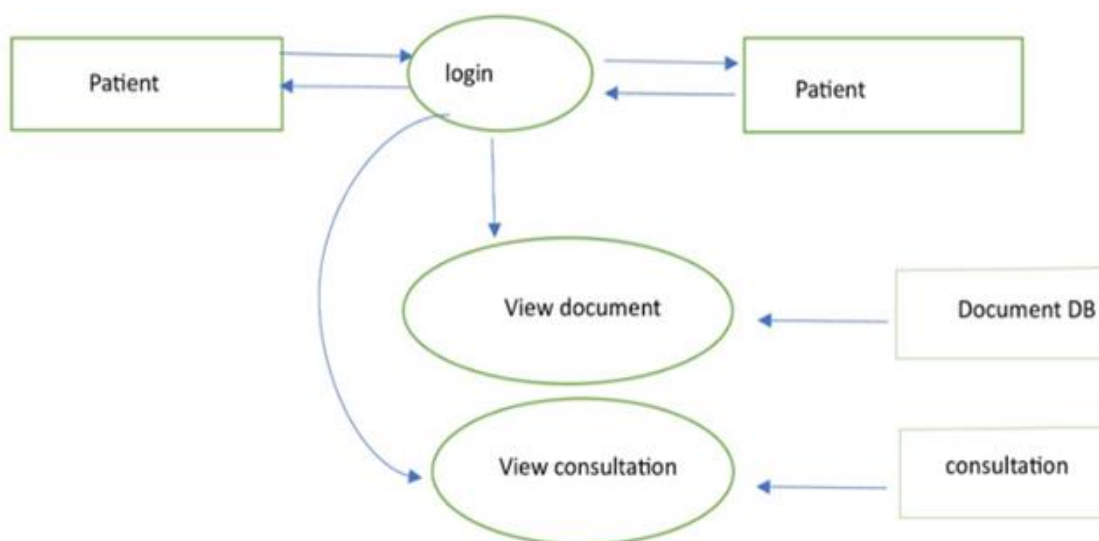
The system provides several functions for the admin to perform. Each component is connected to the main database of the system. The functions are:



**Fig:5.6 Level 1 DFD – Doctor Function**

## 5.5 Patient Level Function

The system provides several functions for the admin to perform. Each component is connected to the main database of the system. The functions are:



**Fig:5.7 Level 1 DFD – Patient Function**

## **5.6 PATIENT OR DOCTOR REGISTRATION**

An account for the patient or doctor is created by the Admin. A patient/Doctor can sign in by providing the necessary details like username, password, etc on the login page given by the admin. After the registration, an account for the user is successfully created by the system.

## **5.7 USER LOGIN**

The registered doctor/patient/admin can log in or sign up to the system. To log in/sign up, the doctor/patient/admin needs to give the necessary credentials like the username and password. Upon successful entry of the details, the user is logged in to the system where the user can access the various resources present in it.

## **5.8 UPLOADING FILES INTO THE SYSTEM**

The admin and the Doctors are the only ones with upload permission. Others can only view the file uploaded in their name. The file selected gets encrypted using a hybrid algorithm containing AES-3DES-RSA. The encrypted files get stored in the web server and an OTP is sent to the registered mail ID for decrypting it after it has been taken out of the server for decryption.

## **5.9 VIEW FILES FROM THE SYSTEM**

The doctor/patient/admin requests a file. The doctor/patient/admin has to click on “view the Document”. The doctor/patient/admin will then receive mail containing the private key. The user has to enter the key received to view the file, they are not downloadable and can only be viewed. If the code enters and the private key matches with the values stored in the database the file gets decoded

## 5.10 TABLE DESIGN

This is one of the major tasks is designing the database. It is important to realize that the design of the system is interrelated so table design of the system is interrelated and so table design cannot be considered in isolation from inputs, outputs, procedures, codes, and security requirements. In this project, the database has to maintain all the information about the username, password, tolerance, picture sequence, sound clips click points, etc.

### 5.9.1 DATABASE TABLE

#### bill\_product

Column	Type	Null	Default	Comments	MIME
id	int(11)	No			
item_description	varchar(100)	No			
price	varchar(10)	No			
tax	varchar(10)	No			
description	text	No			

#### Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	2	A	No	

#### category

Column	Type	Null	Default	Comments	MIME
id	int(50)	No			
name	varchar(100)	No			

#### Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	0	A	No	

#### consultation

Column	Type	Null	Default	Comments	MIME
id	int(11)	No			
reference_no	varchar(100)	No			
consultation_details	text	No			
disease	varchar(100)	No			
medicine	varchar(100)	No			
test	varchar(255)	No			
status	varchar(100)	No			
recover	varchar(100)	No			
doctor_id	varchar(100)	No			
datetime	timestamp	No	CURRENT_TIMESTAMP		

#### Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	0	A	No	

**doctor**

Column	Type	Null	Default	Comments	MIME
id	int(11)	No			
name	varchar(100)	No			
email	varchar(100)	No			
mobile	varchar(100)	No			
password	varchar(100)	No			
qualification	text	No			
specialization	varchar(100)	No			

**Indexes**

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
id	BTREE	Yes	No	id	2	A	No	

**document**

Column	Type	Null	Default	Comments	MIME
id	int(11)	No			
patient_id	varchar(100)	No			
file_title	varchar(250)	No			
file_path	varchar(255)	No			
hmac	varchar(255)	No			
date_time	timestamp	No	CURRENT_TIMESTAMP		

**Indexes**

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	6	A	No	

**patient**

Column	Type	Null	Default	Comments	MIME
id	int(11)	No			
patient_regno	varchar(100)	No			
name	varchar(100)	No			
address	varchar(255)	No			
email	varchar(100)	No			
dob	date	No			
mobile	varchar(20)	No			
password	varchar(100)	No			
doctor	varchar(100)	No			

**Indexes**

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	2	A	No	
patient_regno	BTREE	Yes	No	patient_regno	2	A	No	

**sales\_child**

Column	Type	Null	Default	Comments	MIME
id	int(50)	No			
om_id	int(50)	No			
item_id	int(50)	No			
order_qty	int(50)	No			
price	int(100)	No			

**Indexes**

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	0	A	No	

**sales\_entry**

Column	Type	Null	Default	Comments	MIME
id	int(11)	No			
bill_no	varchar(10)	No			
product	varchar(100)	No			
piece_set	varchar(100)	No			
total_set	varchar(100)	No			
geting_price	varchar(100)	No			
mrp	varchar(100)	No			
tax	varchar(100)	No			
user	varchar(100)	No			

**Indexes**

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	4	A	No	

**staff**

Column	Type	Null	Default	Comments	MIME
id	int(11)	No			
name	varchar(100)	No			
address	varchar(100)	No			
username	varchar(100)	No			
password	varchar(100)	No			

**Fig:5.8.Tables**

## Chapter 6

### SYSTEM REQUIREMENT

#### HARDWARE REQUIREMENTS

- |  |                      |
|--|----------------------|
| <input type="checkbox"/> CPU             | - I3 Processors      |
| <input type="checkbox"/> Hard Disk Space | - 80Gband above      |
| <input type="checkbox"/> Display         | - 15   Color Monitor |
| <input type="checkbox"/> Main Memory     | - 8 Gb               |
| <input type="checkbox"/> Keyboard        | - 104 Keys           |
| <input type="checkbox"/> Clock-Speed     | - 2.6 Ghz            |
| <input type="checkbox"/> Monitor         | - 15 " Svga Colo     |

#### SOFTWARE REQUIREMENTS

- |   |                         |
|---|-------------------------|
| <input type="checkbox"/> Operating System | - Windows 10            |
| <input type="checkbox"/> Backend          | - PHP, MYSQL            |
| <input type="checkbox"/> Front End        | - HTML, CSS, JAVASCRIPT |
| <input type="checkbox"/> IDE              | - Notepad++             |
| <input type="checkbox"/> Tool Kit         | - XAMPP                 |

## Chapter 7

### MODULE DESCRIPTION

The “securing medical image” System consists of 3 modules:

1. Registration & login
2. Upload and Client-side Encryption using a hybrid algorithm
3. Decryption and Download

#### 7.1 Module I: Registration & Login

This section includes the front end of the "secure medical image". The front end consists of a login page, where there is an option to log in for Admin, doctor, or patient. The registration of doctors and patients is done by admin. The three of them have different levels of access.

The admin can view all the users, register a user, upload a file, and download a file. The doctor can also upload and download the files of the patient they are assigned to. The patients can only view the file after authentication.

#### 7.2 Module 2: Client-side Encryption using hybrid algorithm and Uploading

The admin and the doctor have the privilege to upload and download files. while patients can only view the document. When they click on the “upload file” option a private key is made available through email to authenticate the user. They can get their private key which is required to download the files through their registered email ID when they click on the 'view document' option available in the document section. In the upload section, they can browse their files from their system and upload the file, so that the file gets encrypted before uploading it to the web server.

The cryptographic algorithm used here is the combination of the AES-3DES-RSA algorithm. In the encryption Phase, First, split the image into two equal parts. This image could be in any common format such as JPEG, PNG, or GIF. Determine the dimensions of the image, including its width and height. This information is crucial for accurately dividing the image.

The First portion of the image uses a combination of AES (Advanced Encryption Standard) for symmetric encryption and RSA (Rivest-Shamir-Adleman) for asymmetric encryption and is stored in the server.

The second portion of the image uses a combination of 3DES for symmetric encryption and RSA (Rivest-Shamir-Adleman) for asymmetric encryption and is stored in the server.



**7.3 Module 3: Downloading and decryption**

The user can view all the files uploaded in the 'Documents' section of their profile. There is a key pair for each of the files. The user clicks on the button to which file needs to be viewed. At that time, the private key will be sent to the registered email id. Now, the user will be redirected to a page for decryption, where he can enter the private key received via mail. If the entered key is valid, the file will be viewed in its original format.

In the decryption phase, Retrieve the two encrypted portions of the image from the server. Decrypt the second portion using RSA for asymmetric decryption and 3DES for symmetric decryption to obtain the original second part of the image. Decrypt the first portion using RSA for asymmetric decryption and AES for symmetric decryption to obtain the original first part of the image. Combine these two decrypted portions to reconstruct the original image an

## Chapter 8

### IMPLEMENTATION

“Securing Medical Imaging: A Hybrid Cryptographic Approach with Web Deployment” uses a hybrid cryptographic algorithm to encrypt Images at the client side using the combination of AES-3DES-RSA encryption before uploading to the web server.

When encryption is done, the encrypted file is uploaded to the server. The uploaded encrypted file has to go through cloud server-side encryption once again. Then we download the file from the server to our storage and decrypt the data file by applying the private keys. When decryption is done, we get back our desired data which was secure in the server as well as during transmission.

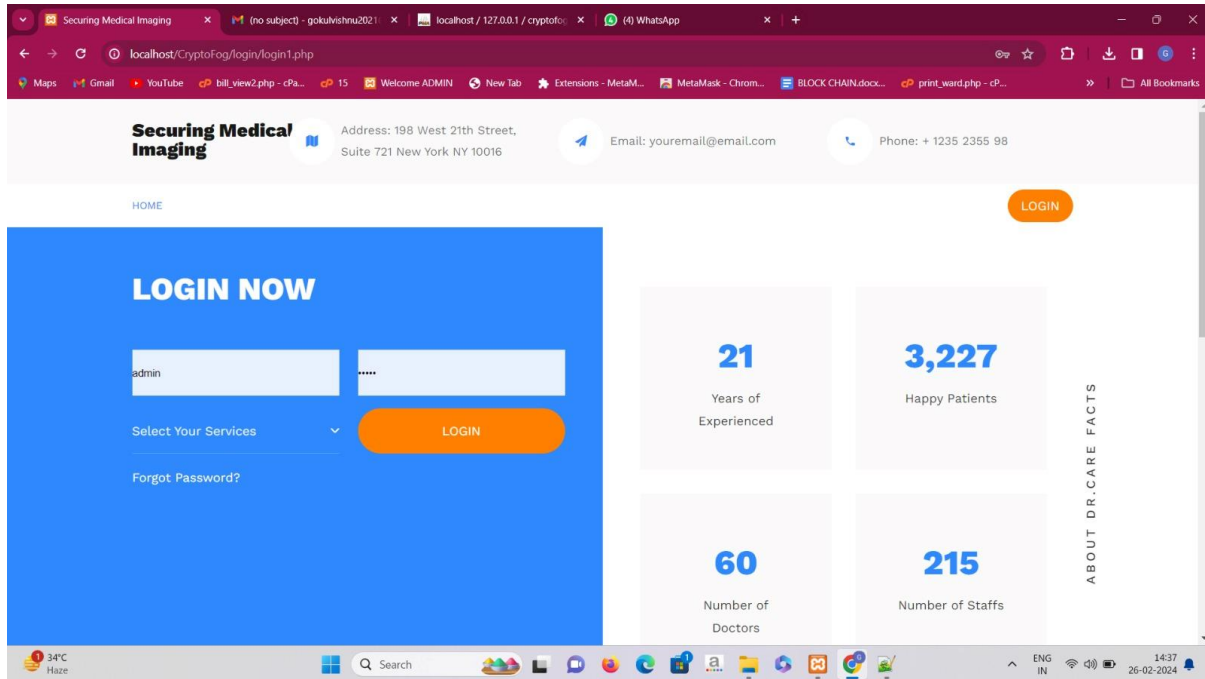
The system introduces a portal through which the admin can register other users, followed by an OTP verification, which is an authentication technique to provide extra security. So, after login in the user can see a page where he can enter his username and password. If the login is successful, there will be a dashboard with different options for different types of users. To upload a new file, which can only be done by the admin or doctors, in the form of a JPEG, he can click on choose a file button and then browse the file from his local storage. After selecting the file, he can give a name and a description for it. Then click on 'upload'. Now, the file will be uploaded to the server after the encryption.

When he clicks the 'document' option he can see a page where there is a table listed with the files that are uploaded with the patient name and a 'view file' button along with each one. To download a particular file, click the same button near each file. Then the user will get the private key generator during the decryption via his registered email ID. After providing the key the document can be viewed. Now the user cannot copy the private key received via the mail and paste in that field and click on the view button. Now the file will be downloaded to his device which is in the decrypted format.

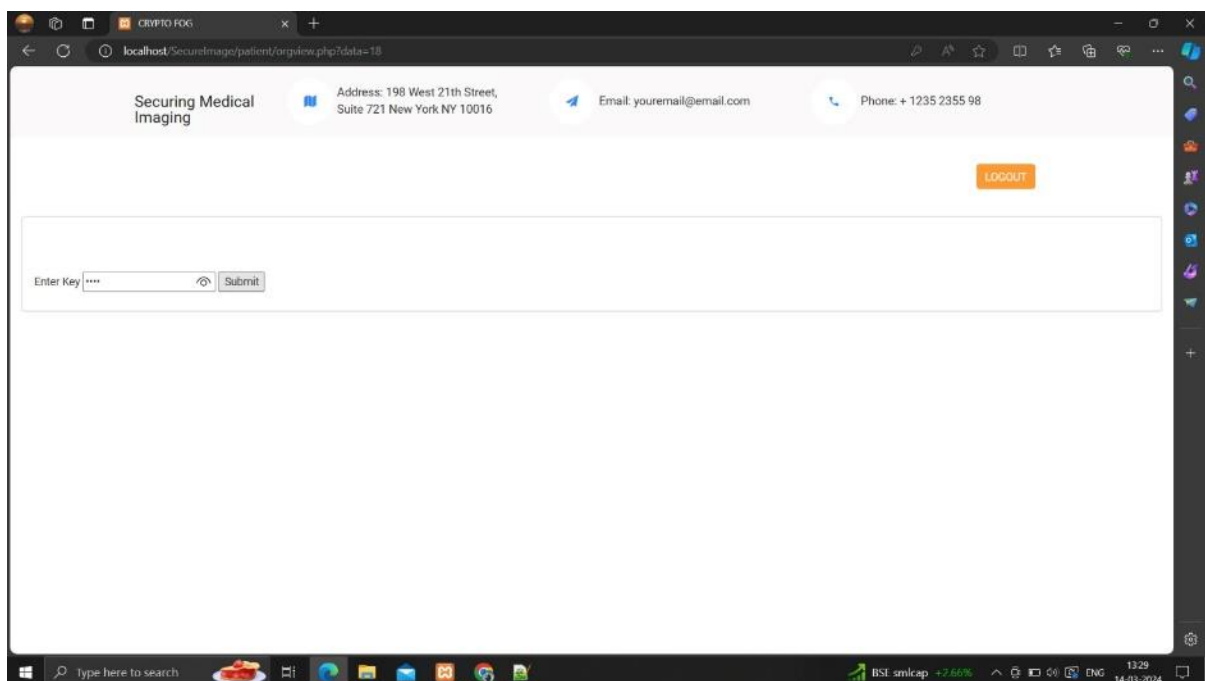
## 8.1 STEPS TO USE THE SYSTEM

1. Click on the URL: <https://clouddeployhub.co.in/Project/SecureImage/login/index.php>

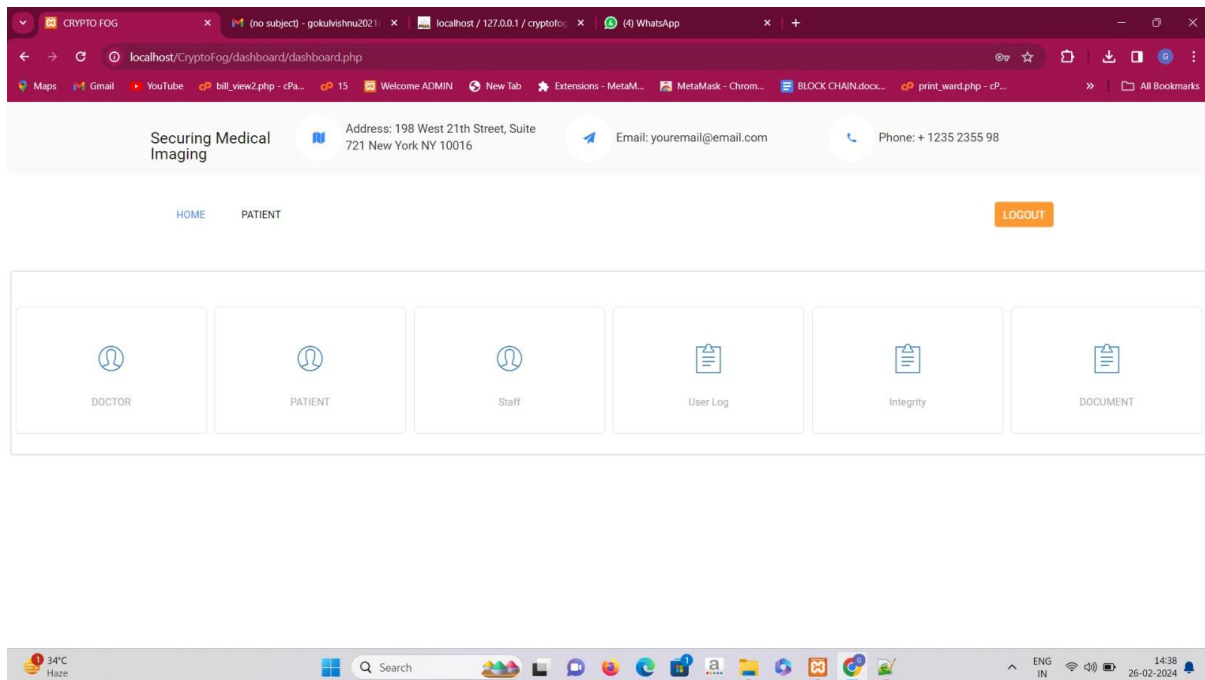
The login page displays the option to log in by entering the email ID and password.



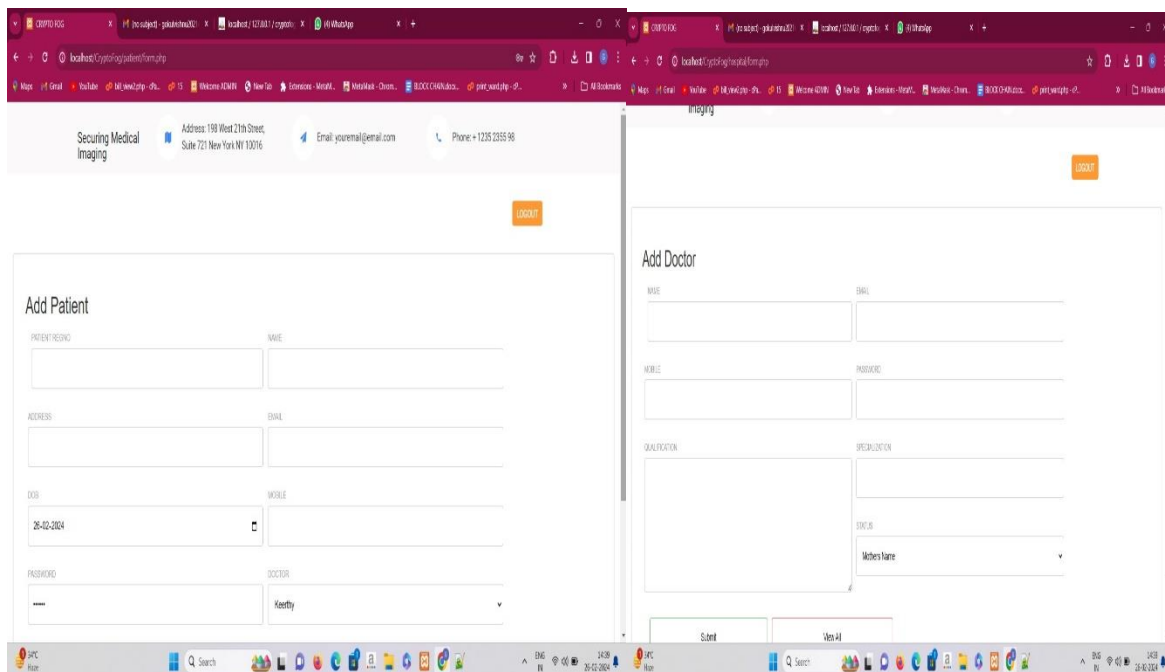
2. now the user will be directed to an email verification link, and enter the OTP received through the registered mail.



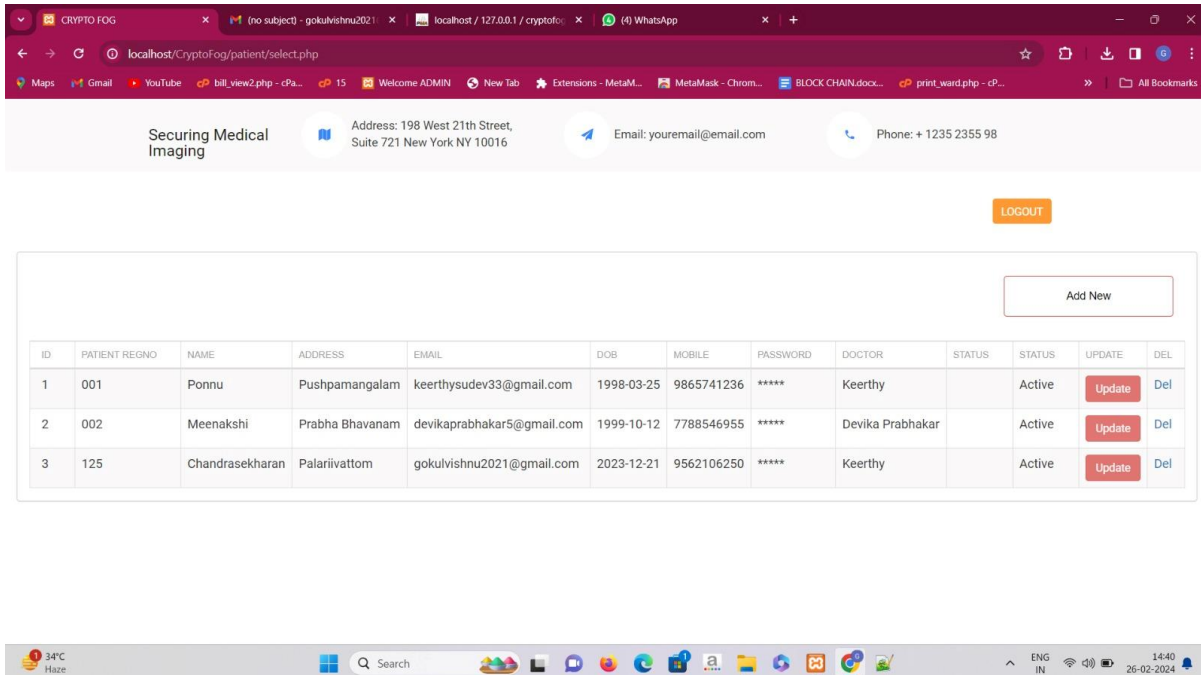
3. The admin user will have the dashboard with all the users and the documents.



4. The admin can add new doctors and patients to the server



## 5. The admin and doctor can upload files for the patients



Securing Medical Imaging

Address: 198 West 21th Street, Suite 721 New York NY 10016

Email: youremail@email.com

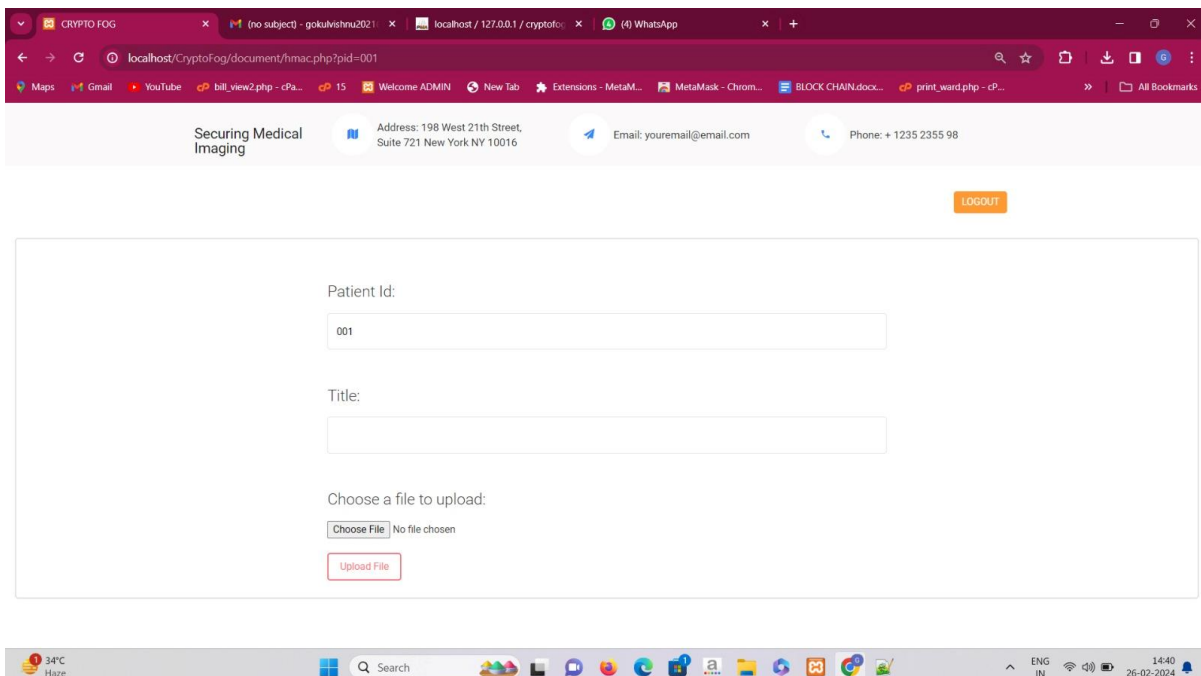
Phone: + 1235 2355 98

LOGOUT

Add New

ID	PATIENT REGNO	NAME	ADDRESS	EMAIL	DOB	MOBILE	PASSWORD	DOCTOR	STATUS	STATUS	UPDATE	DEL
1	001	Ponnu	Pushpamangalam	keerthysudev33@gmail.com	1998-03-25	9865741236	*****	Keerthy		Active	Update	Del
2	002	Meenakshi	Prabha Bhavanam	devikaprabhakar5@gmail.com	1999-10-12	7788546955	*****	Devika Prabhakar		Active	Update	Del
3	125	Chandrasekharan	Palariivattom	gokulvishnu2021@gmail.com	2023-12-21	9562106250	*****	Keerthy		Active	Update	Del

## 6. The user will require the OTP from their registered mail ID to upload the image file to the system.



Securing Medical Imaging

Address: 198 West 21th Street, Suite 721 New York NY 10016

Email: youremail@email.com

Phone: + 1235 2355 98

LOGOUT

Patient Id:

001

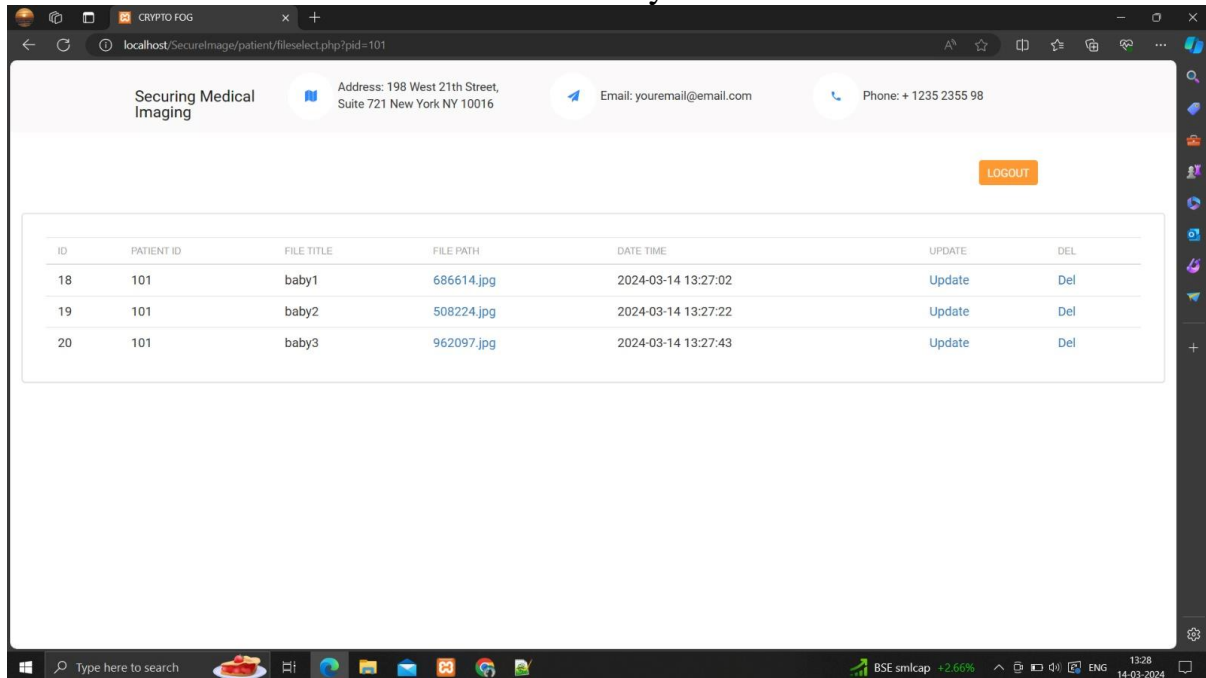
Title:

Choose a file to upload:

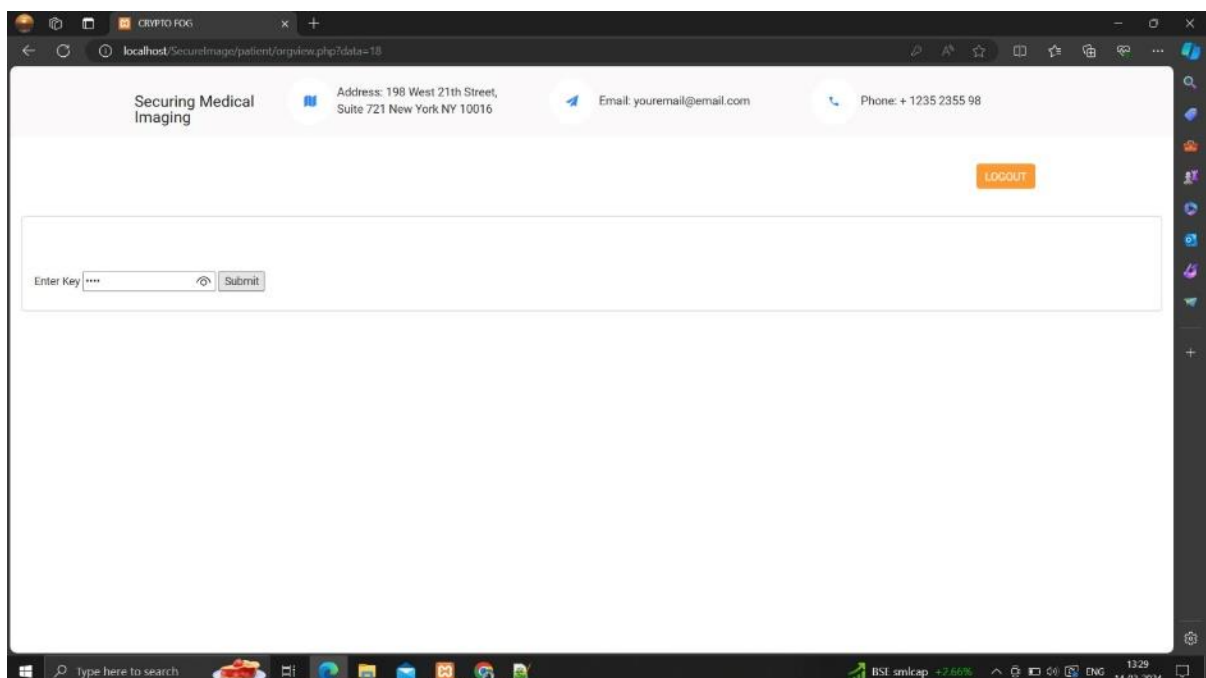
Choose File No file chosen

Upload File

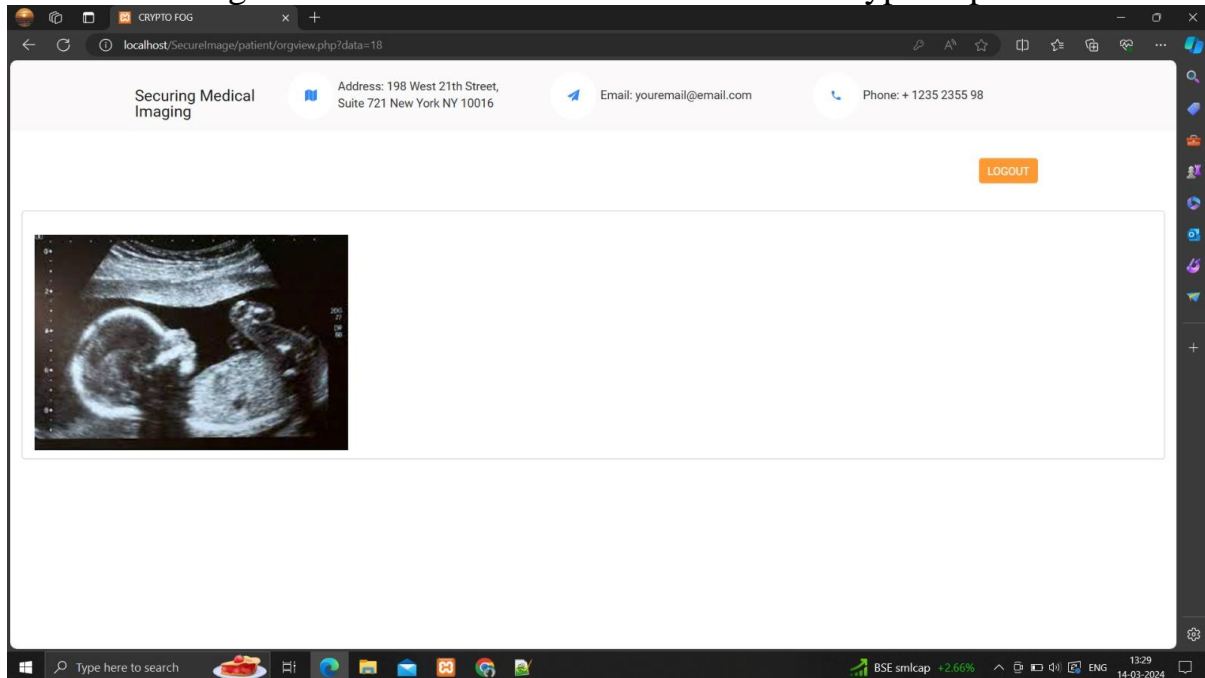
## 7. To view the file select the file name you want to view



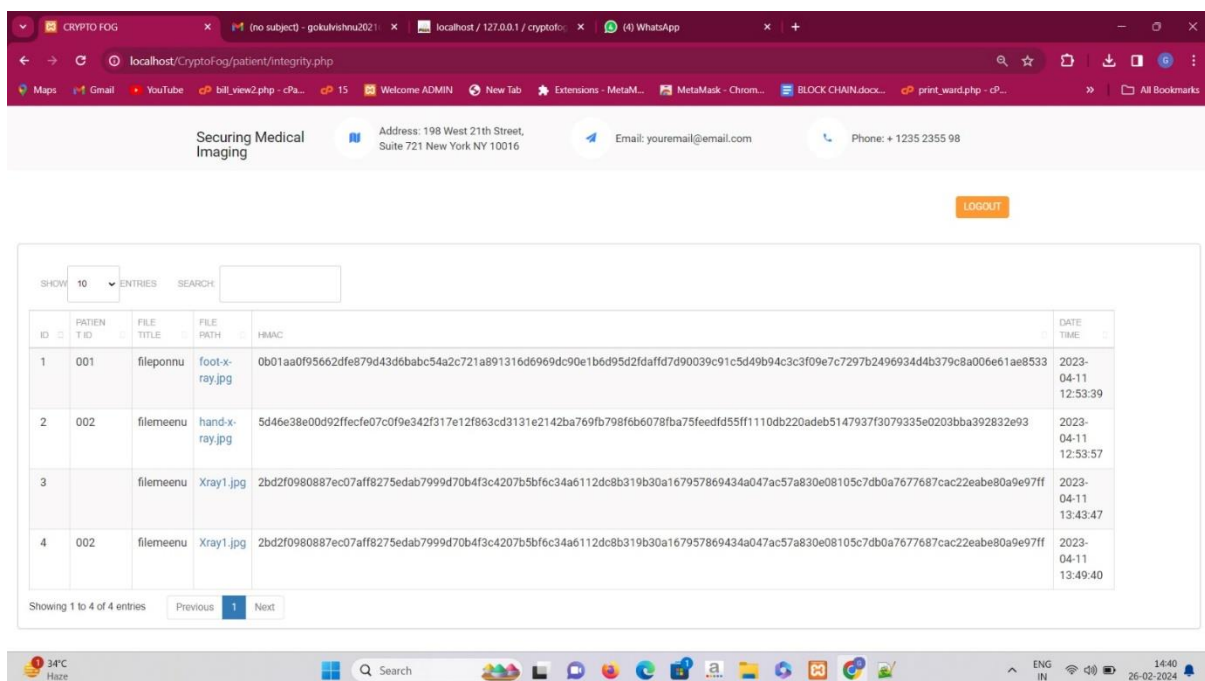
## 8. Enter the OTP received from the mail and enter the 'submit button



## 9. The Image will be viewed after the client-side decryption process



## 10. The integrity of these files can also be checked in the 'integrity' button on the dashboard



## CHAPTER 9

### CONCLUSION

The project could involve developing a client-side encryption system that utilizes asymmetric key cryptography to protect the data before it is uploaded to the web server. The project would aim to address the security concerns associated with server storage by providing a secure and user-friendly solution for encrypting data before it is uploaded to the server.

Before uploading the data to the server, the system would encrypt the data on the client side using a combination of symmetric and asymmetric algorithms. The encrypted data would then be uploaded to the web server service provider, where it would be stored securely. When the client needs to access the data, they would use the system to decrypt the data using their combination of hybrid algorithms.

For symmetric encryption, Using AES or 3DES for client-side encryption is highly secure due to its strong encryption algorithm, providing robust protection for sensitive data. 3DES while not as modern as AES, still offers a good level of security and is widely compatible with legacy systems. Both AES and 3DES are standardized encryption algorithms, ensuring interoperability across different platforms and systems. Additionally, these algorithms are computationally efficient, enabling fast encryption and decryption processes on client devices without compromising security. By implementing AES or 3DES for client-side encryption, organizations can enhance data confidentiality, integrity, and overall protection against unauthorized access or data breaches.

For asymmetric encryption, the RSA algorithm is used for the encryption since it is a widely used public-key cryptosystem that is based on the mathematical factoring of large integers. It can be used for client-side encryption in situations where the client needs to send an encrypted message to a server and wants to ensure that only the server will be able to read the message. Here, the system implemented by us successfully encrypts files to the web server and stores them securely in it. Then the encrypted document can be easily downloaded and decrypted successfully in our system.

Overall, the project would provide an added layer of security for the data stored in the cloud and increase trust in cloud storage as a solution for storing sensitive information.



## APPENDICES

### SOURCE CODE

```

<?php
include('../db/connectionI.php');
include("../header_inner.php");
include("table.php");
$p=$_POST['pid'];
$sql2 = "select * from patient where patient_regno='$p' ";
$result2 = mysqli_query($con, $sql2) or die("Error in Selecting " .
mysqli_error($connection));
$row2 =mysqli_fetch_array($result2);
$r=rand(100000,999999);
$k=0;
?><!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.min.css">
<script src="jquery.min.js"></script>
<script src="bootstrap.min.js"></script>
</head>
<body>
<?php
$keyPair = openssl_pkey_new(array(
    'private_key_bits' => 2048,
    'private_key_type' => OPENSSL_KEYTYPE_RSA,
));
$infile="../patient/public/pub$row2[id].txt";
$handle = fopen($infile, "rb") or die("Could not open a file3.");
$publicKey = fread($handle, filesize($infile));
fclose($handle);

```

```

$target_path = "uploads/";

$target_path = $target_path . "input.jpg";

if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target_path))
{
    copy($target_path,$target_path2);
    //move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target_path2);

    echo "The file ". basename( $_FILES['uploadedfile']['name']).
    " has been uploaded";
}
else{
    echo "There was an error uploading the file, please try again!";
}
$infile=$target_path;

$filename=$_FILES['uploadedfile']['name'];

$file_info = pathinfo($target_path);

// Get the file extension
$file_extension = $file_info['extension'];

?>
<html>
<head>
<title>Uploading Complete</title>
</head>
<body>
<h4>Uploaded File Info:</h4>
<ul>
<li>Sent file: <?php echo $_FILES['uploadedfile']['name']; ?>
<li>File size: <?php echo $_FILES['uploadedfile']['size']; ?> bytes
</ul> </body>
</html>

<?php

    error_reporting(0);
    // Define the file path to the original image
    $image_path = $infile;

    // Load the original image

```

```
$original_image = imagecreatefromjpeg($infile);

// Get the width and height of the image
$width = imagesx($original_image);
$height = imagesy($original_image);

// Calculate the midpoint
$midpoint = ceil($width / 2);

// Create two new images for each part
$image_part1 = imagecrop($original_image, ['x' => 0, 'y' => 0, 'width' => $midpoint, 'height'
=> $height]);
$image_part2 = imagecrop($original_image, ['x' => $midpoint, 'y' => 0, 'width' => $width -
$midpoint, 'height' => $height]);

// Save each part to a new image file
imagejpeg($image_part1, 'part1.jpg');
imagejpeg($image_part2, 'part2.jpg');

// Free up memory by destroying the images
imagedestroy($original_image);
imagedestroy($image_part1);
imagedestroy($image_part2);

// Encryption key and method (replace with your own)
$key = 'your_secret_key';
$method = 'aes-256-cbc';
$method2 = 'des-ede3-cbc';

// Read the image parts
$encrypted_part1 = openssl_encrypt(file_get_contents('part1.jpg'), $method, $key, 0, $key);
$encrypted_part2 = openssl_encrypt(file_get_contents('part2.jpg'), $method2, $key, 0, $key);

$p1="enc/" . $r . "p1.dat";
$p2="enc/" . $r . "p2.dat";
// Save encrypted parts to new files
file_put_contents($p1, $encrypted_part1);
file_put_contents($p2, $encrypted_part2);

openssl_public_encrypt($dataToEncrypt, $encryptedData, $publicKey);

$k="key/" . $r . ".dat";
// Save the encrypted data to a file (for demonstration purposes)
file_put_contents($k, $encryptedData);

?>
```

```

<?php
$myhmac=hash_hmac_file('sha512', $p1, 'secret');
?>
<html>
<head>
<h4> Its HMAC key is: </h4>
</body>
</html>
<?php
echo $myhmac;
$data=$myhmac;
$filename=$r." ".$file_extension;
$sql="insert          into          document(patient_id,file_title,file_path,hmac)
values('$ _POST[pid]','$ _POST[title]','$filename','$myhmac)";
mysqli_query($con,$sql) or die("ERROR :: ".mysqli_error($con));

?>

<?php
error_reporting(0);
session_start();

        $keyPair = openssl_pkey_new(array(
        'private_key_bits' => 2048,
        'private_key_type' => OPENSSL_KEYTYPE_RSA,
    ));

include("../header_inner.php");
include('../db/connectionI.php');
$data=$_REQUEST['data'];
//echo $data[1];

if(isset($_POST['sub2']))
{
if($_POST['rand1']==$_POST['rand2'])
{
        mysqli_query($con,"INSERT INTO user_logo( user_id,status,user,remark) VALUES
('$ _SESSION[userid]','Success','$ _SESSION[user]','decoy)");
        $sql22 = "select * from document where id='$data' ";
        $result22 = mysqli_query($con, $sql22) or die("Error in Selecting " .
mysqli_error($connection));
        $row22 =mysqli_fetch_array($result22);
        $p=$row22['patient_id'];
        $sql2 = "select * from patient where patient_regno='$p' ";
        $result2 = mysqli_query($con, $sql2) or die("Error in Selecting " .
mysqli_error($connection));
        $row2 =mysqli_fetch_array($result2);

```

```

        $r=explode(".", $row22['file_path']);
        $infile2="../document/key/".$r[0].".dat";
        $p1="../document/enc/".$r[0].".p1.dat";
        $p2="../document/enc/".$r[0].".p2.dat";
        $p3="../document/dec/x.jpg";
        $encryptedData = file_get_contents($infile2);
        $infile="../patient/private/prv$row2[id].txt";
        $privateKey = file_get_contents($infile);
        openssl_private_decrypt($encryptedData, $decryptedData, $privateKey);
        $key = $decryptedData;
        $key = 'your_secret_key';
        //echo "ghghhgg ydeeeeeeeeeecguch".$decryptedData;
        $method = 'aes-256-cbc';
$method2 = 'des-ede3-cbc';
// Read the encrypted image parts
$encrypted_part1 = file_get_contents($p1);
$encrypted_part2 = file_get_contents($p2);

// Decrypt the image parts
$decrypted_part1 = openssl_decrypt($encrypted_part1, $method, $key, 0, $key);
$decrypted_part2 = openssl_decrypt($encrypted_part2, $method2, $key, 0, $key);

// Create images from decrypted parts
$image_part1 = imagecreatefromstring($decrypted_part1);
$image_part2 = imagecreatefromstring($decrypted_part2);

// Get the dimensions of the image parts
$width_part1 = imagesx($image_part1);
$width_part2 = imagesx($image_part2);
$height = imagesy($image_part1);

// Create a new image with the combined width
$combined_image = imagecreatetruecolor($width_part1 + $width_part2, $height);

// Copy the image parts into the new image
imagecopy($combined_image, $image_part1, 0, 0, 0, 0, $width_part1, $height);
imagecopy($combined_image, $image_part2, $width_part1, 0, 0, 0, $width_part2, $height);

// Save the recombined image
imagejpeg($combined_image, $p3);

// Free up memory by destroying the images
imagedestroy($image_part1);
imagedestroy($image_part2);
imagedestroy($combined_image);

        echo "<img src='$p3' width='400'>";
    }

}

```

```

else
{
$rand=rand(1000,9999);

$url="https://alc-training.in/gateway.php";

$ch = curl_init();
if (!$ch){die("Couldn't initialize a cURL
handle");} $ret = curl_setopt($ch,CURLOPT_URL,$url);
curl_setopt ($ch,CURLOPT_POST, 1);
curl_setopt($ch,
CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch,CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt ($ch,CURLOPT_POSTFIELDS,"email=$_SESSION[email]&msg=Your key for
access is $rand ..");
$ret = curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

curl_exec($ch); //
curl_close($ch);
?>

<br /><br />

<form action="" method="post">
Enter Key

<input type="password" name="rand1" onpaste='return false;' />
<input type="hidden" name="rand2" value="<?php echo $rand; ?>" />
<input type="submit" name="sub2" />
</form>

<?php
}
?>

```

## REFERENCE

- [1] Diao Salama Abdelminaam, Improving the Security of Cloud Computing by Building New Hybrid Cryptography Algorithms, Information Systems department, Faculty of Computers and Informatics, Benha University, Egypt, 2018
- [2] Md. Abu Musa and Md. Ashiq Mahmood, "Client-side Cryptography Based Security for Cloud Computing System", Institute of Information and Communication Technology Khulna University of Engineering & Technology (KUET), 2021
- [3] Václav Kaczmarczyk, Zdeněk Bradáč, Petr Fiedler, and Jakub Arm, "Client-side data encryption/decryption for a web application", 2016
- [4] Emily Stark, "From Client-side Encryption to Secure Web Applications", Submitted to the Department of Electrical Engineering and Computer Science, 2016
- [5] Secure client-side encryption scheme using hybrid cryptography for web servers" by A. Smith et al. (2021) - This paper proposes a novel hybrid cryptography approach for client-side encryption in web servers to enhance data security and access control.
- [6] "Efficient key management for client-side encryption in web server environments" by B. Johnson et al. (2020) - This work focuses on addressing key management challenges in client-side encryption using hybrid cryptography techniques to reduce computational overhead.
- [7] "Enhancing data confidentiality in web server environments using hybrid cryptography" by C. Williams et al. (2019) - The authors present a hybrid cryptography-based solution for improving data confidentiality in web servers through client-side encryption.
- [8] "Secure data sharing in web servers using hybrid encryption techniques" by D. Brown et al. (2018) - This paper discusses the use of hybrid encryption techniques to facilitate secure data sharing in web server environments, ensuring data privacy and integrity.
- [9] "Hybrid encryption approach for client-side security in web servers" by E. Davis et al. (2017) - The authors propose a hybrid encryption approach to enhance client-side security in web servers, focusing on mitigating key management challenges and ensuring secure data access.
- [10] "Performance evaluation of hybrid cryptography for client-side encryption in web servers" by F. Wilson et al. (2016) - This study evaluates the performance of hybrid cryptography techniques for client-side encryption in web servers, considering factors such as computational overhead and data security.