CEDOCS: Secure Data

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

PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree of

BCA
(CLOUD TECHNOLOGY AND INFORMATION SECURITY MANAGEMENT)

By
Danaa Salam- SB20BCA015
&
Sanaa Salam- SB20BCA036

III DC BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY MANAGEMENT)

Under the guidance of
Veena Antony

DEPARTMENT OF COMPUTER APPLICATIONS
MARCH 2023

CEDOCS: Secure Data

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

PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree of

BCA
(CLOUD TECHNOLOGY AND INFORMATION SECURITY MANAGEMENT)

By
Danaa Salam- SB20BCA015
&
Sanaa Salam- SB20BCA036

III DC BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY MANAGEMENT)

Under the guidance of
Veena Antony

DEPARTMENT OF COMPUTER APPLICATIONS
MARCH 2023

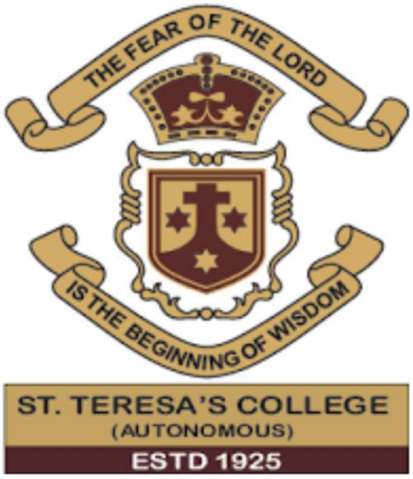
DECLARATION

We, undersigned, hereby declare that the project report, "**CEDOCS: Secure Data**", submitted for partial fulfilment 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 Bonafede work done by us under the supervision of Miss.Veena Antony. This submission represents our ideas in our own words and where ideas or words of others have not been included. We have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or 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 2023

Danaa Salam–SB20BCA015
Sanaa Salam– SB20BCA036

ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM
BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY MANAGEMENT)
DEPARTMENT OF COMPUTER APPLICATIONS



CERTIFICATE

This is to certify that the report entitled "**CEDOCS: Securing Data**", submitted by **Danaa Salam** and **Sanaa Salam** to the Mahatma Gandhi University in partial fulfilment of the requirements for the award of the Degree of BCA (Cloud Technology and Information Security Management) is a Bonafede 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.

Ms.Veena Antony	Ms.Raji S Pillai	
Internal Supervisor	**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 Ms. Raji S Pillai and the Course coordinator **Ms. Sheeba Emmanuel** for the support. We deeply express sincere thanks to our project guide **Ms.Veena Anotony** for her proper guidance and support throughout the project work.

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

We convey our hearty thanks to our parents for the moral support, suggestion and encouragement.

ABSTRACT

Cryptography refers to securing information and communication derived from mathematical concepts and algorithms, to transform messages in ways that are hard to decipher. Cloud Computing is the provisioning of sharing information and resources that are delivered as a service to the end users on demand. In these days a lot of business organizations are using cloud for storing their data. As Cloud enables your business to communicate and share more easily. It also allows better collaboration between employees, enabling multiple users to share and work on data and files at the same time.

Client-side encryption is the cryptographic technique of encrypting data on the sender's side before it is transmitted to a server such as a cloud storage service. Client-side encryption of data before uploading to the cloud is important because it ensures that your data is secure while it is being transferred to the cloud and while it is stored in the cloud. When you encrypt your data before uploading it to the cloud, it is transformed into a scrambled form that can only be decrypted with the correct key. This means that even if your data is accessed by someone unauthorized, they will not be able to read or use it. So, Even if the cloud provider has strong security measures in place, there is still a risk that your data could be accessed by someone unauthorized.

" CEDOCS: Secure Data" proposes a system that uses asymmetric key encryption that would encrypt a file locally at the client-side prior to uploading to the cloud and the file would decrypt after downloading it on the client-side.

So here, in this proposed work, first the encryption is done using RSA algorithm and the encrypted file is uploaded to the Cloud. The uploaded file has to go through cloud server-side encryption once again. Then we download the file from the cloud to the 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 cloud as well as during transmission. By encrypting your data before uploading it to the cloud, you ensure that it is secure and protected at all times.

.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 GENERAL BACKGROUND

In recent decades, Data Security has remained a significant domain. As data is being traversed, there are chances of misuse. Data is a valuable asset that generates, acquires, saves, and exchanges for any company. Protecting it from internal or external corruption and illegal access protects a company from financial loss, reputational harm, consumer trust degradation, and brand erosion. Furthermore, regulations for securing data, imposed by the government and the industry, make it critical for a company to achieve and maintain compliance wherever it does business.

Data Security is now a must-have, and the importance of data security is increasing day by day, hence the financial investments that your company is willing to make should reflect that.

1.2 Information Security

Information security protects sensitive information from unauthorized activities, including inspection, modification, recording, and any disruption or destruction. The goal is to ensure the safety and privacy of critical data such as customer account details, financial data or intellectual property.

The consequences of security incidents include theft of private information, data tampering, and data deletion. Attacks can disrupt work processes and damage a company's reputation, and also have a tangible cost.

1.3 Principles of Information Security

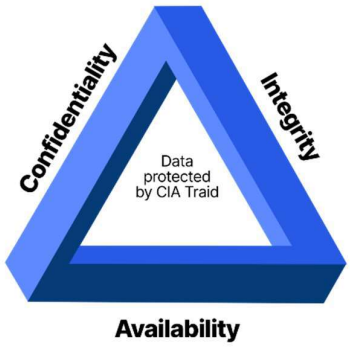


Fig:1.1 CIA Triad

The basic tenets of information security are confidentiality, integrity and availability. Every element of the information security program must be designed to implement one or more of these principles. Together they are called the CIA Triad.

1.3.1 Confidentiality

Confidentiality is roughly equivalent to privacy. Confidentiality measures are designed to prevent sensitive information from unauthorized access attempts. It is common for data to be categorized according to the amount and type of damage that could be done if it fell into the wrong hands. More or less stringent measures can then be implemented according to those categories.

Sometimes safeguarding data confidentiality involves special training for those privy to sensitive documents. Training can help familiarize authorized people with risk factors and how to guard against them. Further aspects of training may include strong passwords and password-related best practices and information about social engineering methods to prevent users from bending data-handling rules with good intentions and potentially disastrous results.

A good example of methods used to ensure confidentiality is requiring an account number or routing number when banking online. Data encryption is another common method of ensuring confidentiality. User IDs and passwords constitute a standard procedure; two-factor authentication (2FA) is becoming the norm. Other options include Biometric verification and security tokens, key fobs or soft tokens. In addition, users can take precautions to minimize the number of places where information appears and the number of times it is actually transmitted to complete a required transaction. Extra measures might be taken in the case of extremely sensitive documents, such as storing only on air-gapped computers, disconnected storage devices or, for highly sensitive information, in hard-copy form only.

1.3.2 Integrity

Integrity involves maintaining the consistency, accuracy and trustworthiness of data over its entire lifecycle. Data must not be changed in transit, and steps must be taken to ensure data cannot be altered by unauthorized people (for example, in a breach of confidentiality).

These measures include file permissions and user access controls. Version control may be used to prevent erroneous changes or accidental deletion by authorized users from becoming a problem. In addition, organizations must put in some means to detect any changes in data that

might occur as a result of non-human-caused events such as an electromagnetic pulse (EMP) or server crash.

Data might include checksums, even cryptographic checksums, for verification of integrity. Backups or redundancies must be available to restore the affected data to its correct state. Furthermore, digital signatures can be used to provide effective nonrepudiation measures, meaning evidence of logins, messages sent, electronic document viewing and sending cannot be denied.

1.3.3 Availability

Availability means information should be consistently and readily accessible for authorized parties. This involves properly maintaining hardware and technical infrastructure and systems that hold and display the information.

This is best ensured by rigorously maintaining all hardware, performing hardware repairs immediately when needed and maintaining a properly functioning operating system (OS) environment that is free of software conflicts. It's also important to keep current with all necessary system upgrades. Providing adequate communication bandwidth and preventing the occurrence of bottlenecks are equally important tactics. Redundancy, failover, RAID -- even high-availability clusters -- can mitigate serious consequences when hardware issues do occur.

Fast and adaptive disaster recovery is essential for the worst-case scenarios; that capacity relies on the existence of a comprehensive DR plan. Safeguards against data loss or interruptions in connections must include unpredictable events such as natural disasters and fire. To prevent data loss from such occurrences, a backup copy may be stored in a geographically isolated location, perhaps even in a fireproof, waterproof safe. Extra security equipment or software such as firewalls and proxy servers can guard against downtime and unreachable data blocked by malicious denial-of-service (DoS) attacks and network intrusions.

1.4 CLOUD COMPUTING

Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user. Large clouds often have functions distributed over multiple locations, each of which is a data center. Cloud computing relies on sharing of resources to achieve coherence and typically uses a "pay as you go" model, which can help in reducing capital expenses but may also lead to unexpected operating expenses for users.

"The cloud" refers to servers that are accessed over the Internet, and the software and databases that run on those servers. Cloud servers are located in data centers all over the world. By using cloud computing, users and companies do not have to manage physical servers themselves or run software applications on their own machines.

The cloud enables users to access the same files and applications from almost any device, because the computing and storage takes place on servers in a data center, instead of locally on the user device. This is why a user can log in to their Instagram account on a new phone after their old phone breaks and still find their old account in place, with all their photos, videos, and conversation history. It works the same way with cloud email providers like Gmail or Microsoft Office 365, and with cloud storage providers like Dropbox or Google Drive.

For businesses, switching to cloud computing removes some IT costs and overhead: for instance, they no longer need to update and maintain their own servers, as the cloud vendor they are using will do that. This especially makes an impact for small businesses that may not have been able to afford their own internal infrastructure but can outsource their infrastructure needs affordably via the cloud. The cloud can also make it easier for companies to operate internationally, because employees and customers can access the same files and applications from any location.

Cloud computing is possible because of a technology called virtualization. Virtualization allows for the creation of a simulated, digital-only "virtual" computer that behaves as if it were a physical computer with its own hardware. The technical term for such a computer is virtual machine. When properly implemented, virtual machines on the same host machine are sandboxed from one another, so they do not interact with each other at all, and the files and applications from one virtual machine are not visible to the other virtual machines even though they are on the same physical machine.

1.4.1 CLOUD DEPLOYEMENT MODELS

In contrast to the models discussed above, which define how services are offered via the cloud, these different cloud deployment types have to do with where the cloud servers are and who manages them.

The most common cloud deployments are:

Private cloud: A private cloud is a server, data centre, or distributed network wholly dedicated to one organization.

Public cloud: A public cloud is a service run by an external vendor that may include servers in one or multiple data centres. Unlike a private cloud, public clouds are shared by multiple organizations. Using virtual machines, individual servers may be shared by different companies, a situation that is called "multitenancy" because multiple tenants are renting server space within the same server.

Hybrid cloud: hybrid cloud deployments combine public and private clouds, and may even include on-premises legacy servers. An organization may use their private cloud for some services and their public cloud for others, or they may use the public cloud as backup for their private cloud.

Multi-cloud: multi-cloud is a type of cloud deployment that involves using multiple public clouds. In other words, an organization with a multi-cloud deployment rents virtual servers and services from several external vendors — to continue the analogy used above, this is like leasing several adjacent plots of land from different landlords. Multi-cloud deployments can also be hybrid cloud, and vice versa

1.4.2 CLOUD SERVICE MODELS

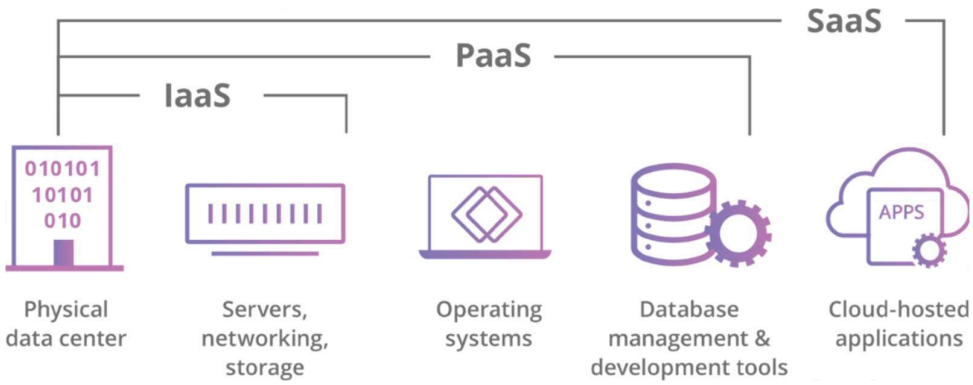


Fig 1.2 Cloud Service Models

Software-as-a-Service (SaaS): Instead of users installing an application on their device, SaaS applications are hosted on cloud servers, and users access them over the Internet. SaaS is like renting a house: the landlord maintains the house, but the tenant mostly gets to use it as if they owned it. Examples of SaaS applications include Salesforce, MailChimp, and Slack.

Platform-as-a-Service (PaaS): In this model, companies don't pay for hosted applications; instead, they pay for the things they need to build their own applications. PaaS vendors offer everything necessary for building an application, including development tools, infrastructure, and operating systems, over the Internet. PaaS can be compared to renting all the tools and equipment necessary for building a house, instead of renting the house itself. PaaS examples include Heroku and Microsoft Azure.

Infrastructure-as-a-Service (IaaS): In this model, a company rents the servers and storage they need from a cloud provider. They then use that cloud infrastructure to build their applications. IaaS is like a company leasing a plot of land on which they can build whatever they want — but they need to provide their own building equipment and materials. IaaS providers include DigitalOcean, Google Compute Engine, and OpenStack.

Formerly, SaaS, PaaS, and IaaS were the three main models of cloud computing, and essentially all cloud services fit into one of these categories. However, in recent years a fourth model has emerged:

Function-as-a-Service (FaaS): FaaS, also known as serverless computing, breaks cloud applications down into even smaller components that only run when they are needed. Imagine if it were possible to rent a house one little bit at a time: for instance, the tenant only pays for the dining room at dinner time, the bedroom while they are sleeping, the living room while they are watching TV, and when they are not using those rooms, they don't have to pay rent on them.

FaaS or serverless applications still run on servers, as do all these models of cloud computing. But they are called "serverless" because they do not run on dedicated machines, and because the companies building the applications do not have to manage any servers.

1.5 CRYPTOGRAPHY

Cryptography is technique of securing information and communications through use of codes so that only those person for whom the information is intended can understand it and process it. Thus, preventing unauthorized access to information. The prefix “crypt” means “hidden” and suffix graphy means “writing”. In Cryptography the techniques which are use to protect information are obtained from mathematical concepts and a set of rule-based calculations known as algorithms to convert messages in ways that make it hard to decode it. These algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on internet and to protect confidential transactions such as credit card and debit card transactions.

Techniques used For Cryptography: In today’s age of computers cryptography is often associated with the process where an ordinary plain text is converted to cipher text which is the text made such that intended receiver of the text can only decode it and hence this process is known as encryption. The process of conversion of cipher text to plain text this is known as decryption.

Features Of Cryptography are:

Confidentiality: Information can only be accessed by the person for whom it is intended and no other person except him can access it.

Integrity: Information cannot be modified in storage or transition between sender and intended receiver without any addition to information being detected.

Non-repudiation: The creator/sender of information cannot deny his intention to send information at later stage.

Authentication: The identities of sender and receiver are confirmed. As well as destination/origin of information is confirmed.

1.5.1 TYPES OF CRYPTOGRAPHY

In general, there are three types of cryptography:

Symmetric Key Cryptography: It is an encryption system where the sender and receiver of message use a single common key to encrypt and decrypt messages. Symmetric Key Systems are faster and simpler but the problem is that sender and receiver have to somehow exchange key in a secure manner. The most popular symmetric key cryptography system is Data Encryption System(DES).

Hash Functions: There is no usage of any key in this algorithm. A hash value with fixed length is calculated as per the plain text which makes it impossible for contents of plain text to be recovered. Many operating systems use hash functions to encrypt passwords.

Asymmetric Key Cryptography: Under this system a pair of keys is used to encrypt and decrypt information. A public key is used for encryption and a private key is used for decryption. Public key and Private Key are different. Even if the public key is known by everyone the intended receiver can only decode it because he alone knows the private key.

1.5.2 CLIENT-SIDE CRYPTOGRAPHY

Client-side cryptography refers to the practice of encrypting data on the client's device before it is sent to the server. This means that the data is protected while in transit and while it is stored on the server, and can only be decrypted by the client who holds the encryption key. This provides an additional layer of security, as the server does not have access to the unencrypted data and is therefore less vulnerable to hacking or data breaches.

In client-side cryptography, the client's device, such as a computer or mobile device, is responsible for encrypting the data before it is sent to the server. This is typically done using a symmetric key encryption algorithm, such as AES, where the same key is used for both

encryption and decryption. The client then sends the encrypted data to the server, where it is stored in an encrypted form.

When the client wants to retrieve the data, they must provide the correct decryption key, which is typically a password or a private key. The server then sends the encrypted data back to the client, who uses the decryption key to decrypt and view the data.

One of the main advantages of client-side cryptography is that it provides an additional layer of security for the data, as the server does not have access to the decryption key and therefore cannot decrypt the data. This means that even if the server is hacked or data is stolen, the data will still be protected and unreadable to the attacker.

Another advantage is that the data remains private and only accessible by the client. This provides more control over the data and its access.

However, client-side cryptography also has some disadvantages, such as the need for the client to manage the encryption key, and the possibility of the key being lost or stolen, which would make the data unreadable. Additionally, the client-side encryption could be vulnerable to malware or other malicious software on the client's device.

There are several techniques that can be used for client-side encryption of data before uploading it to the cloud:

Symmetric key encryption: This is a method where the same key is used for both encryption and decryption. A common symmetric key encryption algorithm is AES.

Asymmetric key encryption: This method uses a pair of keys, one for encryption and one for decryption. A common asymmetric key encryption algorithm is RSA.

File-based encryption: This technique involves encrypting individual files or folders on the client's device before uploading them to the cloud. This can be done using built-in encryption tools or third-party software

Volume-based encryption: This technique involves encrypting the entire storage device on the client's device, such as a hard drive or SSD, before uploading it to the cloud.

File/folder level access control: This technique uses a set of rules to grant or deny access to specific files or folders. This is useful when sharing data with other users.

Token-based authentication: This technique uses tokens to authenticate the client's device and grant access to the data in the cloud.

Secure Hash Algorithm (SHA) : This technique uses a one-way function to hash the data, which means that the data can be verified for integrity but cannot be decrypted.

Digital signature: This technique uses a private key to sign the data and a public key to verify the signature. This is a way to ensure that the data has not been tampered with.

Need to encrypt data before uploading it to a cloud for several reasons:

Security: Cloud-based storage is vulnerable to hacking and data breaches. By encrypting the data on the client's device before uploading it to the cloud, the data is protected while in transit and while it is stored on the server. This provides an additional layer of security, as the server does not have access to the unencrypted data and is therefore less vulnerable to hacking or data breaches.

Privacy: Cloud-based storage is typically shared among multiple users, and data stored in the cloud is accessible by the cloud provider. By encrypting the data on the client's device, the client has more control over the data and its access. The data remains private and only accessible by the client, which is important for sensitive information such as financial data or personal information.

Compliance: Many industries have regulations and compliance standards that require data to be encrypted before it is transmitted or stored. Encrypting the data on the client's device before uploading it to the cloud can help organizations meet these requirements.

Data integrity: Cloud-based storage can be vulnerable to data corruption, and data stored in the cloud can be tampered with or modified. By encrypting the data on the client's device, it can be verified that the data has not been tampered with or modified before upload.

Cost-effective: By encrypting data on client-side, the cloud provider doesn't have to spend extra resources on encrypting the data on their side, which can be cost-effective for both the client and the provider.

It's It's important to note that client-side encryption alone is not a sufficient security measure, and must be used in conjunction with other security measures such as secure network protocols, firewalls, and access controls to provide a comprehensive security for data in cloud.

1.5.3 RSA ENCRYPTION

The RSA algorithm has been a reliable source of security since the early days of computing, and it keeps solidifying itself as a definitive weapon in the line of cybersecurity.

Before moving forward with the algorithm, let’s get a refresher on asymmetric encryption since it verifies digital signatures according to asymmetric cryptography architecture, also known as public-key cryptography architecture.

What Is Asymmetric Encryption?

In Asymmetric Encryption algorithms, you use two different keys, one for encryption and the other for decryption. The key used for encryption is the public key, and the key used for decryption is the private key. But, of course, both the keys must belong to the receiver.

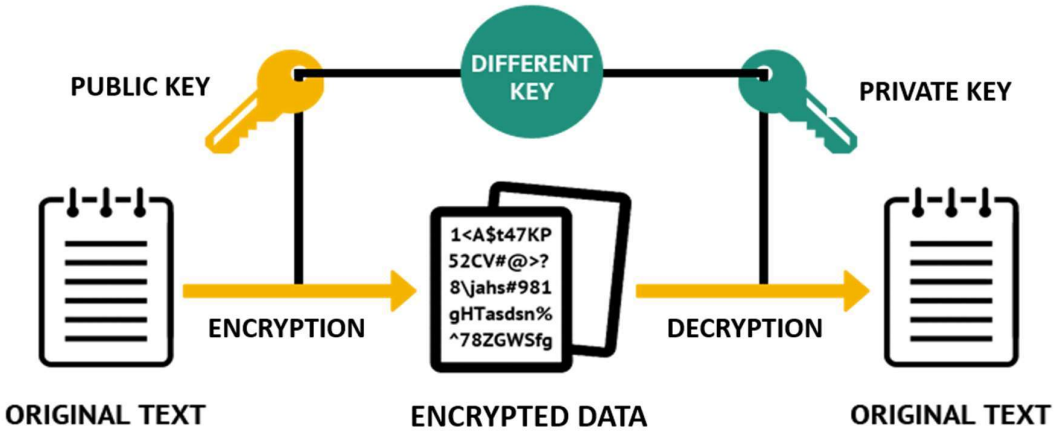


Fig 1.3 Asymmetric Encryption

As seen in the image above, using different keys for encryption and decryption has helped avoid key exchange, as seen in symmetric encryption.

For example, if Alice needs to send a message to Bob, both the keys, private and public, must belong to Bob.

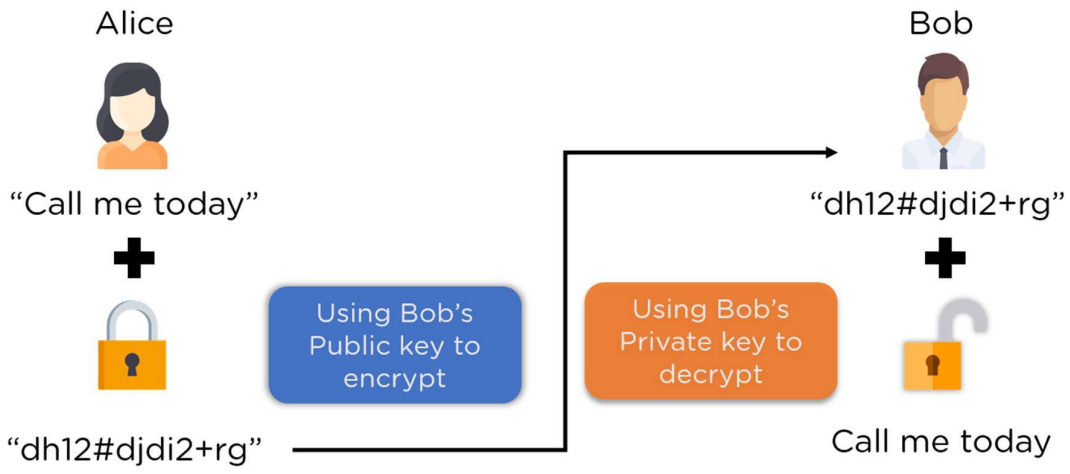


Fig 1.4 public key cryptography

The process for the above image is as follows:

Step 1: Alice uses Bob's public key to encrypt the message

Step 2: The encrypted message is sent to Bob

Step 3: Bob uses his private key to decrypt the message

This eliminates the need to exchange any secret key between sender and receiver, thereby reducing the window of exploitation.

What Is the RSA Algorithm?

The RSA algorithm is a public-key signature algorithm developed by Ron Rivest, Adi Shamir, and Leonard Adleman. Their paper was first published in 1977, and the algorithm uses logarithmic functions to keep the working complex enough to withstand brute force and streamlined enough to be fast post-deployment. The image below shows it verifies the digital signatures using RSA methodology.

RSA can also encrypt and decrypt general information to securely exchange data along with handling digital signature verification. The image above shows the entire procedure of the RSA algorithm. You will understand more about it in the next section.

RSA in Data Encryption

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 in decrypting 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-1mod(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 mod n.

If the ciphertext is c, plaintext = cd mod n

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

N = p * q = 221

D = e-1mod(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 mod n = 82.

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

Advantages of RSA

No Key Sharing: RSA encryption depends on using the receiver's public key, so you don't have to share any secret key to receive messages from others.

Proof of Authenticity: Since the key pairs are related to each other, a receiver can't intercept the message since they won't have the correct private key to decrypt the information.

Faster Encryption: The encryption process is faster than that of the DSA algorithm.

Data Can't Be Modified: Data will be tamper-proof in transit since meddling with the data will alter the usage of the keys. And the private key won't be able to decrypt the information, hence alerting the receiver of manipulation.

This sums up this lesson on the RSA Algorithm.

1.6 PROBLEM DEFINITION

Uploading confidential data to the cloud without encrypting it at the client side can pose a security risk because the data is vulnerable to interception and unauthorized access while in transit over the internet. Additionally, if the cloud storage provider's security measures are compromised, the data could be accessed by unauthorized individuals. Encrypting the data at

the client side before uploading it to the cloud helps to protect the data by making it unreadable to anyone without the proper decryption key.

Encrypting data at the client side before uploading it to the cloud is a best practice for protecting sensitive information because it ensures that the data is protected both in transit and at rest. Data in transit is vulnerable to interception by hackers or other malicious actors who may be able to access it as it travels over the internet. Once the data reaches the cloud storage provider, it is stored on servers that may also be vulnerable to attack.

Without client-side encryption, data stored in the cloud can be accessed by anyone with access to the cloud provider's servers, including unauthorized individuals who may have gained access through a security breach. In addition, cloud storage providers may be required by law enforcement or other government agencies to turn over data stored on their servers, which could include sensitive information.

Client-side encryption is a process by which data is encrypted on the user's device before it is uploaded to the cloud. This ensures that the data is unreadable to anyone who intercepts it in transit, and also makes it more difficult for unauthorized individuals to access the data if the cloud storage provider's security measures are compromised. The data can only be decrypted by someone with the proper decryption key, which is typically held by the user.

In summary, client-side encryption is important for protecting confidential data that is uploaded to the cloud because it ensures that the data is protected both in transit and at rest, and makes it more difficult for unauthorized individuals to access it.

1.7 OBJECTIVES

The main objective of this project is to safely store and secure files of an individual in a cloud storage. The present situation bought out the need for implementing encryption of data before uploading it to the cloud storage. Thys the concept of providing a security based on client-side cryptography using asymmetric encryption is introduced.

The most popular asymmetric cryptographic algorithm,RSA is used here.This is implemented for secure upload and download of our confidential data on cloud, for users who have registered in the system. The system also authenticates the identity of the person. This is performed before the user get access to files stored on the cloud

CHAPTER 2

LITERATURE SURVEY

Cryptography indicates 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.

However, in the modern era, cryptography is repeatedly related to cloud computing. But, moving data into a cloud is a huge modification and has real involvement that makes users lapse before one can sign up for the desired service which can cause unwanted instruction on sensitive information and data lost. For the security of cloud data, a symmetric algorithm had been introduced by previous research work which used simple algorithms and had performance issues. In their research paper they have used asymmetric key encryption that would encrypt a file locally at the client-side prior to uploading to the cloud.[1].

Data security and access control is one of the most challenging ongoing research work in cloud computing, because of users outsourcing their sensitive data to cloud providers. Existing solutions that use pure cryptographic techniques to mitigate these security and access control problems suffer from heavy computational overhead on the data owner as well as the cloud service provider for key distribution and management. The paper addresses this challenging open problem using capability based access control technique that ensures only valid users will access the outsourced data. This work also proposes a modified Diffie-Hellman key exchange protocol between cloud service provider and the user for secretly sharing a symmetric key for secure data access that alleviates the problem of key distribution and management at cloud service provider. The simulation run and analysis shows that the proposed approach is highly efficient and secure under existing security models.[2]

Another paper proposes a simple, secure, and privacy-preserving architecture for inter-Cloud data sharing based on an encryption/decryption algorithm which aims to protect the data stored in the cloud from the unauthorized access.Cloud computing is the concept implemented to remedy the Daily Computing Problems. Cloud computing is basically virtual pool of resources and it provides these resources to users via internet. It provides IT services as on-demand services, accessible from anywhere, anytime and by authorized user. It offers a range of

services for end users; among which there's Storage as a service. Storage as a service (SaaS) is a Cloud business model in which a service provider rents space in its storage infrastructure to individuals or companies. The data stored in the cloud can be sensitive to the business. The problematic is that these data are likely to be exploited by the provider or other unauthorized persons. Currently, most of cloud storage users protect their data with SLAs contracts and are based on the trust and reputation of the provider. This weakness has motivated us to think about solutions that enable users to secure their data to prevent malicious use.[3].

Cloud Computing is very flexible in nature that helps to quickly access the resources efficiently from the third party service provider to expand the business with low capitalization cost. A cloud storage system stores large number of data in its storage server. Since the data is stored for a long term over the internet it does not provide the data confidentiality and make the hackers to steal the data provided in the storage system and even when data forwarded to cloud environment, it lacks data integrity and makes the cloud user unsatisfied. In this paper, we study about different encryption technique to protect the cloud storage environment. The paper concisely covers some of the existing cryptographic approaches that can be used to improve the security in cloud environment4].

Cloud computing has become the latest technological computing area providing a large number of advantage to the different organization with its different business model at low cost. Nevertheless, there is always a security concern when uploading sensitive data in the cloud server. Client-side encryption is a common and better solution for ensuring end users that a third party user cannot access the uploading data. Cloud service providers maintain different techniques to protect data but like google drive, most of the company do not use client- side encryption. In the paper, They proposed a way to protect the data from hacking or losing when storing or uploading the data in the cloud server using the combination of Advanced Encryption Standard and Secure Hash Algorithm with Initial Vector[5].

Another paper describes a complete set of practical solution to file encryption based on RSA algorithm. With analysis of the present situation of the application of RSA algorithm, we find the feasibility of using it for file encryption. On basis of the conventional RSA algorithm, we use C + + Class Library to develop RSA encryption algorithm Class Library, and realize Groupware encapsulation with 32-bit windows platform. With reference of this

Groupware on Net platform, you can realize the window application of encryption operation on any files with RSA algorithm. RSA public key encryption algorithm is developed by Ron Rivest, Adi Shamir and Leonard Adleman in 1977, is the most influential public-key encryption algorithm, and has been recommended for ISO public key data encryption standard. RSA algorithm is an asymmetric cryptographic algorithm, the asymmetric, meaning that the algorithm requires a key pair, use one of the encryption, you need to be decrypted with another.RSA technology has formed a relatively complete international norm in all aspects of electronic security field. On the hardware side, it is used in a variety of sophisticated consumer electronics products with the mature IC technology. In terms of software applications, mainly in the Internet, RSA is widely used in encrypted connection, digital signatures and digital certificates core algorithms.[6].

Another paper introduced secure RSA for secure file transmission. There are many cases where we need secure file transmission for example in banking transactions, e-
shopping etc .In this paper we present modified RSA algorithm for secure file transmission. RSA algorithm is asymmetric key cryptography also called Public Key cryptography. Two keys are generated in RSA, one key is used for encryption &other key which is only known to authenticated receiver can decrypt message. No other key can decrypt the
message. Every communicating party needs just a key pair for communicating with any number of other communicating parties. Once someone obtains a key pair, he /she can communicate with anyone else. RSA is a well known public key cryptography algorithm and was one of the first great advances in public key cryptography. Even if it is efficient algorithm it is vulnerable to attackers. With the help of all brute force attacks hacker can obtain private key.Many improvements has been done to improve RSA like BATCH RSA, MultiPrime RSA, MultiPower RSA, Rebalanced RSA, RPrime RSA etc. As craze of internet is increasing exponentially, it is used for email, chatting, transferring data and files from one end to other. It needs to be a secure communication among the two parties.This paper focused on file transfer using Secure RSA, which eliminates some loopholes of RSA that might prevent a hacker from stealing and misuse of data. This paper also presentscomparison between RSA file transfer and Secure RSA file transfer. [7].

RSA can be used for key exchange as well as digital signatures and the encryption of small blocks of data. Today, RSA is primarily used to encrypt the session key used for secret key

encryption (message integrity) or the message's hash value (digital signature). RSA's mathematical hardness comes from the ease in calculating large numbers and the difficulty in finding the prime factors of those large numbers. Although employed with numbers using hundreds of digits, the math behind RSA is relatively straight-forward[8].

Client-side encryption is an effective approach to provide security to transmitting data and stored data. This paper proposed user authentication to secure data of encryption algorithm with in cloud computing. Cloud computing allows users to use browser without application installation and access their data at any computer using browser. This infrastructure guaranteed to secure the information in cloud server.[9].

There are important concerns when trusting sensitive information to the cloud. Health and financial records, for instance, suffer strict legal restrictions to data escrow. Organizations holding such information need to assure end-users and authorities that a third party will never access restricted data.In the paper they introduced a idea of Client-side encryption for privacy-sensitive applications on the cloud. Client-side encryption is a common solution in literature. Most works fail, however, to reason the impact of security solutions on performance and usability. Homomorphic and order preserving encryption systems can mitigate such negative impacts, as they allow the computation of regular searches over encrypted records on the cloud, while preserving information confidentiality and the privacy if end-users.[10].

CHAPTER 3

EXISTING SYSTEM

The existing research work introduced a system using symmetric key algorithm that would help to encrypt sensitive information locally at client-side. Then transmit the encrypted information to cloud storage to store it. To retrieve or use the information we have to download it from the cloud storage. When the download is completed, the information is decrypted by applying the same encryption keys which were stored in local storage.

The symmetric algorithm is developed using java with the Eclipse IDE and then implemented on Amazon S3. Amazon S3 provides constancy and flexibility that allows users to store as well as retrieve data at any moment and from anyplace using the web. At the beginning of the functional workflow, an Amazon S3 bucket and a folder inside the bucket was created in which encrypted files would be stored. Then program which was prewritten in java is run and choose a text file that will be encrypted. The program encrypted the text file using the existing algorithm and produce an encrypted file and an encryption key file which are kept safely on the local storage. When encryption is done, the file which contains ciphertext is uploaded to the Amazon bucket. The uploaded encrypted file has to go through Amazon's server-side encryption once again. Then download the file from the Amazon bucket to the local storage and decrypt the data file applying the encryption keys which was stored safely on local storage. When decryption is done, we get back our desired plain text which was secure in Amazon's S3 bucket as well as during transmission. The encrypted data was also not corrupted by Amazon's encryption and decryption process and survive through the process.

3.1 Drawbacks Of Existing System

- It cannot handle or process files other than text files
- Performance is low for large file.
- 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

3.2 Functional Workflow of existing system

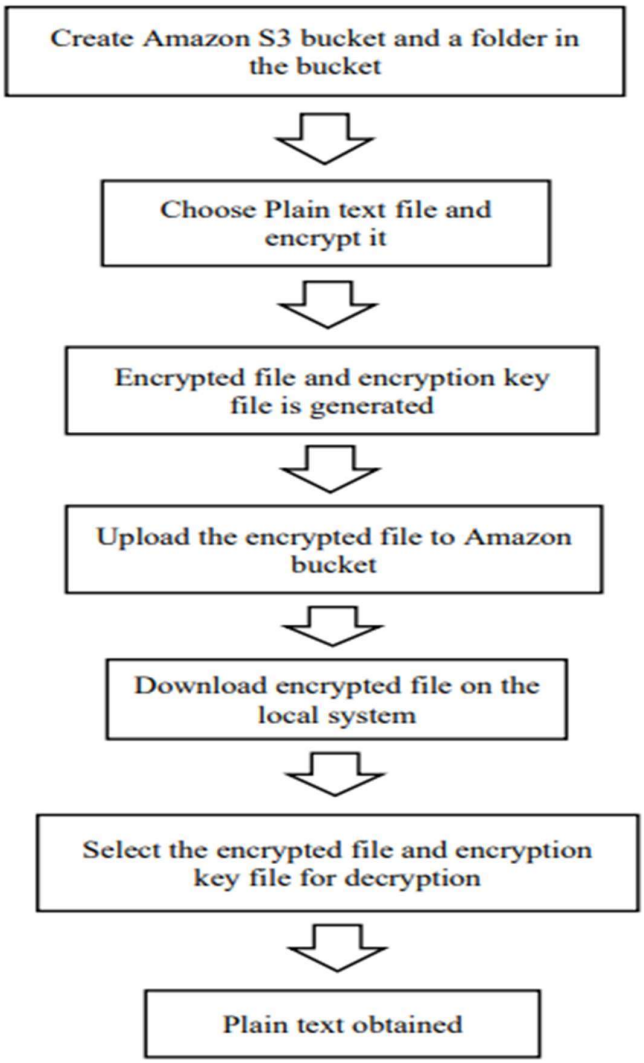


Fig.3.4.Existing system

CHAPTER 4
PROPOSED SYSTEM

The existing system is implemented to secure data before uploading to the cloud using client-side cryptographic method. This uses a symmetric cryptographic algorithm. Though this system provides security to data stored in the cloud, asymmetric cryptography proves to be more effective and difficult to crack. So here the proposed system uses asymmetric cryptographic algorithm to encrypt data at client side using the public key before uploading to the cloud .

When encryption is done, the encrypted file is uploaded to the Cloud. The uploaded encrypted file has to go through cloud server-side encryption once again. Then we download the file from the cloud to the 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 cloud as well as during transmission. The system introduces a portal through which the user can register followed by an OTP verification ,which is an authentication technique to provide extra security. Then after logging in the user can browse the file and encrypt the file. Then the user can decrypt the file by receiving another mail containing the private key link that is sent to the user's mail-id. By uploading the private key, the user successfully completes the authentication, and the decrypted file will get downloaded.

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. One of the main benefits of using RSA for client-side encryption is that it allows the client to encrypt data without having to share a secret key with the server. This means that the client can send the server an encrypted message without the server being able to decrypt it, even if the server is compromised. Another benefit of RSA is that it is relatively fast and efficient, especially compared to some other public-key cryptosystems.

4.1 Merits of proposed system

- It overcomes the biggest flaw of symmetric key cryptography since it doesn't need to exchange any keys that can decrypt data.
- Since Assymmetric algorithm links the private and public keys together, a message is decrypted using a private key. It stands as evidence that the message originated from the rightful owner who has the private key
- 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.
- Since we are using RSA algorithm, it is safe and secure for transmitting confidential data.
- This system can encrypt not only the text files,but also images.It can handle pdf,docx,png,jpeg.

CHAPTER 5

SYSTEM DESIGN ARCHITECTURE

5.1ARCHITECTURE DIAGRAM

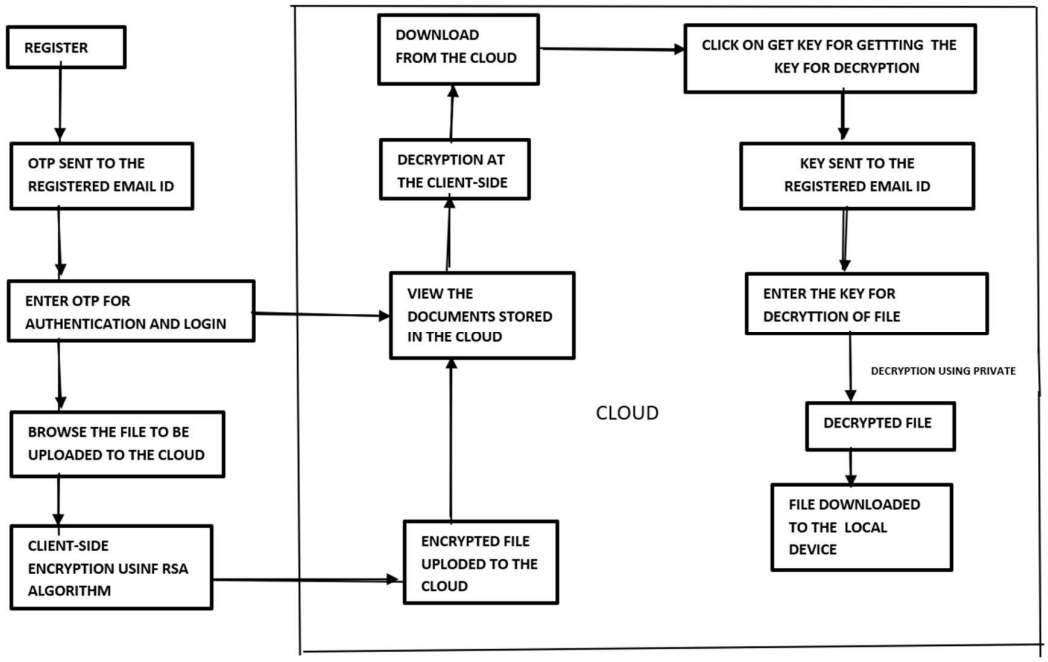


Fig5.1 Proposed system

5.2.ADMIN LEVEL FUNCTIONS

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

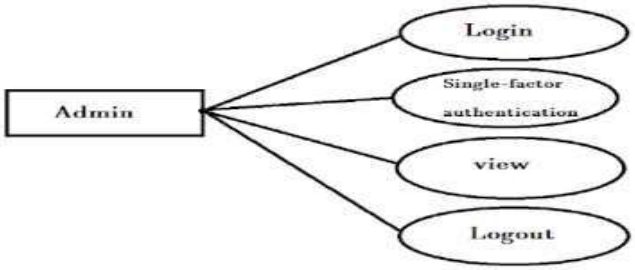


Fig.5.2 Admin Level Functions

5.3.USER LEVEL FUNCTIONS

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

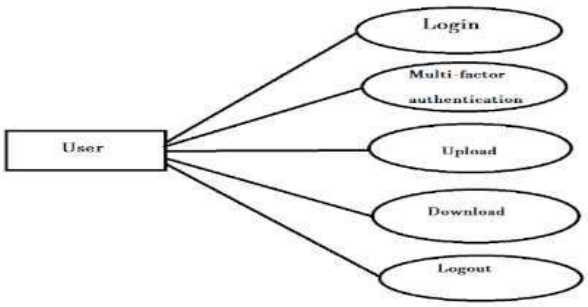


Fig.6.2.User Level Functions

5.4 USER REGISTRATION

New users can create an account on the registration page. A user/owner can register or sign in by providing the necessary credentials like username, password etc in the registration/login page. After the registration, an account for the user is successfully created by the system.

Fig.6

5.4 Registration

5.5.USER LOGIN

The registered users/owner/admin can log in or sign up to the system. In order to log in/signup, the user/owner/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.

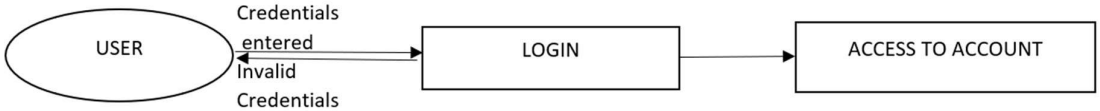


Fig.5.5 login

5.6.UPLOADING FILES INTO THE SYSTEM

The user has upload permission. Different types of files such as pdf, ppt, audio, video, etc can be uploaded. The file selected gets encrypted using an RSA algorithm. The encrypted file gets stored in the cloud and the key of the RSA algorithm is send to the user's registered mail id for decrypting it after it has been taken out of the cloud for decryption. This is done after the 2nd stage of encryption is completed within the cloud.

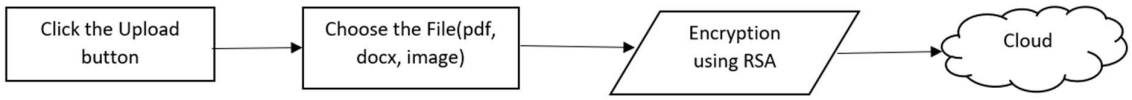


Fig 5.6 Uploading

5.7.DOWNLOADING FILES FROM THE SYSTEM

The user requests a file. The user has to click on "view the Documents". The user will then receive mail containing the private key. The user has to enter the key received and upload the private key file. If the code entered and private key match with the values stored in database the file gets decoded.

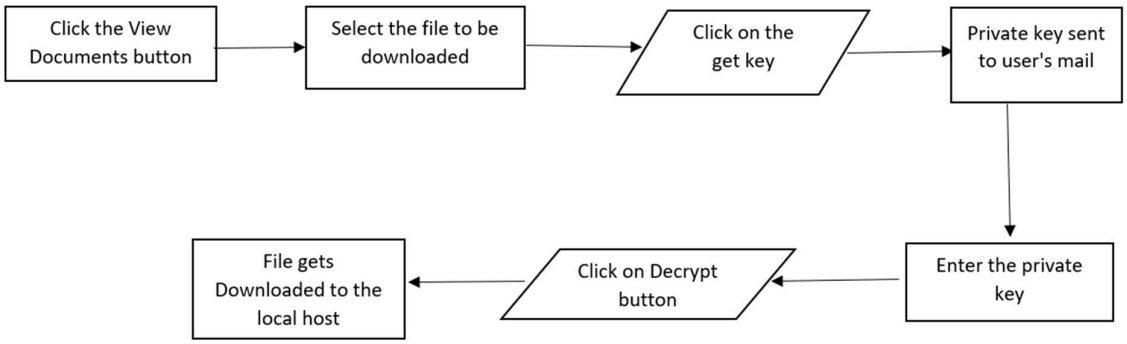


Fig 5.7.Downloading

CHAPTER 6
SYSTEM REQUIREMENTS

6.1 SOFTWARE REQUIREMENTS

i. Front end: HTML, CSS

ii. Back end:

- Back-end Database: MySQL
- Back-end Scripting: Python
- Back-end Server: Apache

iii. Operating System:

- Client: Windows 10,11

iv. Cloud Platform: MilesWeb

6.2 HARDWARE REQUIREMENTS

i. System

- Windows 10 PC/Laptop
- RAM: 4 GB
- System-type: 64-bit
- Network Connectivity

CHAPTER 7

MODULE DESCRIPTION

The CEDOCS System consists of 3 modules:

1.Registration & login

2.Upload and Client-side Encryption using RSA

3.Decryption and Download

7.1 Module I: Registration & Login

This section includes the front-end of the "CEDOCS:"

The front end consists of a login page, where there is an option to register for new users. There are two types of accounts that can be created with different levels of access. They are admin and user. The admin can view all the users. The users can only upload and download the files they want. The users once registered can login to their respective accounts. After login they are directed to a dashboard where view document and upload options are available.

7.2 Module 2: Client-side Encryption using RSA and Uploading

The admin and the owner have the privilege to upload and download files. They can get their private key which is required to download the files through their registered email id when they click on the 'Get key' option available in the view documents 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 Cloud Storage.

The cryptographic algorithm used here is the RSA encryption. RSA is the most common public-key algorithm, named after its inventors Rivest, Shamir, and Adelman (RSA).

RSA is a specific algorithm for public-key encryption that is widely used for secure data transmission. It uses a pair of keys - a public key and a private key - to encrypt and decrypt data.

The setup of an RSA cryptosystem involves the generation of two large primes, say p and q, from which, the RSA modulus is calculated as n = p * q. The greater the modulus size, the higher is the security level of the RSA system. The recommended RSA modulus size for most settings is 2048 bits to 4096 bits.

RSA algorithm uses the following procedure to generate public and private keys:

Select two large prime numbers, p and q.

Multiply these numbers to find n = p x q, where n is called the modulus for encryption and decryption.

Choose a number e less than n, such that n is relatively prime to (p - 1) x (q -1). It means that e and (p - 1) x (q - 1) have no common factor except 1. Choose "e" such that 1<e < φ (n), e is prime to φ (n),

gcd (e,d(n)) =1

If n = p x q, then the public key is <e, n>. A plaintext message m is encrypted using public key <e, n>. To find ciphertext from the plain text following formula is used to get ciphertext C.

C = m^e mod n

Here, m must be less than n. A larger message (>n) is treated as a concatenation of messages, each of which is encrypted separately.

To determine the private key, we use the following formula to calculate the d such that:

De mod {(p - 1) x (q - 1)} = 1

Or

De mod φ (n) = 1

The private key is <d, n>. A ciphertext message c is decrypted using the private key <d, n>. To calculate plain text m from the ciphertext c the following formula is used to get plain text m.

m = c^d mod n

Module 3: Downloading and decryption

The user can view all the files uploaded in the 'View Documents' section. And there is a 'Get Key' for each of the files. User clicks on that button of which file we need to download. 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 downloaded in decrypted format.

CHAPTER 8

IMPLEMENTATION

CEDOCS: Client-side Encryption of Data On Cloud Storage uses asymmetric cryptographic algorithm to encrypt data at client side using the public key using the R.A encryption before uploading to the cloud .

When encryption is done, the encrypted file is uploaded to the Cloud. The uploaded encrypted file has to go through cloud server-side encryption once again. Then we download the file from the cloud 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 cloud as well as during transmission.

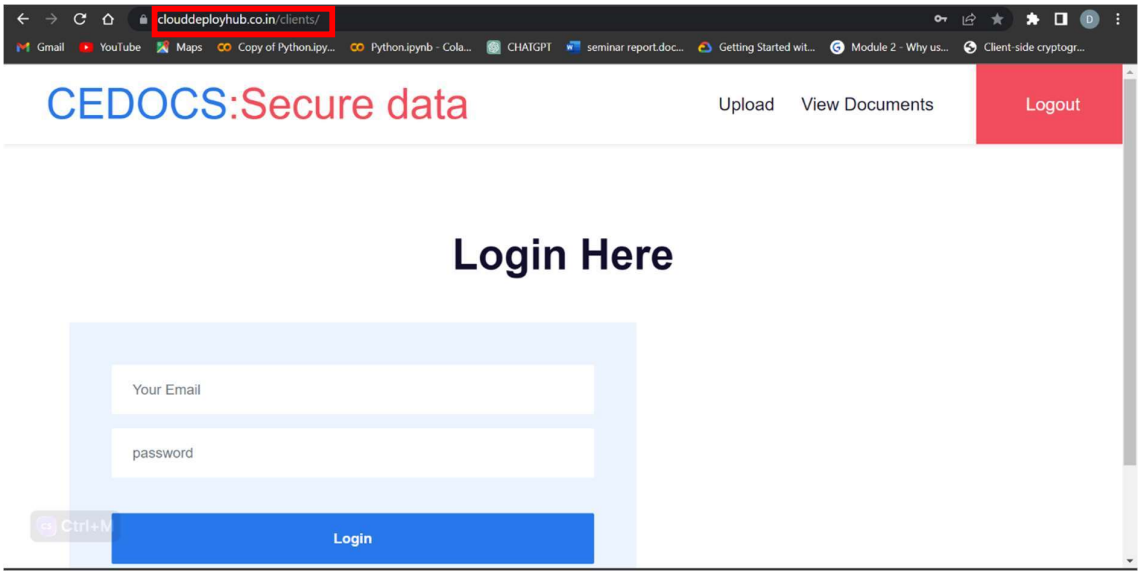
The system introduces a portal through which the user can register, followed by an OTP verification ,which is an authentication technique to provide extra security.

So, after login in the user can see a page where he can enter his username and password. If the login is successful, there are two options one is upload and another one is view document. To upload a new file, which is in the form of PDF, word document or images, 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 cloud. When he clicks the 'View document' option he can see a page where there is a table listed with the files that are uploaded with their description and a 'get key' button along with each one. In order to download a particular file, click the 'get key button' near to each file. Then the user will get the private key generator during the RSA encryption via his registered email ID. Then he will be redirected to a new page where he can decrypt and download the file. There is a field to enter the private key. Now the user can copy the private key received via the mail and paste in that field and click on the encrypt button. Now the file will be downloaded to his device which is in the decrypted format.

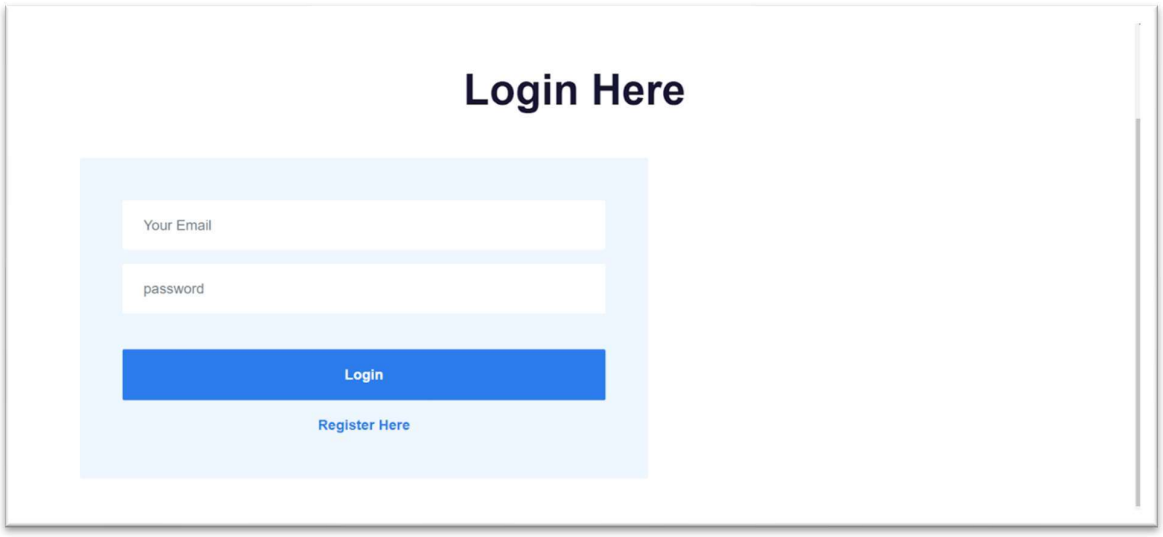
STEPS TO USE THE SYSTEM

1.Click on the url: clouddeployhub.co.in/clients/

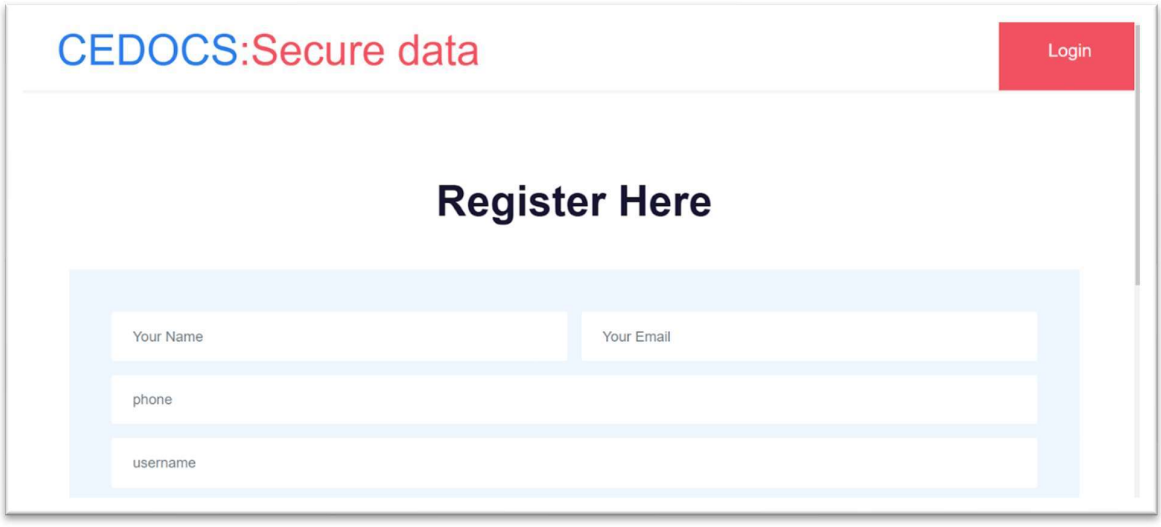
The login page displays the option to login by entering the email id and password.



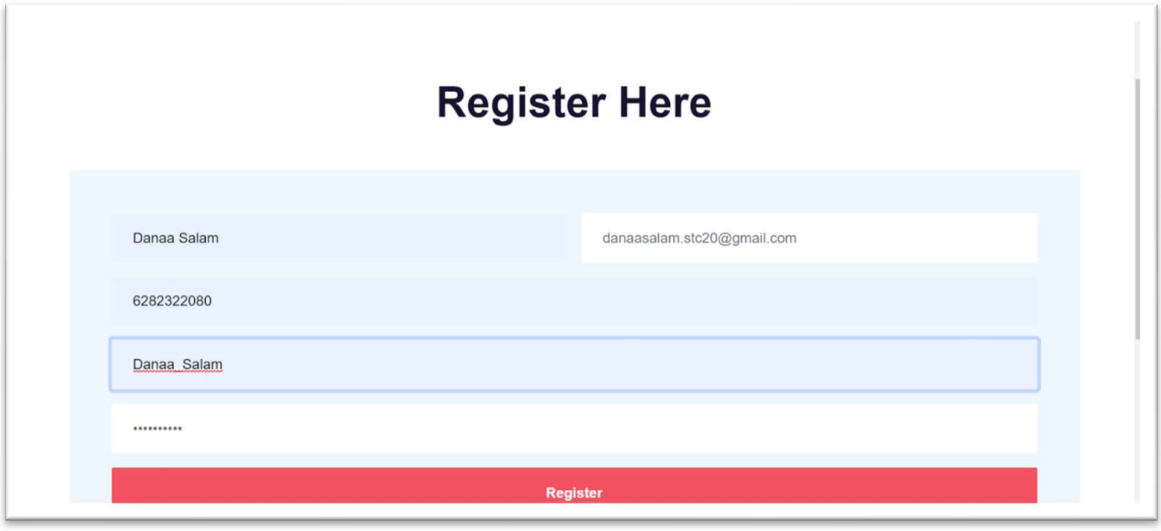
2.For new users, register here option can be selected



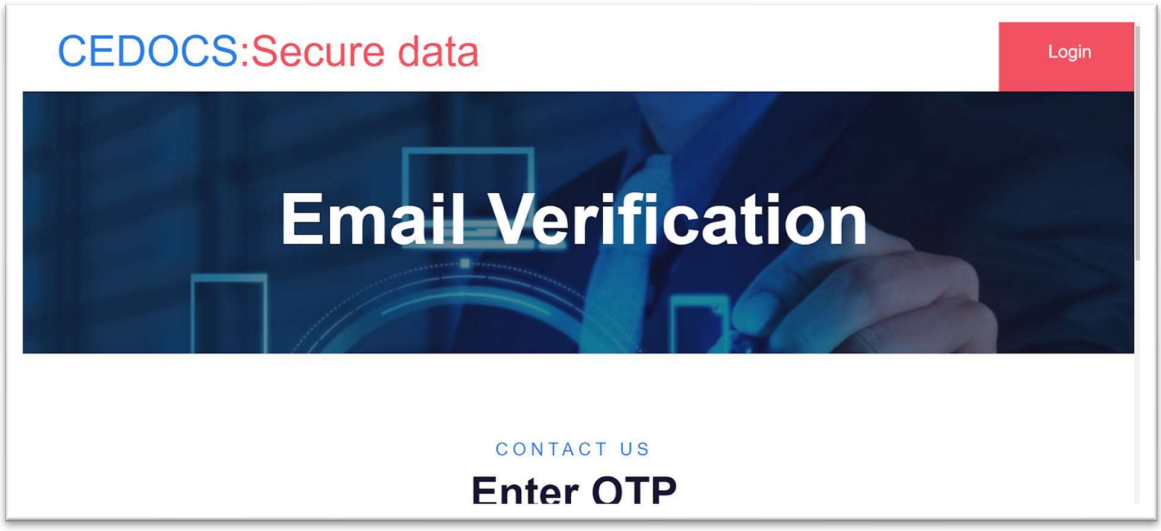
3.Upon clicking **Register here** option, take the user to a new registration page.



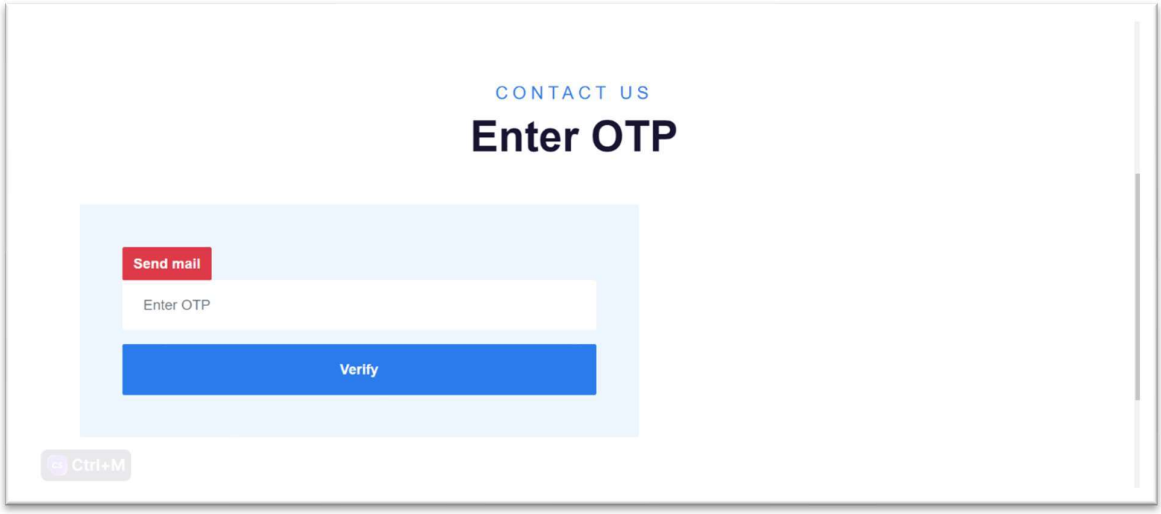
4.The new user needs to enter the name, email address, phone number and username. Then click **Register** button.



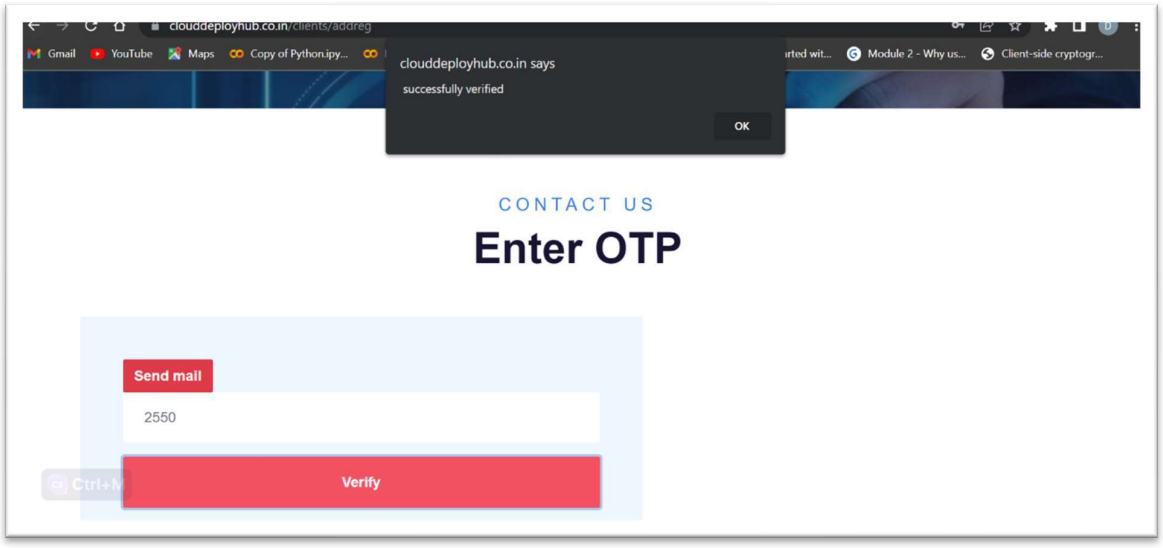
5.Now the user will be directed to an email verification page



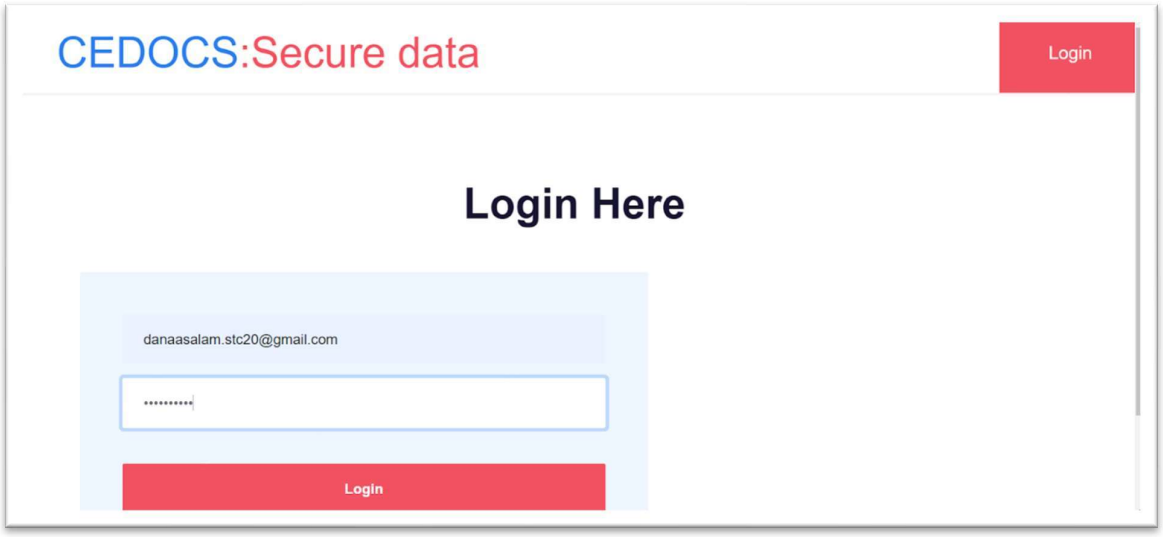
6.Click on sent mail button and then Enter the OTP received through the registered email id.



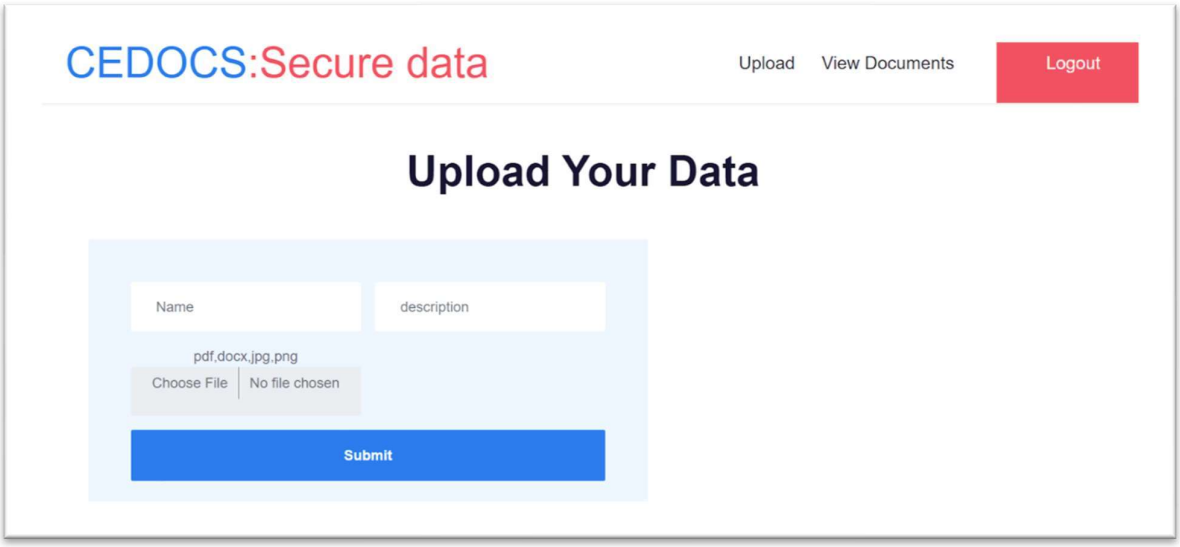
7.After clicking on verify button, a pop up appears showing that the user successfully verified



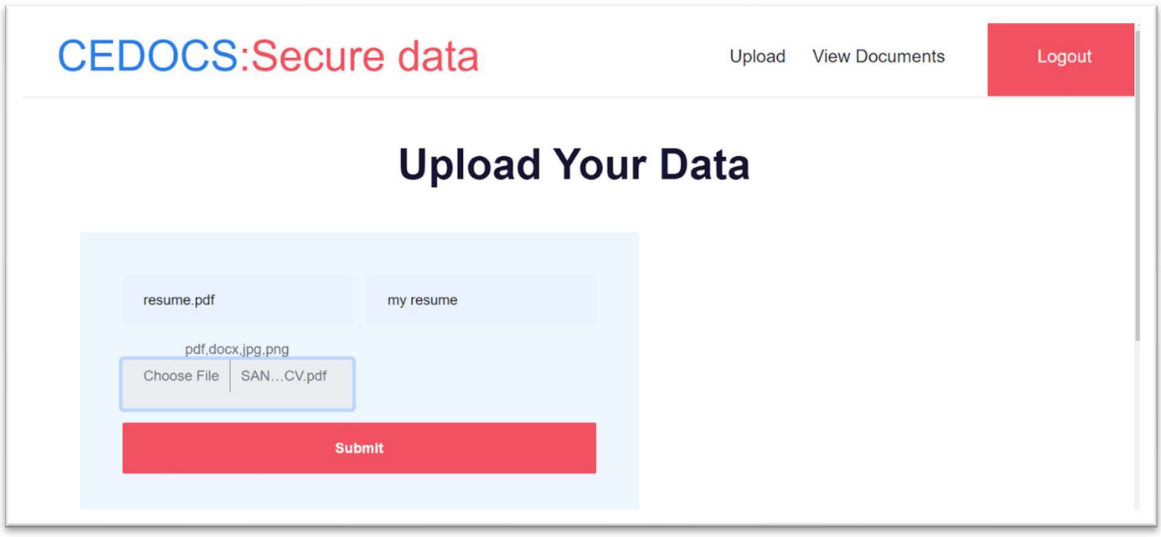
8.Now the registered user can enter their email id and password to login



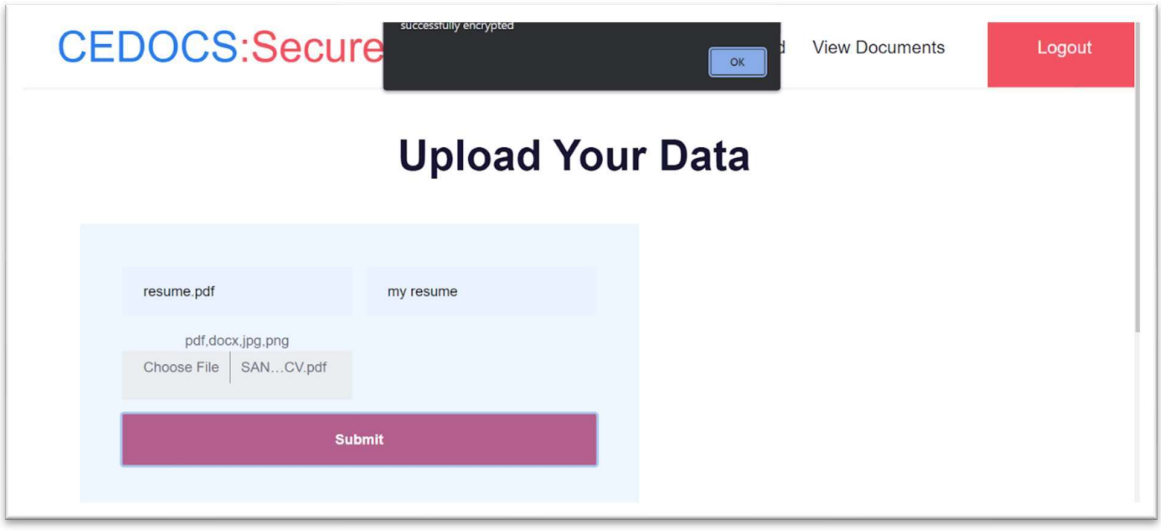
9.Click on upload tab to upload the required files from the device to the cloud



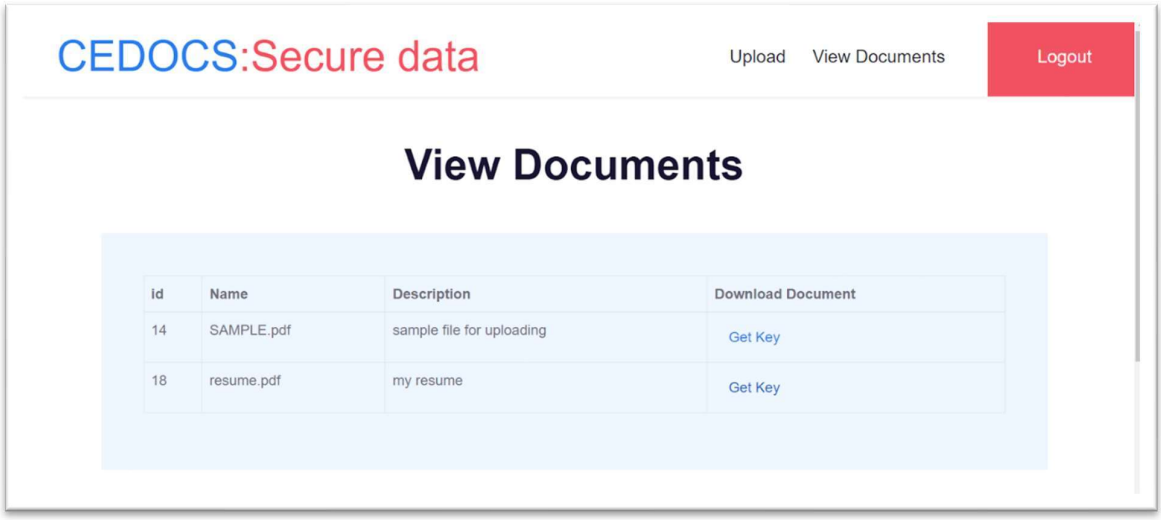
10,Enter the name and description of the file .And the browse the file to be uploaded. It can be image, document or pdf file



11.After clicking on submit button, a prompt message appears showing that successfully encrypted.

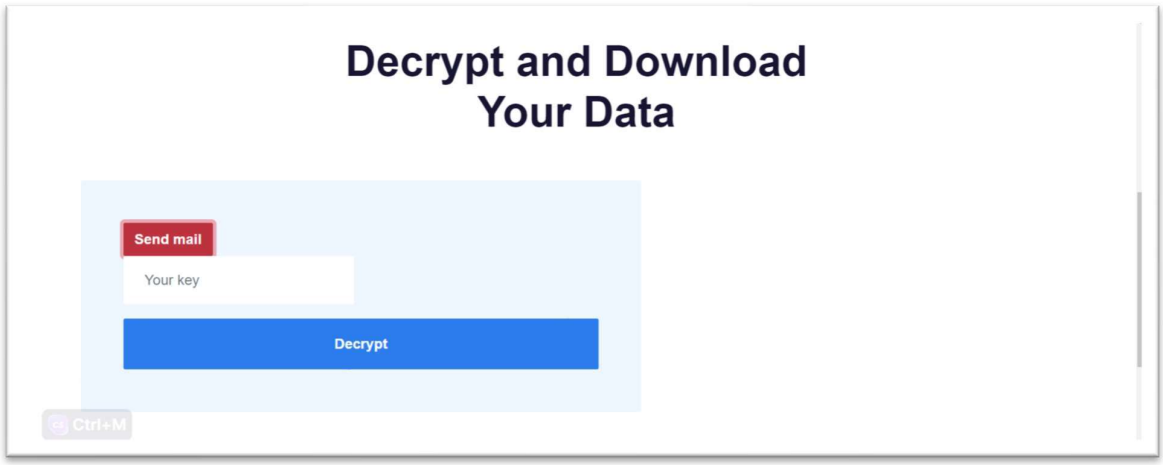


12,Click on view document tab to view the documents that are encrypted and uploaded to the cloud.Inorder to decrypt and download any files, the user can click on the get key option

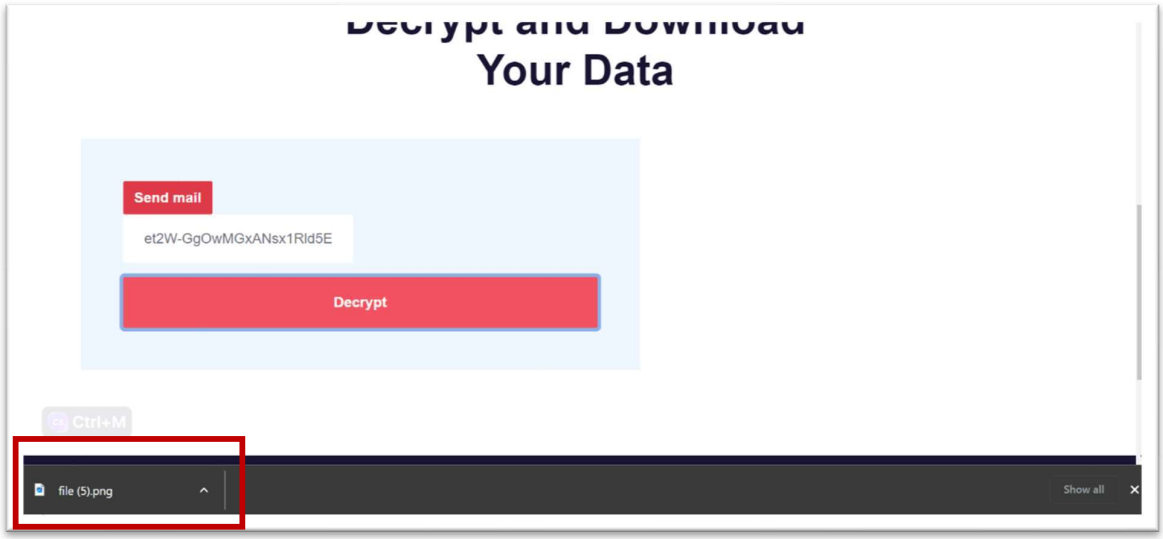


13.Upon clicking the Get key option, the user will be directed to a decryption page, where the user can enter the private key for decryption of the file.

14.Now click on the send mail button to receive the key.The private key will be received through the registered mail id .Enter that private key in the 'Your Key' field. And click on Decrypt button.



15.Now the file will be downloaded to the Client device and decrypted automatically.



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 cloud.The project would aim to address the security concerns associated with cloud storage by providing a secure and user-friendly solution for encrypting data before it is uploaded to the cloud.

Before uploading the data to the cloud, the system would encrypt the data using the client's public key. The encrypted data would then be uploaded to the cloud 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 private key. 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 encrypted file which is in the form of pdf,docx or image in to the cloud storage and stores it 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

```
from django.http import HttpResponse
from django.shortcuts import render
#from .models import testtb
from django.shortcuts import redirect
# FILE UPLOAD AND VIEW
from  django.core.files.storage import FileSystemStorage
# SESSION
from django.conf import settings
from .models import *
import rsa
from ML import rsa_encryption, mail,tmail
import base64
import random
import mimetypes
#from django.core.mail import EmailMessage
#import smtplib
def first(request):
   return render (request,'login.html')
def index(request):
   return render (request,'login.html')
def reg(request):
   return render(request,'register.html')
def addreg(request):
   if request.method == 'POST':
      name=request.POST.get('name')
         username=request.POST.get('username')
         email=request.POST.get('email')
         phone=request.POST.get('phone')
         password=request.POST.get('password')
```

```
        user=register(name=name,username=username,email=email,phone=phone,password=
password,status="pending")
        user.save()
        otp=random.randint(1000,9999)
        request.session['otp']=otp
        request.session['email']=email

        message = """From: No Reply <test@clouddeployhub.co.in>
To: Person <{}>
Subject: Test Email

""".format(email)
        tmail.mail_send([email],message+"your otp is:{}".format(str(otp)))
        return render(request,'otp_verification.html',{'mail_id':email,'otp':otp})

def otp_verifcation(request):
        if request.method == 'POST':
                usr_otp=request.POST.get('otp')
                otp=request.session['otp']
                user=register.objects.get(email=request.session['email'])
                if str(usr_otp)==str(otp):
                        user.status="verified"
                        user.save()
                else:
                        user.delete()
                return render(request,'register.html')
def login(request):
   return render(request,'login.html')
def logint(request):
   email = request.POST.get('email')
   password = request.POST.get('password')
   if email == 'admin@gmail.com' and password == 'admin':
```

```
        request.session['logintdetail'] = email
        request.session['logint'] = 'admin'
        return render(request, 'admin/index.html')

    elif register.objects.filter(email=email,password=password, status='verified').exists():
        userdetails=register.objects.get(email=request.POST['email'], password=password,
status='verified')
        if userdetails.password == request.POST['password']:
            request.session['uid'] = userdetails.id
            request.session['uname'] = userdetails.name
            request.session['uemail'] = email
            request.session['user'] = 'user'
            return render(request,'index.html')
    else:
        if request.POST:
            return render(request, 'login.html', {'res': {'status':'Invalid Email Or Password '}})
        else:
            return render(request, 'login.html')
def logout(request):
    session_keys = list(request.session.keys())
    for key in session_keys:
        del request.session[key]
    return redirect(first)

def up(request):
    return render(request,'uploading.html')
def view_docs(request):
            sel=document.objects.filter(user_id=request.session['uid'])
            #print(sel)
            return render(request,'view_documents.html',{'res':sel})

def download_document(request,id):
    sel=document.objects.get(id=id)
```

```
    request.session['doc_id'] = id
    doc_name=sel.name
    doc_desc=sel.description
    #print(doc_name)
    message = """From: No Reply <test@clouddeployhub.co.in>
To: Person <{}>
Subject: Test Email

""".format(request.session['uemail'])
    tmail.mail_send([request.session['uemail']],message+"Your document key i
:{}".format(str(sel.key)))
    #mail.send_otp(str(request.session['uemail']),str(sel.key))
    return
render(request,'download_document.html',{'mail_id':request.session['uemail'],'otp':str(sel.key
)})

def imageup(request):
        if request.method == 'POST':
                name=request.POST.get('name')
                description=request.POST.get('description')
                myfile = request.FILES['photos']
                key=rsa_encryption.generate_enc_key()
                original_base64 = base64.b64encode(myfile.read())
                #print("\ndata:\n\n",original_base64)
                encrypted_base64=rsa_encryption.encrypt_file(original_base64,key)
                #print("\nencrypted data:\n\n",encrypted_base64)
                #fs = FileSystemStorage()
                #filename = fs.save(myfile.name,myfile)
        user=document(user_id=request.session['uid'],key=key.decode(),document=encrypted
_base64.decode(),name=name,description=description)
                user.save()
                return render(request,'uploading.html')
```

```
def decrypt(request):
        if request.method == 'POST':
                key=request.POST.get('key')
                key=bytes(key,'utf-8')
                #print(key)
                #print(type(key))
                sel=document.objects.get(id=request.session['doc_id'])
                name=sel.name
                #name="doc.docx"
                encrypted_doc=bytes(sel.document,'utf-8')
                decrypted_base64=rsa_encryption.decrypt_file(encrypted_doc,key)
                decrypted=base64.b64decode(decrypted_base64)
                filename="file."+name.split(".")[-1]
                with open("media/"+filename, 'wb') as dec_file:
                        dec_file.write(decrypted)
                path = open("media/"+filename, 'rb')
                mime_type, _ = mimetypes.guess_type("media/"+filename)
                response = HttpResponse(path, content_type=mime_type)
                response['Content-Disposition'] = "attachment; filename=%s" % filename
                return response
def dash(request):
   return render(request,'admin/index.html')
def userview(request):
   sel=register.objects.all()
   return render(request,'admin/viewuser.html',{'res':sel})
def users(request):
   return render(request,'admin/user.html')
def adduser(request):
   if request.method == 'POST':
      name=request.POST.get('name')
      username=request.POST.get('username')
      email=request.POST.get('email')
      phone=request.POST.get('phone')
```

```
        password=request.POST.get('password')
user=register(name=name,username=username,email=email,phone=phone,password=passwo
rd)
        user.save()
        return render(request,'admin/user.html')
'''def texts(request):
    return render(request,'usertext.html')
def addtext(request):
    if request.method == 'POST':
        message=request.POST.get('message')
        name=request.POST.get('name')
        publicKey, privateKey = rsa.newkeys(512)
        privateKeyPkcs1PEM = privateKey.save_pkcs1().decode('utf8')
        encMessage = rsa.encrypt(message.encode(),publicKey)
        print(publicKey, privateKey,encMessage)
        user=text(user_id=request.session['uid']
,name=name,public_key=publicKey,private_key=str(privateKeyPkcs1PEM),message=encMe
ssage)
        user.save()
    return render(request,'usertext.html')

def view_text(request):
            sel=text.objects.filter(user_id=request.session['uid'])
            print(sel)
            return render(request,'view_text.html',{'res':sel})

def decrypt_text(request):
            if request.method == 'POST':
                        key=request.POST.get('key')
                        privateKeyReloaded = rsa.PrivateKey.load_pkcs1(key.encode('utf8'))
                        decMessage = rsa.decrypt(encMessage, privateKeyReloaded).decode()
            return render(request,'usertext.html')'''
```

CRYPTOGRAPHY

```
#importing required python libraries or packages..
from cryptography.fernet import Fernet
def generate_enc_key():
        # key generation
        key = Fernet.generate_key()   return key
def encrypt_file(original,key):
        # using the generated key
        fernet = Fernet(key)
        # encrypting the file
        encrypted = fernet.encrypt(original)
        return encrypted
def decrypt_file(encrypted,key):
        # using the generated key
        fernet = Fernet(key)
        # decrypting the file
        decrypted = fernet.decrypt(encrypted)
        return decrypted
"""
key=generate_enc_key()
print("\nGenerated Key:\n\n",key)
# opening the original file to encrypt
with open('test.docx', 'rb') as file:
        original_base64 = base64.b64encode(file.read())
encrypted_base64=encrypt_file(original_base64,key)
print("\nEncrypted_data:\n\n",encrypted_base64)
decrypted_base64=decrypt_file(encrypted_base64,key)
print("\nDecrypted_data:\n\n",decrypted_base64)
decrypted=base64.b64decode(decrypted_base64)
# opening the file in write mode and
# writing the decrypted data
with open('decrypted.docx', 'wb') as dec_file:
        dec_file.write(decrypted)"""
```

REFERENCES

[1] 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),March 2021

[2] Sunil Sanka,Chittaranjan Hota,Muttukrishnan Rajarajan,"Secure Data Access in Cloud Computing", Birla Institute of Technology and Science-Pilani,December 2019

[3] Zaid Kartit, Ali Azougaghe, H. Kamal Idrissi, M. El Marraki, M. Hedabou, M. Belkasmi and A. Kartit "Applying Encryption Algorithm for Data Security in Cloud Storage",February 2016

[4] R. Kirubakaramoorthi, D. Arivazhaganand D. Helen, "Survey on Encryption Techniques used to Secure Cloud Storage System",AMET University,May 2020

[5] Md. Mahidul Islam,Md. Zahid Hasan, "A Novel Approach for Client Side Encryption in Cloud Computing",Daffodil International University, February 2019

[6] Suli Wang,Ganlai Liu, "File encryption and decryption system based on RSA algorithm" International Conference on Computational and Information Sciences-2011

[7] Rajan.S.Jamgekar, Geeta Shantanu Joshi,"File Encryption and Decryption Using Secure RSA",International Journal of Emerging Science and Engineering (IJESE)ISSN: 2319–6378, Volume-1, Issue-4, February 2013

[8] Gary C. Kessler Embry-Riddle,"An Overview of Cr view of Cryptography (Updated V aphy,Aeronautical University(Updated Version, 3 March 2016)"

[9] Mr. Niteen Surv, Mr. Balu Wanve, Mr. Rahul Kamble,"Framework for Client Side AES Encryption Technique in Cloud Computing", IEEE International Advance Compaing Conference (IACC) 2015

[10] Stefano M P C Souzaa, Ricardo S Puttinia, "Client-side encryption for privacy-sensitive applications on the cloud",Universidade de Brasılia-DF 70910-900, October 2016