Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 10, Issue. 5, May 2021, pg.49 – 59

Secure File Storage & Sharing on Cloud Using Cryptography

Madhumala RB[1]; Sujan Chhetri[2]; Akshatha KC[3]; Hitesh Jain[4]
Faculty of Engineering & Technology, Jain University, Bangalore, India
[1] madhumala8887@gmail.com; [2] sujanxchhetri@gmail.com; [3] akshathakcjain11@gmail.com; [4] hiteshjain1699@gmail.com
DOI: 10.47760/ijcsmc.2021.v10i05.005

Abstract— In today's world, simply having the capacity to transfer a file from one location to another isn't enough. Businesses today face multiple security threats and a highly competitive environment. So they need a secure file transfer system to protect and reliably transfer their sensitive, business-critical data. Secure file transfer is a method of data sharing via a secure, reliable delivery method. Also, we use this between a client and a server. Cryptography is a technique that we use for securing information and communication in the presence of third parties. We use this technique to ensure that only those persons to whom the information is intended can read this. By using cryptography, we can prevent unauthorized users from accessing the information which is shared privately. In this paper, the plan proposed is to overcome the issues regarding the data that are being stored by the users on the cloud should be encrypted rather than storing them in a plain form such that the data will be protected from the attackers who are trying to read, delete or manipulate the data. Our application is focused on securely authenticating the user, before storing and sharing files, To create an application that lets a user encrypt and decrypt any type of file without any changes in the size during encryption & decryption, store every user data in the encrypted form on the cloud, to provide a communication medium between users via the chat application, to give direct access to the file for CRUD operation only to the owner.

Keywords— Cryptography, Cloud Computing, MongoDB, GoogleOauth2.0, Web Development

I. INTRODUCTION

Data sharing via a cloud platform is popular. With a highly encrypted cloud platform, today's need for secure transmission can be achieved. Focusing on Reliable and Secure Data sharing our research involves building a cloud platform that relies on Node.js, Express.js, MongoDB Atlas, GoogleOauth2.0, crypto.js, Crypto, Heroku, and so on. We are using a cloud computing model that stores data on the Internet through a cloud computing provider who manages and operates data storage as a service. This gives us agility, global scale, and durability, with "anytime, anywhere" data access. One of the cloud Storage which we are going to Use is Mongo DB Atlas which is a fully managed cloud database for modern applications. The Socket.IO enables real-time bidirectional event-based communication. It works on every platform, browser, or device, focusing equally on reliability and speed. Socket.IO is built on top of the WebSockets API (Client-side) and Node.js (used for Server-side). The Socket.IO also helps us to create a chat app in just a few lines of code for Instant messaging.

Heroku is a platform as a service that enables developers to build, run, and operate applications entirely in the cloud which can be used to deploy, manage, and scale modern apps. In this application, we are using the AES

encryption algorithm (256-Bit) because it is faster and secure than other available symmetric encryption algorithms. We aim to provide a Secure Cloud Platform as a Web Application for file encryption, storage & sharing.

II. LITERATURE REVIEW

As we have done so much research related to our project, we found that Madhu Sharma & Dr. Ashish Sharma[1] showed that a Chaotic map based encryption algorithm can encrypt/decrypt the data being shared with the different types of data. Songling Fu et al[2] showed how we can Provide secure efficient fine grained sharing using RES and DES. Akhil K.M ,Praveen Kumar And MPushpa B.R[3] showed that AES based secure model for cloud data security provides increased security to data while being stored and transferred. Isji Karabey And Gamze Akman[4] Showed that In this encryption and identity authentication have been added to a simple chat application to let clients talk to each other instantly over a secure channel in the internet. Bharati Mishra and Debsish Jena.[5] provided End-to end security for files while sharing them. After performing so many researches we have found few limitations that, The size of each of the generated shares varies slightly in comparison to the input plaintext, Backup may be slower, There is Higher internet utilization, there are only Limited methods provided for file sharing, RSA & DSA algorithm can be very slow in cases where large data needs to be encrypted by the same computer, There is Data leakage problem, there is a Increase in the decrypted file size, Encryption methods are Limited to file types and Only few approaches are applied to Real time application.

III. CURRENT LIMITATIONS

Based on the literature survey, we identified that there is a high demand for secure sharing and storage mechanisms. The data being stored and shared are vulnerable to attacks. The plain form of the files makes it more easier to target the user than the encrypted. It may be a text file or personal user information, everything needs to be encrypted and shared via a secure medium. However, most of the users are unaware of these kinds of tools and mechanisms and if they are aware then are not using them effectively. By studying and analysing many users over the internet we got to know that many files being shared are vulnerable to attacks as they are not encrypted and are easy to target.

IV. OBJECTIVES

- To authenticate a user securely with GoogleOauth2.0 and create a profile of the authenticated user.
- To provide secure access to the stored files i.e. only the authenticated user has access rights to their file.
- To provide the secured transmission of files between the users.
- To securely share the decryption password through the chat client.
- To store the data in encrypted form on the cloud.
- To provide a Secure communication between users.

V. PROPOSED SYSTEM

Our application is focused on securely authenticating the user, before storing and sharing of files. We are trying to create an application that lets a user encrypt and decrypt any type of file without any changes in the size during encryption & decryption. The application stores every user data in the encrypted form on the cloud. The application provides a communication medium with the chat application. Direct access to the file for CRUD operation is only allowed to the owners.

ARCHITECTURE DIAGRAM

The Architecture diagram for the proposed system has a client-server architecture. All the encrypted files are stored in a cloud-based database i.e. MongoDB Atlas. The user authentication data is fetched from google and stored a few of the required data into our database. The files stored can be accessed by the users using this proposed web application from the database. In this web application, users can store and share the file in an encrypted format. The password-based encryption tool lets users enter a unique set of passwords to encrypt the file. Before storing the data into the database it is encrypted with the bcrypt module. CRUD operation is only possible by the owner on their file. For sharing the files with others, users get a chat client. Users can create a custom room and invite another user and have a chat on sharing the file and decrypting it with a password. The chat application details can be forwarded via the form. Through that chat application, the client user can share the password in a secure manner and the file can be downloaded and decrypted by another user.
The whole tool and application is pushed to Github and is deployed to Heroku via continuous integration. Any changes in the codebase automatically gets pushed to GitHub and deployed to Heroku in real-time.

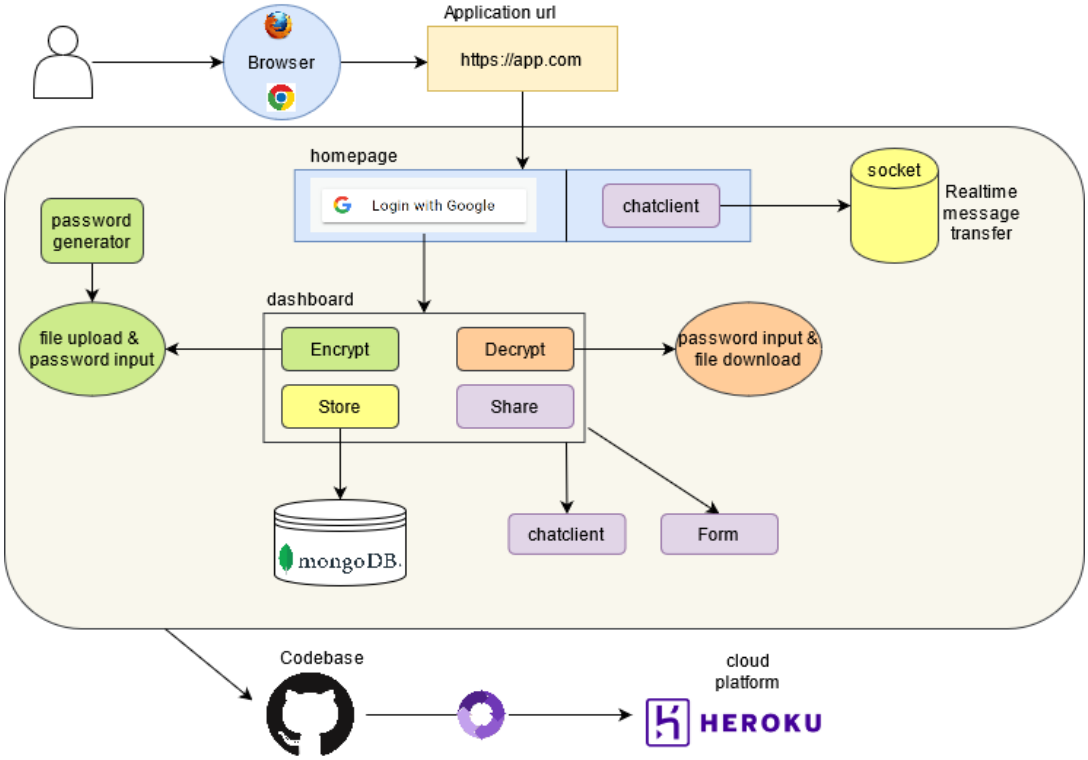

Fig. 1 Architecture diagram for the proposed system

ACTIVITY DIAGRAM

The diagram surfacely gives an overview of the project flow. Each step or block, when bootstrapped together, will result in the platform that we are trying to build for the secure transmission of files over the cloud as a web application.

The user must be authenticated before the whole process occurs. The user will be redirected to google's login form where the user must enter a valid email id and password to be authenticated and use the application. If the user provides an invalid credential then the login will fail and the end-user won't be able to log in. The user's profile is created once the user is securely authenticated.

Once the user's credential is validated the user data is stored in our MongoDB database. The profile image and username are fetched from the data stored in the MongoDB database. Each user profile gets a unique id which is used to authenticate them to their own dashboard.

The dashboard enables a user to access the user's stored files and the encryption tool. The encryption tool can be accessed via the encrypt button. The user gets redirected to the tool once the button is clicked. Select a file to encrypt. Enter the password or Copy from the password generator. Download the encrypted file.

The form is used to send the chat room detail or the file link to the end-user. The form is integrated with the Gmail client. The user can enter the mail of another user and add a subject. The content can be added below the subject. The send email button lets the user send the mail once clicked.

The chat client is the implementation of socket programming which enables users to send and receive data via the web sockets. The chat client is used by the users to securely transmit the encrypted password and the file download link. The chat data is cleared once the user leaves the room. A user can create a room and share the room id with another user to join the chat. The user can share each other's locations as well with the location button.

The user will upload the encrypted file such that the downloadable link will be generated. The generated link can be used to download the file. The same link is shared via any of the sharing tools. The links get expired after 24 hours and the user has to reupload and reshare the link in case the link expires.

The user will upload the file which can be accessible later. The user can select a file and submit it to upload the file for further use. Once the user uploads the file the file is available in this section. Users can download and use the file later. The user can also delete the file if not required.

The encryption tool can be accessed via the decrypt button. The user gets redirected to the tool once the button is clicked. Select a file to decrypt. Enter the password. Download the decrypted file.

The users log out once the task is done.

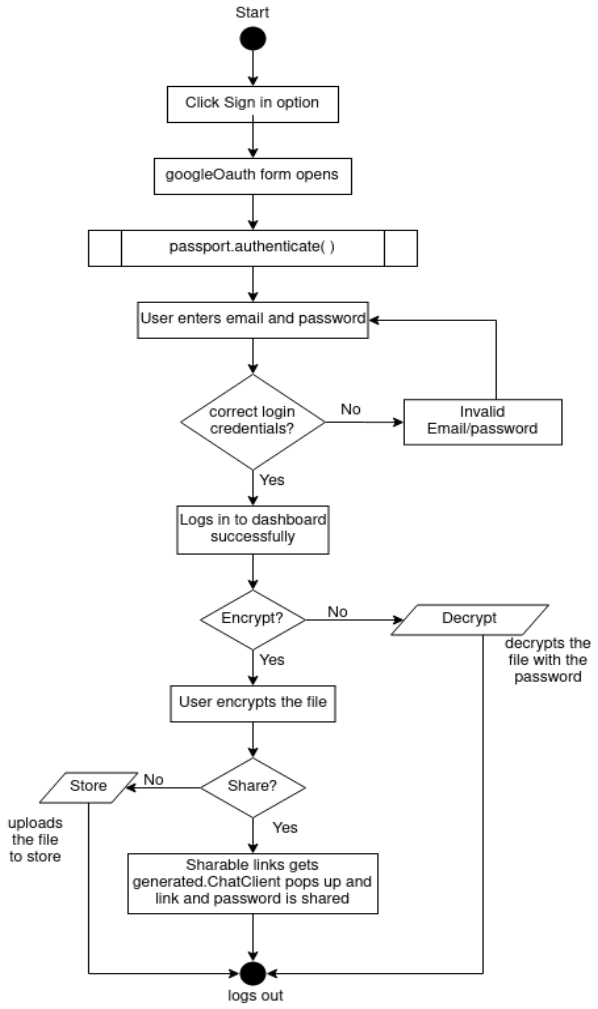

Fig. 2 Activity diagram for the proposed system

VI. METHODOLOGIES

A. Secure Login

The user must be authenticated before the whole process occurs. Passport Oauth2.0 will provide a secure login to the application. By clicking on the Log in with Google button, the user will be redirected to google's login form where the user must enter a valid email id and password to be authenticated and use the application. If the user provides an invalid credential then the login will fail and the end-user won't be able to log in.

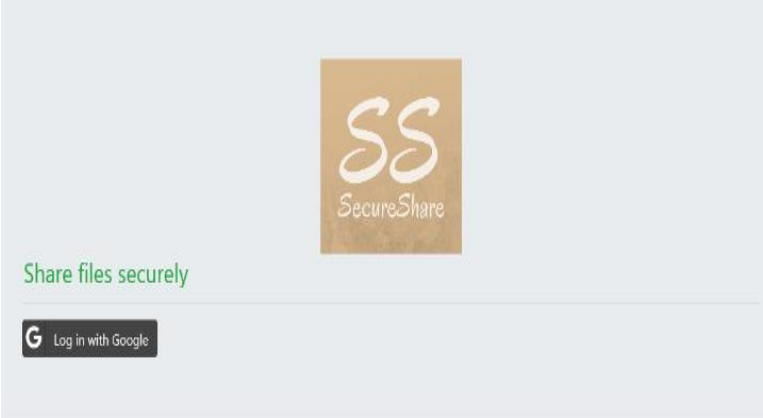

Fig. 3 Secure login

B. Profile Creation

The user's profile is created once the user is securely authenticated. Once the user's credential is validated the user data is stored in our MongoDB database. As per the schema written, we store the image URL and the user's image, and the user name. The profile image and username are fetched from the data stored in the MongoDB database. Each user profile gets a unique id which is used to authenticate them to their own dashboard. The dashboard enables a user to access the user's stored files and the encryption tool.

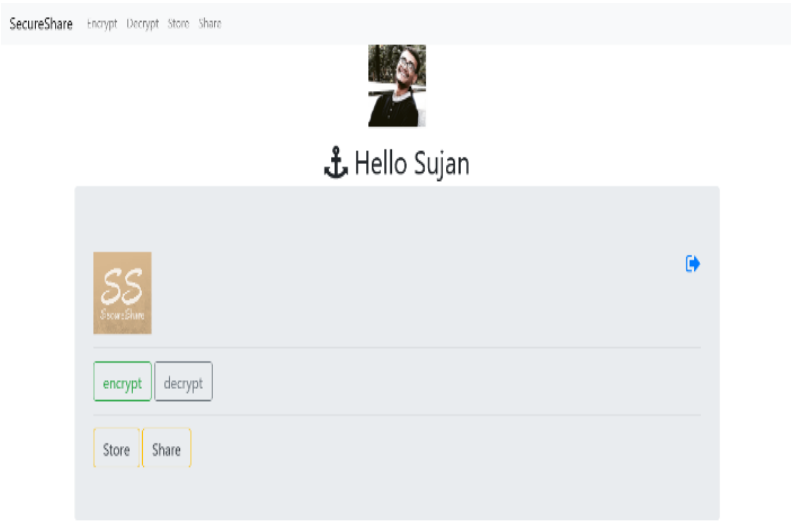

Fig. 4 Profile creation

C. File Encryption

The encryption tool can be accessed via the encrypt button. The user gets redirected to the tool once the button is clicked. The Encryption process mainly involves three steps:
Step 1: Select a file to encrypt
Step 2: Enter the password or Copy from the password generator.
Step 3: Download the encrypted file

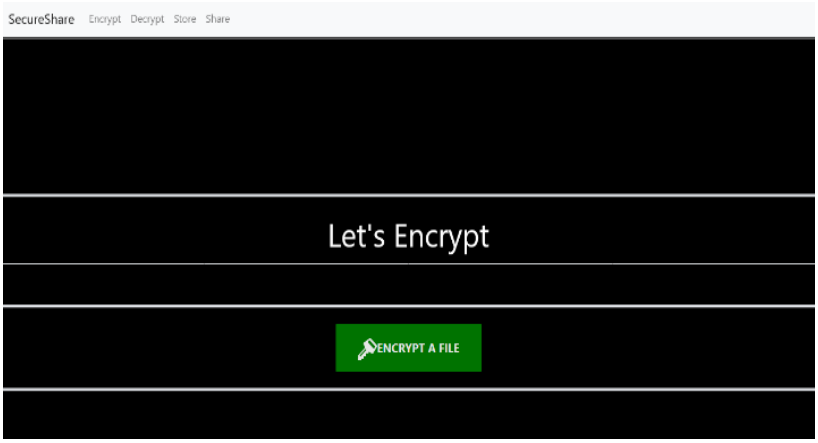

Fig. 5 File encryption

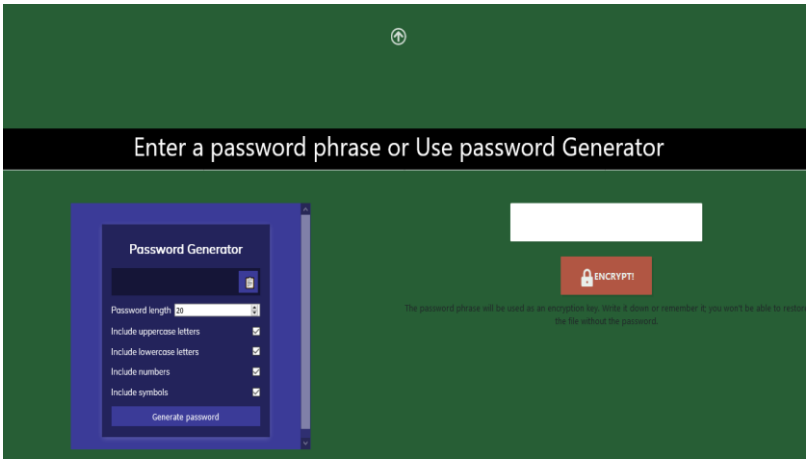

Fig. 6 Password generation or input

D. Notification Form

The form is used to send the chat room detail or the file link to the end-user. The form is integrated with the Gmail client. The user can enter the mail of another user and add a subject. The content can be added below the subject. The send email button lets the user send the mail once clicked.

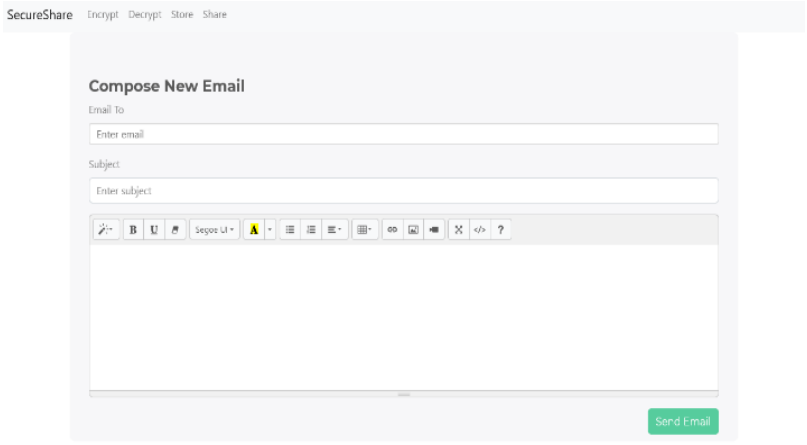

Fig. 7 Notification form

E. Chat Client

The chat client is the implementation of socket programming which enables users to send and receive data via the web sockets. The chat client is used by the users to securely transmit the encrypted password and the file download link. The chat data is cleared once the user leaves the room. A user can create a room and share the room id with another user to join the chat. The user can share each other's locations as well with the location button.

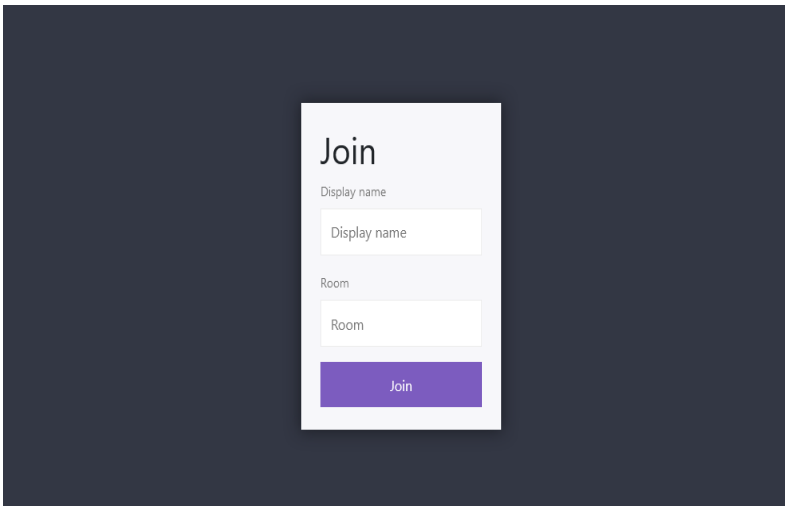

Fig. 8 Chat client homepage

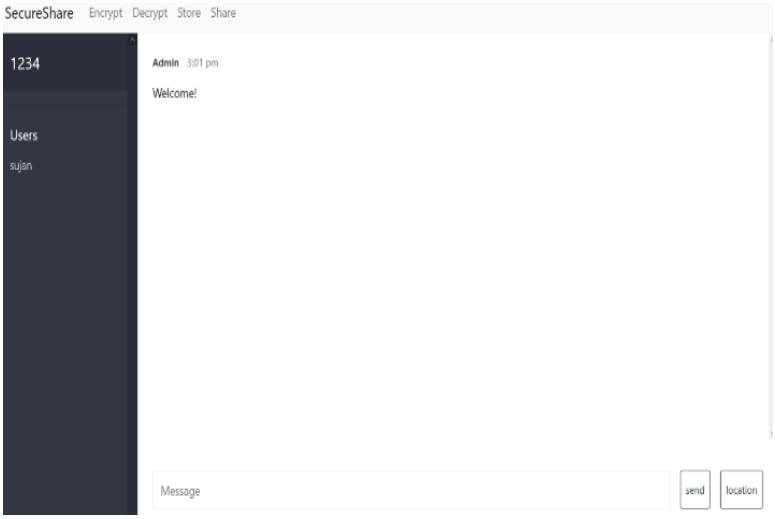

Fig. 9 Chat client chat box

F. File Store

The user will upload the file which can be accessible later. The user can select a file and submit it to upload the file for further use. Once the user uploads the file the file is available in this section. Users can download and use the file later. The user can also delete the file if not required.

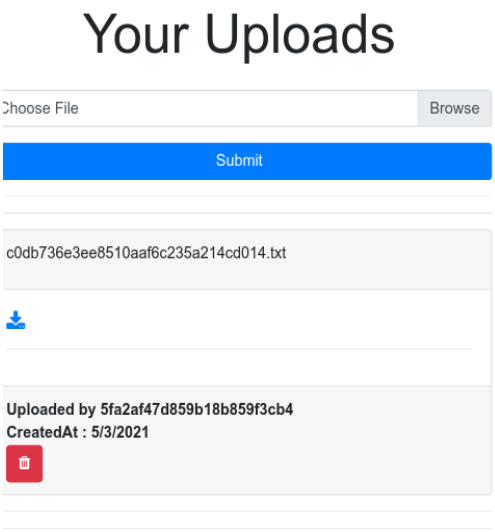

Fig. 10 File storage

G. File Share

The user will upload the encrypted file such that the downloadable link will be generated. The generated link can be used to download the file. The same link is shared via any of the sharing tools. The links get expired after 24 hours and the user has to reupload and reshare the link in case the link expires.

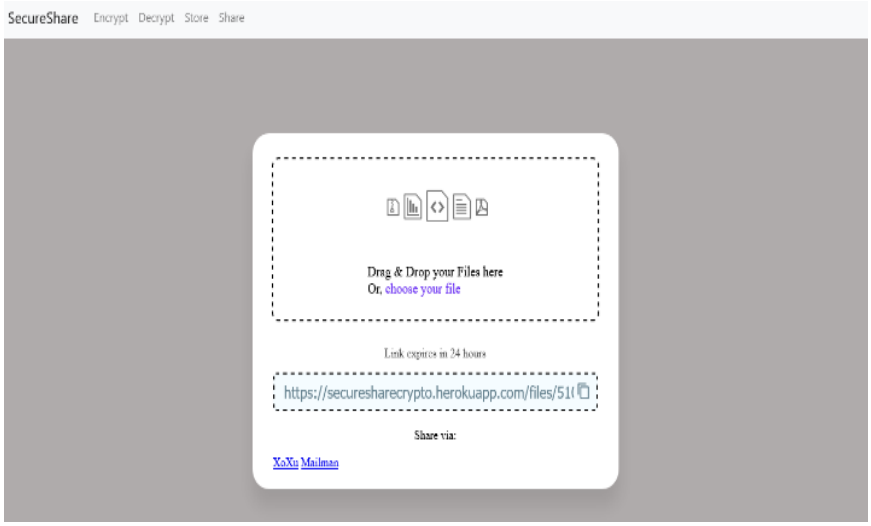

Fig. 11 Link generation for file sharing

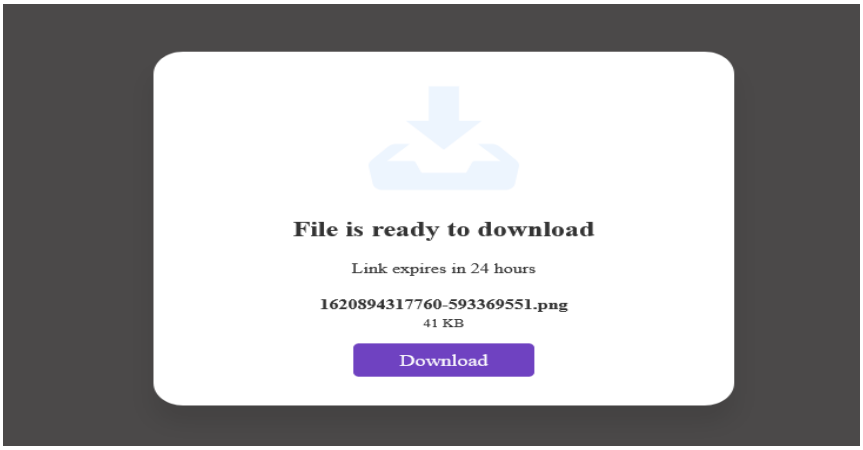

Fig. 12 File download

H. File Decryption

The encryption tool can be accessed via the decrypt button. The user gets redirected to the tool once the button is clicked. The Encryption process mainly involves three steps:
Step 1: Select a file to decrypt
Step 2: Enter the password
Step 3: Download the decrypted file

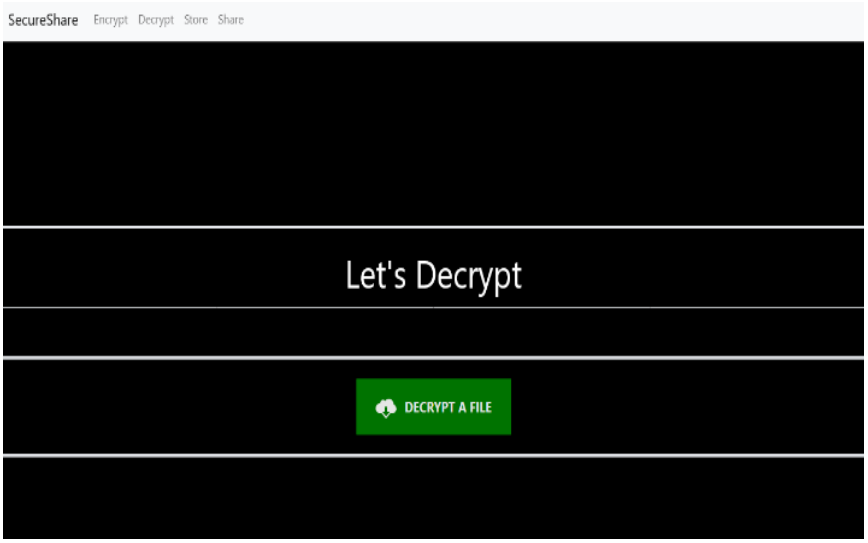

Fig. 13 File decryption

VII. RESULTS AND DISCUSSIONS

In this study the proposed web application was built to provide an encryption/decryption tool and securely share and store the files using cryptographic modules and packages. From the practical implementation we got the following results:

TABLE 1

ENCRYPTED AND DECRYPTED FILE SIZE IN BYTES

File Name	Original File size	Encrypted File size	Decrypted File size
graphql.png	63,470	1,12,896	63,470
assignment.pdf	15,68,561	27,88,632	15,68,561
psiphon3.exe	7,402,376	14,506,378	7,402,376

Table 1, it shows that there is a difference in encrypted and decrypted file size. This happens because the blocking length in the header cipher text file is caused by both the encryption process and the use of the extension .encrypted. But the decrypted file does not change the file contents and maintains the original size. Also the tool can encrypt various file types without any error or hindrance.

TABLE 2

ENCRYPTED AND DECRYPTED TIME IN SECONDS

File Name	Encryption time	Decryption time
graphql.png	0.02	0.02
assignment.pdf	0.03	0.03
psiphon3.exe	0.02	0.02

Table 2, shows the time taken for encryption and decryption of the files in our application. The Encryption and Decryption tool takes almost similar time to encrypt and decrypt the file, although it may vary with size of the file.

VIII. CONCLUSIONS

With the growing demands on the cloud storage platforms, simplifying the Interface and solidifying the underlying security is the main concern as the sharing of an individual's data is not as private as one assumes. The vulnerability of cloud storage and lack of security results in the loss of millions of data. Thus, in order to enhance the process of security for the storage and sharing of data we've created this web application which uses a cryptographic approach to secure the user data. We can also improve our proposed algorithm for encrypting and decrypting larger files. We can add push notifications and shareable links that'll expire after the stipulated time. Additionally, we can add other authentications like GitHub and Twitter.

ACKNOWLEDGEMENT

This research is supported by Jain University, Faculty of Engineering and Technology. We are thankful to the Professors of our department who assisted this research. We express our appreciation towards our Guide, Madhumala RB, Assistant Professor, Dept of ISE, FET-JU, for providing her guidance and support during this research. We are also thankful for her comments and suggestions on the earlier versions of the manuscript, although any existing errors are our own and should not stain the reputations of respectable professionals.

REFERENCES

[1]. Madhu Sharma, Dr. Ashish Sharma, A secret file sharing scheme with chaos-based encryption. Institute of Electrical and Electronics Engineers.2019.
[2]. Songling FU, Xiangke LIAO, Lianyue HE, Chenlin HUANG, Xiaodong TANG, Si ZHENG. Efficient and Fine-Grained Sharing of Encrypted Files .Institute of Electrical and Electronics Engineers.2010.
[3]. Zhang Jian-hua and Zhang Nan. Cloud Computing-based Data Storage and Disaster Recovery. Institute of Electrical and Electronics Engineers.2011.
[4]. ISJl Karabey, Gamze Akman. A Cryptographic Approach for Secure Client-Server Chat Application using Public Key Infrastructure. Institute of Electrical and Electronics Engineers.2016.
[5]. R.Nivedhaa and J.Jean Justus. A Secure Erasure Cloud Storage system using Advanced Encryption Standard algorithm and Proxy Re-encryption. Institute of Electrical and Electronics Engineers.2018.
[6]. Qiwu zou, Yuzhe Tang, Kai Li, Charles A.Kamhoua, Kevin Kwiat, Laurent Njilla. ChainFS: Blockchain-Secured Cloud Storage. Institute of Electrical and Electronics Engineers.2018.
[7]. Yongmei WEI, Yong Wee Foo. A cost-effective and reliable cloud storage. Institute of Electrical and Electronics Engineers.2014.

[8]. S V V Satya Surya Sravan Kumar Mangi, Saddam Hussian.SK, Leelavathy.N. An Approach for Sending a Confidential Message to the Restricted Users in Defence Based Organization. Institute of Electrical and Electronics Engineers.2019.

[9]. V. Valli Kumari, D.V.NagaRaju, K.Soumya, KVSVN Raju. Secure Group key Distribution Using Hybrid Cryptosystem .Institute of Electrical and Electronics Engineers.2010.

[10]. Bharati Mishra and Debsish Jena. CCA Secure Proxy Re-Encryption Scheme for Secure Sharing of Files through Cloud Storage. Institute of Electrical and Electronics Engineers.2018.

[11]. Priyanka P.Kendrekar and M.K Chavan. Cryptographic Implementation of Aggregate-Key Encryption for Data Sharing in Cloud Storage. Institute of Electrical and Electronics Engineers.2016.

[12]. Swatee Pachaghare and Prof. Pragti Patil. Improving Authentication and Data Sharing Capabilities of Cloud using a Fusion of Kerberos and TTL-based Group Sharing. Institute of Electrical and Electronics Engineers.2020.

