Secure-e-Share: Data leakage Detection and Prevention with Secured Cloud Storage.

Aditya Jaiswal
Department of Computer Science
Shah And Anchor Kutchhi Engineering College
Mumbai, India
aditya.jaiswal15974@sakec.ac.in

Vansh Purohit
Department of computer Science
Shah And Anchor Kutchhi Engineering College
Mumbai, India
vansh.purohit15866@sakec.ac.in

Vivek Jhawar
Department of Computer Science
Shah And Anchor Kutchhi Engineering College
Mumbai, India
vivek.jhawar16061@skec.ac.in

Yash Jadhav
Department of Computer Science
Shah And Anchor Kutchhi Engineering College
Mumbai, India
yash.jadhav15574@sakec.ac.in

Prof. Karuna Borhade
Department of Computer Science
Shah And Anchor Kutchhi Engineering College
Mumbai, India
karuna.borhade@sakec.ac.in

***Abstract* – Data privacy and security has always been a prime concern of any user or an organization, especially when it comes to sharing of confidential files either for different end users or in any organization. Secure-e-Share is one such file-sharing system that creates a secure environment for sharing files of different formats (i.e., zip, Docx, txt, CSV, pdf, etc.). The key protection point of this system is that it encrypts the file and then stores it in the cloud storage which makes the database secure, as only encrypted files are stored. The decryption of a file is done only when the authorized individual accesses it with a unique key provided by the sender. For detection, after every successful transfer, the counter of successful transfer is increased which shows the success rate of the web app, and every case of data leakage is also prevented and detected and recorded creating a threat-free environment.**

Keywords* – *Python, MongoDB-Atlas, HTML, CSS, JavaScript, GitHub.

I. Introduction

Data leakage has been a very serious issue posing a big threat to people's privacy. Moreover, when this threat is coupled with data, impersonation, and identity theft it becomes a daunting problem for casual users as well as big organizations resulting in facing a huge loss. In this digital age, everything is electronic from sending emails, conducting important meetings online, and sharing crucial information and confidential files. While doing so we seldom think of our own and our data's privacy.

Secure-e-Share creates a safe environment for sharing confidential files with a two-layer safety:

1) Encrypted data in cloud storage
2) Decryption only by a unique private key.

Meaning, the data sent by the user is firstly encrypted and then stored in the cloud storage. At the receiver's end, they can access the file only by a unique private key that the file owner provides. This private key is something that helps in the decryption of the file resulting in making the file accessible to the authorized person. By this method, it is made sure that the file is received at the correct destination and that the shared data or files are safe and secure.

II. Literature survey and Technology stack

A. *Literature Survey*

To get a clear idea of how the existing systems work and how our system solves the issue faced by similar existing web apps; Secure-e-Share was developed considering that it stands out with its uniqueness and specifications. For this purpose, similar applications and various algorithms were thoroughly studied. Also, many existing Research papers were taken into account for better knowledge and to get more relevant information on our subject. The first research paper reviewed was MEGA. MEGA is a very old and reliable file-sharing service provider which makes it a perfect review subject. Research by cryptographers in Zurich, Switzerland revealed a severe vulnerability in their system. MEGA software is known for high confidentiality right from the time of establishment it was so secure that it soon became a home for piracy after becoming popular. However, after recent studies, the researchers found a way to hack their encryption system. This security flaw revolves around MEGA's RSA encryption. Through this flaw, an attacker could access the user's private key stored in the servers which were owned by MEGA in an encrypted format. With each attack, the hacker could boil down the possibility of a certain key being the user's private key, to be precise – 512 attempts. In other words, after running their attack 512 times, the attacker could freely access the user's crucial information through their account, completely exploiting their privacy.

Secure-e-Share efficiently solves this issue, wherein it shares and encrypts a file on the go, which means that it encrypts the uploaded file with a unique key. When a user attempts to share a file, the receiver is required to ask for the unique key, as long as the receiver has not asked for the key and the corresponding permission is not granted, the file must not be decrypted by any means. This secure encryption system forms the cornerstone of our idea.

979-8-3503-9874-8/23/$31.00 ©2023 IEEE

The research paper provides information about data leakage detection as they have considered a business scenario where a person shared information where the data leakage is detected which harms the users' privacy. The paper suggests some techniques for prevention such as data allocation strategies, fake records, and leakage models. The paper also talked about leakage of the confidential data at more than one source and the similarity of data showing proof of data leaked. [2] Focuses mainly on database security and the threats to the database and the possibility of database security risks and many issues concerning the safety of the database. It talks about the growing threats on the repositories by different techniques by hackers resulting in data theft and unauthorized access. It suggests some ideas on database security by looking at the vulnerability of the databases. Suggestions like packet sniffing, encryption of data, and firewalls were given based on the issues. [3] This paper states data leakage prevention using a timestamp and suggests an impactful solution for the problem of data leakage and suggests how it can be reduced. The main ideology behind this was using a timestamp mechanism that controls the data usage of that data and makes it unavailable after a particular period; this results in the reduction of data as after a certain time stamp the data will not be leaked as it is not available. [4]. This paper focuses on security in cloud storage and fetching of data safely by only the authorized person and suggestions for the same.

After taking many notes from all the research papers and similar existing software and their vulnerabilities we came up with the final idea of Secure-e-Share. The secure-e-Share name itself represents “Secure Encrypted Sharing” of information where we count the successful data transfer and data leakage case and create a fully encrypted system for this we used our encryption and decryption algorithm which works on the ideology of distorting the file with a key value which is random and unknown to anyone and encrypts the file and makes it inaccessible.

B. Technology stack

While developing a web app we came up with the main programming language as python and used a framework as Flask which provides simple and easy integration and synchronization while acting as an effective interface between the front and backend. For the backend database, we used MongoDB-Atlas as cloud storage and used the functionality of GridFS provided by MongoDB for storing files. For frontend UI and specification of responsiveness to the website, we used HTML, CSS, and JavaScript. We use GitHub for version control.

Also “Integrated Development Environment (IDE)” was used, which is “VsCode”. Vscode is a compiler that is easy to use and supports most languages. Its interface is attractive and divided into sections which makes it user-friendly. It has various extensions which make the developers’ work easy.

Fig. 1-Python and Mongo

III. METHODOLOGY

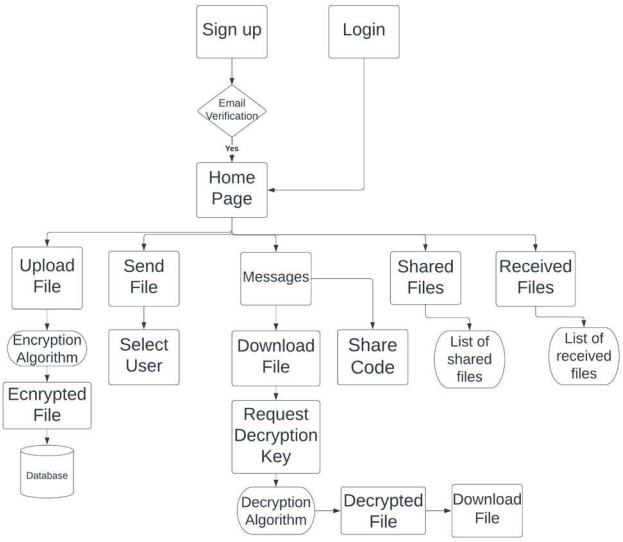

Fig. A Activity flow diagram of Secure-e-Share

While browsing through various existing file-sharing prototypes, it was observed that there is some sort of flaw in any system that was picked. While almost every system provided users with an appealing user interface and promising security, one could still find some way to gather loopholes in the system. Secure-E-Share is a project that attempts to avoid most of the flaws.

Simplicity is the key in a service like this however, oversimplification just simplifies the task of an attacker trying to cheat the system. At the same time adding complex features just kills the purpose of a plain - secure - safe file-sharing system. Providing exactly what the user needs is the best way to ensure a smooth development flow. Speaking about users and their needs, we first need to define “the target audience”, that is the type of users expected to use this system. It can be anyone, friends trying to share their memories with each other, colleagues, or large enterprises sharing extremely confidential data. The file-sharing system ensures security and simplicity, it provides nothing less than what is needed to conveniently share data and nothing more.

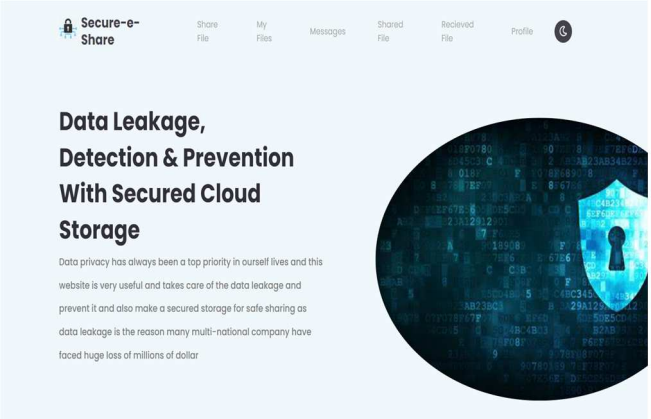

Fig. 2.1 Landing Page Light Theme

Fig. 2.2- Landing Page Dark Theme

Hence while keeping these points in mind we proceeded with creating Secure-E-Share. Fig 2.1 and Fig 2.2 shows the landing page of the website in both light and dark themes respectively. While keeping the interface simple, it speaks about everything the system provides. The first option, which is also one of the core features of the website, enables users to upload a file from their local storage to the Secure-e-Share file server storage. In Fig 3 the files are first encrypted and then moved to the user's account.

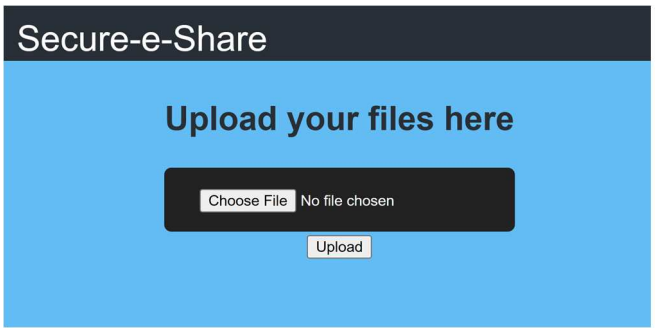

Fig. 3-Upload file from the user's system

These stored files can be easily viewed by clicking the My Files button on the navigation bar. As shown in Fig. 4 this section allows the users to view a list of their files and share the same with other users. Each file is linked with a button "Send file" that takes the user to a new page, where they can select whom to share the file with as shown in Fig. 5. Once selected a link to download the same is provided to the receiver. Fig.4. and Fig.5. shows the sender's end.

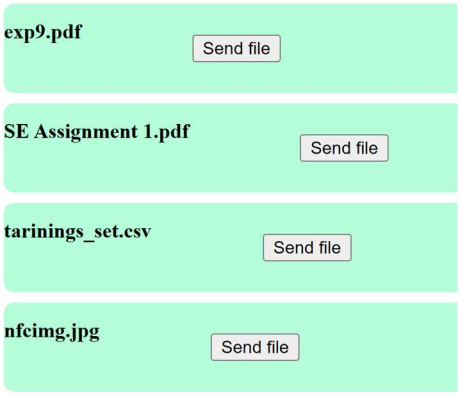

Fig. 4-Uploaded files in Secure-e-Share file server storage

Fig. 5- List of Secure-e-Share user's with search engine

Here is where Secure-e-Share stands out from a lot of other files sharing systems, instead of allowing the receiver to straight away download the files, they are required to ask for 'permission' to open it from the sender as shown in Fig. 6. This does seem like a redundant task but it provides a very useful workaround to solve the data leakage issue. The permission is such that the sender receives a notification and has a choice here to send a verification code which enables the receiver to unlock the document as shown in Fig. 7. This code is stored securely at the time of uploading the document in an encrypted form and updated once the file is successfully shared. The below figure shows the receiver's end.

Fig. 6-Receiver receives the file but need to grant for key

The Messages section is where all the granting and denying of permission takes place, every notification is sent to the users through messages section as shown in Fig. 8. This simple structure helps the users to keep their account clean and organized. Fig.7. displays the sender's end and the Fig.8. displays the receiver's end.

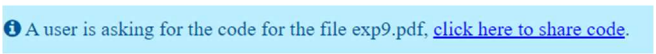

Fig. 7-Sender get an notification from receiver requesting for unique key

Fig. 8-Receiver gets the key of the file

Now the question arises that how actually the data leakage cases are detected? What if the user by any other means gets the code without even requesting it? This case was handled by using a flag variable which is initially set to 1 representing that the sender has not yet granted the key value so if the receiver does not request for key and directly attempts to access the file, firstly the encrypted file will be received as he has not requested for the key and secondly the data leakage case is detected and the counter for it will increase. This will prevent the data leakage from users and detect it and if the database is hacked by the user he will be only left with a chunk of encrypted file which will only decrypt with the random, unknown unique key and for every file there will be a different key so hacking becomes pretty much impossible.

Next sections are for data gathering and analysis purposes, Shared and Received File sections do exactly what you can guess by their name. The Shared File section will show the files shared by the user to others, while the Received File section will let one see the files they received from others as shown in Fig. 9.1 and 9.2. This helps them keep a track of all the transactions taking place through an account, this feature can prove to be extremely helpful at the enterprise level, where organizations can use organization accounts to check the flow of their data should there be any discrepancies.

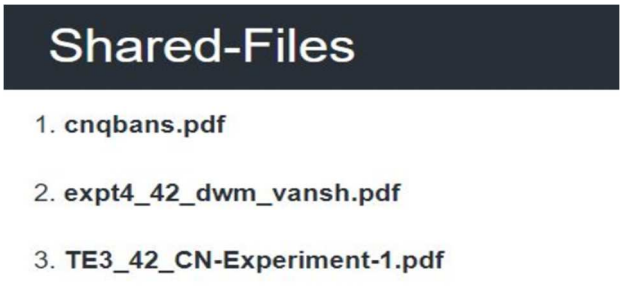

Fig. 9.1-List of shared files

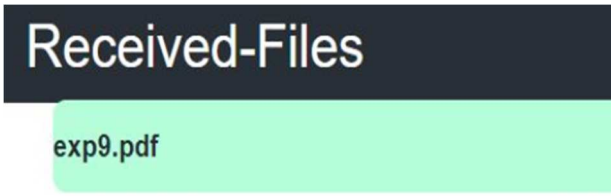

Fig. 9.2-List of received files

Fig. 10 User's Profile page

Lastly a profile section where the user can check their current saved credentials and logout from the system as shown in Fig. 10. At the bottom of the home page we provide an analysis of successful transfers and data leaks. But how do we determine that a file has been successfully transferred or someone tried to sneak into the system? This is handled by the security provided through keys. A successful transfer is one where a key is sent from the sender side and the file is unlocked using it. But what if the receiver somehow just finds the code? If the code is granted from the sender's side, then and only then we allow the receiver to unlock or more precisely decrypt the file. An attempt made by entering the correct key without asking for permission will just give the receiver an encrypted file, thereby maintaining data security. Fig. 11.1 shows the counter status before a successful transfer and Fig. 11.2 shows counter status after a successful transfer which is incremented by 1.

Fig. 11.1-Before successful transfer

Fig. 11.2-After successful transfer

IV. Results & Future Scope

The result analysis is a very crucial part in any project or research as it gives the output of the whole work and one can determine the success ratio of the project or research. The project contains a counter for both successful transfer of file and data leakage which shows the accuracy of the project and helps us judge the efficiency and reliability of the project. For result analysis the project has been tested with over 50+ cases and has an accuracy score of 100 percent and gives the desired output and is tested with all the possible test cases. The project has also been tested with all the data leakage cases and that too gives an accuracy score of 100

percent.

Fig. 12 shows the live score of the counters.

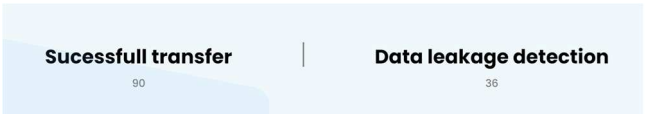

Fig. 12- Live Counter

Comparison of Secure-e-Share with existing systems :

	DATA LEAKAGE COUNTER	ENCRYPTED CLOUD STORAGE	PRIVATE KEY	FILE SHARING	2 LAYER ENCRYPTION SECURITY
Secure-e-Share	✓	✓	✓	✓	✓
Mediafire	X	✓	X	✓	X
Mega	X	✓	X	✓	X
DropBox	X	✓	X	✓	X

For future scope working on an enhanced feature for cloud security which sends a virus to the unauthorized person's system if they try to hack the database as the virus will not even allow them to access the encrypted file making the system more safer and reliable, also adding firewalls to the system.

V. CONCLUSION

To conclude, it can be stated that data is successfully detected as well as prevented from leaking. In addition, cloud storage is also safe since all the data stored in it is encrypted. Even if there is the slightest chance of anyone intruding in the cloud storage, it will be difficult to retrieve any data since it all will be encrypted. Data should always be safe and secure, strongly believing in this motto we have made this Secure-e-Share which ticks all the checklist of requirements to keep data secure. Taking the future scope into consideration additional features like payloads and firewall will ensure the utmost safety of the data in the cloud storage.

VI. REFERENCES

[1] Papadimitriou, P., & Garcia-Molina, H. (2011). Data Leakage Detection. IEEE Transactions on Knowledge and Data Engineering, 23(1), 51–63. doi:10.1109/tkde.2010.100.

[2] Mousa, A., Karabatak, M., & Mustafa, T. (2020). *Database Security Threats and Challenges. 2020 8th International Symposium on Digital Forensics and Security (ISDFS).* doi:10.1109/isdfs49300.2020.91164363]

[3] Peneti, S., & Rani, B. P. (2016). *Data leakage prevention system with timestamp. 2016 International* Conference *on Information Communication and Embedded Systems (ICICES).* doi:10.1109/icices.2016.7518934

[4] Zhang, X., Du, H., Chen, J., Lin, Y., & Zeng, L. (2011). *Ensure Data Security in Cloud Storage. 2011 International Conference on Network Computing and Information Security.* doi:10.1109/ ncis.2011.64