International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 04 Issue: 06 | June -2020 ISSN: 2582-3930

Secure File Sharing System Using Java

Ronak Karani[1], Tejas Choudhari[2], Anindita Bhajan[3] , Madhu Nashipudimath[4]

[1,2,3]Student, Pillai College of Engineering, New Panvel, Maharashtra – 410206, India
[4]Assistant Professor, Pillai College of Engineering, New Panvel, Maharashtra – 410206, India

Abstract - Application to store files securely on an untrusted server and give access to other users is essential. Due to the growing usage of mobile devices and advances in networking technologies a lot of files and data are generated and getting stored on different servers. Storing confidential data over a network has various security risks. To prevent any such attacks data stored on the server is encrypted. Cryptography is a popular modern technique for encryption. The problem with this is it requires trust on the server to encrypt the data. The creator of data has no control over encryption. Also, there isn't a mechanism provided to give access to other users securely. The current mechanism encrypts the file on a server and decrypts the file on the server before giving access to other users. Solutions currently used only encrypt the file thus only ensuring confidentiality but for a system to be secure, there should be mechanisms to ensure integrity and authentication. Thus a system is required which eliminates the need for a trusted server and ensures integrity and authentication. We have developed an application where users can share files without the need for the server to be secure and taking into consideration authentication and integrity.

***Key Words*:** Cryptography, Confidentiality, Integrity, Authentication, RSA, Digital signature, Storage and Security.

1.INTRODUCTION

The owner has a file locally on his machine which he uploads on the server and gives access to other users. Sharing passwords can be done to share files and relying on file servers for security. It is not the most appropriate way of relying on other entities for the security of the files. The owner can encrypt the file and store it on the server. The token for a file can be generated and used to share files with other users. The shared user accesses the file by checking its integrity and authenticating the owner. This way the system is more secure and the owner has control over the security. Integrity is ensuring that unauthorized users can not make any modifications. Authentication is about making sure that data was shared by the actual owner and not any attacker impersonating the owner. Hashing is a popular scheme for ensuring integrity. Digital signatures are used for authenticating the owner. The basic idea of a digital signature is that the sender generates the digital signature for data using its private key and sends the signature along with the file to the receiver. The receiver verifies the signature using the public key of the sender.

To implement such system cryptography can be used. Cryptography is the science and art of transforming messages to make them secure and immune to attacks. Prior to modern-day cryptography data was manipulated in a particular pattern that can easily be identified. Modern-day cryptography is heavily based on mathematics and computation theory. Algorithms are designed considering computational hardness. To encrypt large data symmetric key cryptography can be used. Symmetric-key cryptography is based on substitution and permutation of symbols. Symmetric key cryptography uses a single key to encrypt and decrypt the data. A single secret key is shared between both the sender and the receiver. The major disadvantage is that if n people are present in a system then n*(n-1)/2 total keys would be required. Symmetric key cryptography is based on sharing secrecy. AES is the latest and most widely used symmetric-key cryptography algorithm. AES block size is 128 bits and the number of rounds is dependent on the key size. The key size is generally 16,24, 32 bytes long. A particular round process in AES consists of following operations- substitution bytes, shifting rows, mix columns, add round keys. Asymmetric key cryptography is another cryptography technique. Asymmetric-key cryptography is based on applying mathematical functions to numbers. The function used should be a trapdoor one-way function. Asymmetric key cryptography consists of two keys- a public key and a private key. The basic idea is that the sender encrypts the message using a public key and then the receiver decrypts using his own private key. A system of n people would require only n personal secrets. Asymmetric key cryptography is based on personal secrecy. RSA is the most popular public-key cryptosystem. RSA cryptosystem uses two exponents e and d. e is the public key and d is the private key. Sender to create ciphertext C= Pemod n. Receiver to retrieve plaintext P= Cd mod n.

2.LITERATURE SURVEY

A literature review is an objective, critical summary of published research literature relevant to a topic under consideration for research. Ten published articles have been referred to in order to create a firm base about the project. Following is a brief overview of all the ten papers that have been referred:

How files can be securely stored on the cloud and also discusses the problem with using a single algorithm to encrypt the file and how ineffective it will be on the cloud. The method discussed in paper splits the file into blocks and each block is encrypted using AES, BRA, blowfish, RC6 algorithms. The key information and information about which file uses which algorithm is sent to the receiver using Steganography.[1]

The integrity of files and restoring the files if integrity is violated. The proposed system uses a pattern of each protected file to determine its modification. Methods used for pattern generation are cryptographic hash functions. The system uses a database that stores the names of files that need to be protected and their hash codes. To check the integrity of the file the hash code of the file is produced and checked with one in the database. If the file is verified positively then access is granted otherwise the administrator is alerted and if a saved copy is available of the same file then the file is restored.[2]

The integrity of files and restoring the files if integrity is violated. The proposed system uses the pattern of each

protected file to determine its modification. Methods used for pattern generation are cryptographic hash functions. The system uses a database which stores the names of files that need to be protected and their hash codes. To check the integrity of the file the hash code of the file is produced and checked with one in the database. If the file is verified positively then access is granted otherwise the administrator is alerted and if a saved copy is available of the same file then the file is restored.[3]

Providing the facility to securely store and share the data in a specific group using the cloud for storage. The method proposed in the paper uses group signature and encryption techniques. The advantages of the method proposed is that data owners can store the file without revealing their identity to others in the cloud.[4]

The PKI has the disadvantage that the mathematical relation between public and private keys is maintained. Paper proposes a new PKI scheme with addressable elements (PKA). The approach proposed removes the mathematical relation between public and private keys using addressable cryptographic tables.[5]

The secret key can be shared with other users to whom access needs to be given. The problem with using a single key to encrypt all data and using different keys for different files. The solution described in the paper tries to address both the problem using key aggregation. In key aggregation, different data files are encrypted with different keys and then for decryption, a single aggregated key is used. The encryption algorithm used is AES and the system is being implemented in java using the Keystore data structure.[6]

Security tradeoff improves the performance of data transmission and increases security. Also, MD5 hashes are no longer considered cryptography secure.[7]

Security mechanisms will prevent confidential data from being misused making the system more reliable. High speed: The proposed method will make encryption and decryption with the proper key much faster than usual. In RSA, there is a high chance of guessing the combination until and unless the exponent size is made higher than 2048 bits.[8]

The proposed algorithm is a Multilevel Encryption and Decryption algorithm. Thus, in our proposed work, only the authorized user can access the data. Even if some intruder gets the data, he must have to decrypt the data at each level which is a very difficult task without a valid key. It is time-consuming as multiple encryption and decryption take place.[9]

In this paper, The system satisfies confidentiality, integrity, and authentication. It provides access control. The confidentiality of data is totally dependent on a trusted crypto server.[10]

3. METHODOLOGY

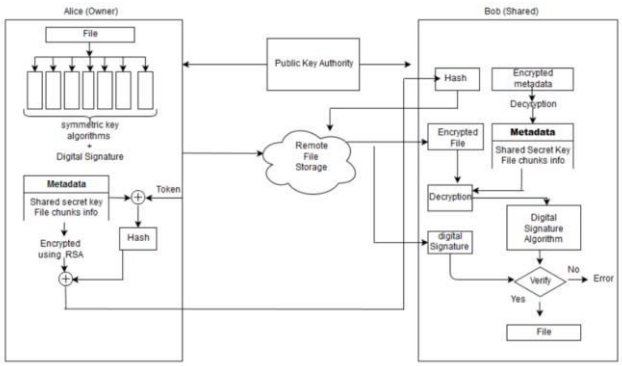

Fig 3.1 Block Diagram

In our proposed system there are four blocks each having different functionality. Two blocks are implemented on the client machine and the other two blocks on two different servers.

- The file is divided into chunks and then every chunk is encrypted using the AES algorithm and a digital signature for the file is generated. A metadata file is created consisting of secret keys and information about file chunks.
- On the server, files are stored and a table is maintained to map hash codes with file names.
- A different server is maintained as a trusted center for the distribution of the public keys.
- Lastly, there is a block for downloading the file. The file downloaded is decrypted then it's digital signature is verified before showing the file to the user.

The system is implemented using a Java programming language. The java programming language is chosen due to its wide range of libraries and data structures available to make it easier to implement cryptographic algorithms and also due to platform independence, robustness and abstraction features of java. The system is based on a client-server architecture. A file-sharing application is developed with support for chatting. Java Swings is used for the development of a user interface. Web socket technology is used to develop real-time file sharing between users. The UI has two major interfaces, one for login and another file-sharing app. The backend system has two major servers - file server and chat server. The file server is used to securely store files and make files available for download when requested. The chat server is the server for real-time communication between connected users. The user can log in using their username and password. The whole real-time communication is carried out on the basis of a few events. Connect event- whenever a new user comes online connect event is fired. Disconnect event- whenever a user goes offline disconnect event is fired. Chat event- when a user wishes to send a message to another user a chat event is fired. File event- when a user wishes to securely store a file or share a file the file event is fired. When a file is shared the file gets saved on the file server and both the sender and receiver's UI get a download button. On clicking the download button the file gets downloaded on user’s machines. Before uploading the file, the file gets divided into chunks and each chunk gets encrypted and digital signature is produced and then uploaded to the server along with digital signatures. During downloading the file gets decrypted and its signature gets verified.

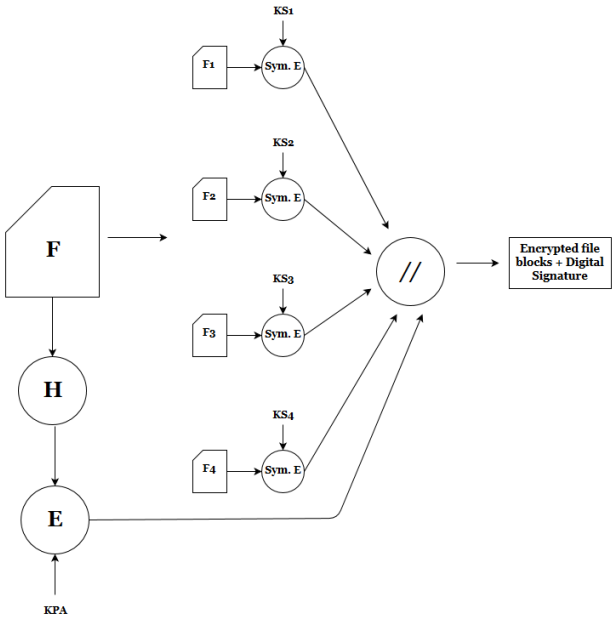

Fig 3.2. Encryption process
Notation:
F: File
H: Hash Function
E: Asymmetric Encryption
KPA: Private key of Alice
F1, F2, F3, F4: File chunks
Sym. E: Symmetric Encryption
KS1, KS2, KS3. KS4: Symmetric secret key
//: Append all files together

1. *Client*

The client program will be running on a user machine. The main job of the client program is to upload the file and second to download the file with shared access. To upload the file following steps are followed- the file is divided into chunks and the information regarding chunks is saved in a separate file on the user's machine. Every chunk is then encrypted using the AES algorithm and secret keys are saved in the same file containing file chunk information. The file's unique signatures are generated using the RSA Digital Signature Algorithm which requires the private key of the owner. The encrypted file chunks and the digital signature is sent to the server using socket programming. When the user shares the file the metadata content stored in the file which was created on the user machine while encrypting the file is encrypted using the RSA algorithm which will require the public key of the receiver. Encrypted metadata is sent to the user with whom the owner wants to share access.

When a user wishes to view the shared files the client program sends a request to the server with the filename. The server processes the request and sends the encrypted file stored on the server along with the digital signature to the user. The client program on the user machine decrypts the file using the keys that were shared by the owner in metadata information shared. Then the file is then given to a digital signature algorithm. The digital signature generated is matched with the downloaded digital signature. If both of them match then the file will be shown to the user else the integrity of the file might have been violated which is informed to the user.

2. Server

The file server program saves the file on the server and identifies user requests and accordingly sends the file back. The file server has two programs listening on ports-3333 and 3332. The port 3333 is used to send files to the requested user. The port 3332 is used to save files on the server. Chat servers use WebSockets to enable real-time communication between clients. As soon as there is a change on the server it is notified to all the users connected.

4. RESULT

Fig 4.1. Login screen of an application

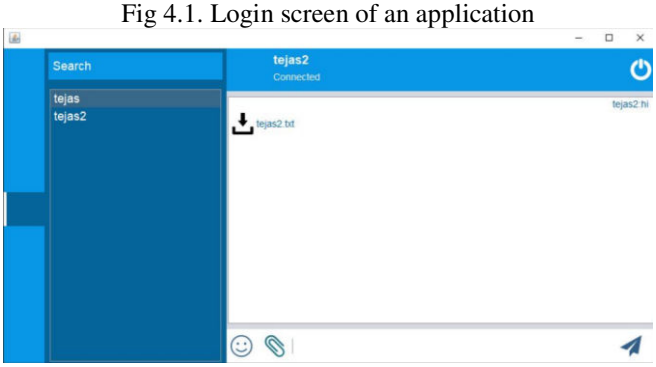

Fig 4.2 Owner of file

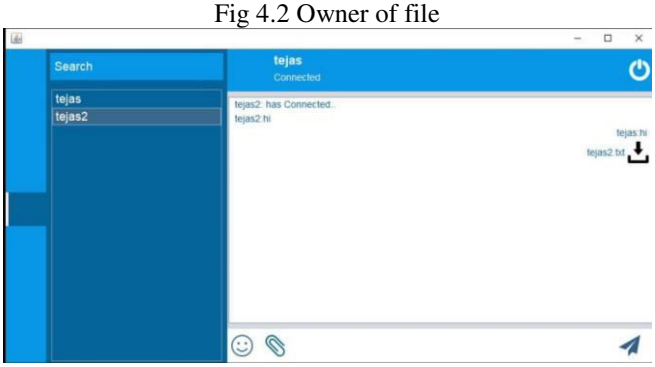

Fig 4.3 Shared File

The solution developed was analyzed for file size and time taken to upload a file to the server and download it from the server.

The time we're almost similar differences was so less that it can not be distinguished. Also, the time for file upload will depend on a variety of factors like processing power of client machine, network bandwidth, etc.

For the size of the file uploaded we got the following results presented below in graph. The original file size was 9 MB and after encryption, the file uploaded was 12.96 MB. File sizes were used to compare and the original size and encrypted file size were plotted to get the following bar graphs.

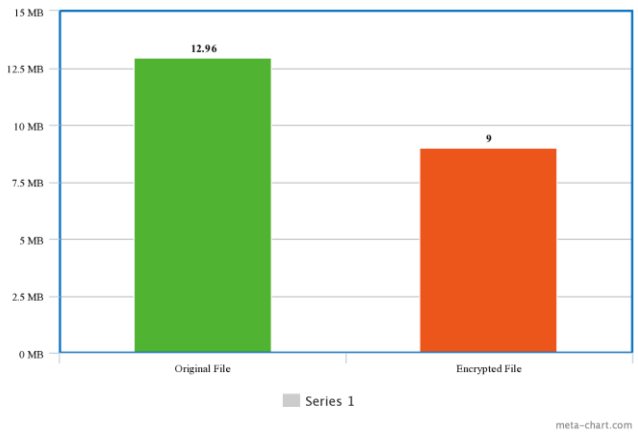

Fig 4.4. File size analyzed

4. CONCLUSIONS

Based on the survey it was identified that secure file storage and sharing would not only require confidentiality but also authentication and integrity. To overcome these drawbacks an architecture was designed which tries to provide a complete solution for securely storing the files and implemented in the form of a java application where users can securely store and share files.

ACKNOWLEDGEMENT

We would like to thank HOD, Principal and Management of Pillai College of Engineering, New Panvel, India for their continued support and cooperation during this research work.

REFERENCES

[1] Punam V. Maitri, Aruna Verma, "Secure file storage in cloud computing using hybrid cryptography algorithm", IEEE WiSPNET 2016.
[2] M. Malarvizhi, J. Angela Jennifa Sujana, T.Revathi, "Secure File Sharing Using Cryptographic Techniques in Cloud", 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE).
[3] Jerzy Kaczmarek, Michał Wróbel, Modern Approaches to File System Integrity Checking, 2008 1st International Conference on Information Technology.
[4] Rashi Dhagat, Purvi Joshi, New Approach of User Authentication Using Digital Signature, 2016 Symposium on Colossal Data Analysis and Networking (CDAN).
[5] Bilal Habib, Bertrand Cambou, Duane Booher, Christopher Philabaum, Public Key Exchange scheme that is Addressable (PKA), IEEE CNS 2017.
[6] Fareed Zaffar, Gershon Kedem, Ashish Gehani, Paranoid: A Global Secure File Access Control System, ACSAC 2005.
[7] Tulip Dutta, Amarjyoti Pathak 2016 "Secure Data Sharing in Cloud Storage Using Key Aggregation Cryptography".
[8] Vishwanath S Mahalle, Aniket K Shahade "Enhancing the data security in Cloud by implementing hybrid (Rsa & Aes) encryption algorithm".
[9] Bhale Pradeep Kumar Gajendra, Vinay Kumar Singh, More Sujeet, "Achieving cloud security using third party auditor, MD5 and identity-based encryption", ICCCA2016.
[10] Mr. Rohit Barvekar, Mr. Shrajal Behere, Mr.Yash Pounikar, Ms. Anushka Gulhane, AN APPROACH TO HYBRID CRYPTOGRAPHY ON CLOUD ENVIRONMENT, 2018.
[11] Shakeeba S. Khan, Prof.R.R. Tuteja, "Security in Cloud Computing using Cryptographic Algorithms", 2015.
[12] Anjali Patil, Nimisha Patel, Dr. Hiren Patel "Secure data sharing using cryptography in a cloud environment", 2016.
[13] Fortine Mata, Michael Kimwele, George Okeyo, "Enhanced Secure Data Storage in Cloud Computing Using Hybrid Cryptographic Techniques (AES and Blowfish)"
[14] B.Swathi, Dr. Bhaludra Raveendranadh Singh, "Secure File Storage in Cloud Computing using Hybrid Cryptography Algorithm", 2017.
[15] Joseph Selvanayagam, Akash Singh, Joans Michael, Jaya Jeswani, "Secure File Storage on Cloud using Cryptography", 2018.
[16] Swarna C, Marrynal S. Eastaff, " Secure File Storage in Cloud computing using Hybrid Cryptography Algorithm", 2018.
[17] P.Chinnasamy, P.Deepalakshmi, "Design of Secure storage for Health-care Cloud using Hybrid Cryptography", 2018.
[18] Bindu Bala, Lovejeet Kamboj, Pawan Luthra, "Secure File storage in Cloud Computing using Hybrid Cryptography Algorithm", 2018.
[19] Sarita Kumari, "A research paper on Cryptography Encryption and Compression Techniques", 2017.
[20] Prof. Swapnil Chaudhari, Mangesh Pahade, Sahil Bhat, Chetan Jadhav, Tejaswini Sawant, "A research paper on New Hybrid Cryptography Algorithm", 2018