CS6611 - Creative and Innovative Project

Secure Data Sharing in Dynamic Group , based on Cloud

Team Members:

Athiban T - 2018103031

Shree Harish S - 2018103591

Prathesh N - 2018103576

Team No : 8

Batch N

12-04-2021

Contents:

Introduction:

- We must provide security guarantees for the sharing data files since they are outsourced. Unfortunately, because of the frequent change of the membership, sharing data while providing privacy-preserving is still a challenging issue, especially for an untrusted cloud due to the collusion attack.
- Moreover, for existing schemes, the security of key distribution is based on the secure communication channel, however, to have such a channel is a strong assumption and is difficult for practice. In this paper, we propose a secure data sharing scheme for dynamic members.
- We have a group manager who organizes the members of the group and we have the users who join and leaves the group accordingly. Then we extend this principe to multiple groups such that each user can be part of multiple groups and each group can have multiple members who can upload and download the files in the group.

Literature Survey

Sno	Author	Publication	Year	Methodology	Advantages	Limitations
1	M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu	"Plutus: Scalable Secure File Sharing on Untrusted Storage," Proc. USENIX Conf. File and Storage Technologies	2003	Data Encryption Standard (DES) algorithm, Lazy revocation.	a cryptographic storage system that enables secure data sharing on untrustworthy servers based on the techniques that dividing files into filegroups and encrypting each file_group with a file-block key.	the complexities of user participation and revocation in these schemes are linearly increasing with the number of data owners and the revoked users.
2	Shucheng Yu, Cong Wang, Kui Ren, and Weijing Lou	"Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing," Proc. ACM Symp. Information, Computer and Comm. Security	2010	Bilinear Mapping , Attribute Based Encryption, Proxy Re-encryption	It exploited and combined techniques of key policy attribute-based encryption, proxy re-encryption and lazy re-encryption to achieve fine-grained data access control without disclosing data contents.	the single-owner manner may hinder the implementation of applications, where any member in the group can use the cloud service to store and share data files with others.

Literature Survey

Sno	Author	Publication	Year	Methodology	Advantages	Limitations
3	Xuefeng Liu, Yuqing Zhang, Boyang Wang, and Jingbo Yang	"Mona: Secure Multi-Owner Data Sharing for Dynamic Groups in the Cloud," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 6, pp. 1182-1191, June 2013.	2013	Bilinear Mapping, Dynamic Broadcast Encryption	Mona was very useful in a way that it need not update the key when you add or remove members.	Mona suffered from the problems of establishment of secure communication channels for key distribution and also collusion attacks which led to revoked users receiving files that were not meant to be received by them.
4	R. Lu, X. Lin, X. Liang, and X. Shen	"Secure Provenance: The Essential of Bread and Butter of Data Forensics in Cloud Computing," Proc. ACM Symp. Information, Computer and Comm. Security, pp. 282-292, 2010.	2010	Bilinear map and complex assumptions	a secure provenance scheme by leveraging group signatures and ciphertext-policy attribute-based encryption techniques . Each user obtains two keys after the registration while the attribute key is used to decrypt the data	the revocation is not supported in this scheme

Problem Statement:

- We need to provide a way of storing the data across managers and members.
- Files will be encrypted before uploading and it will be decrypted after downloading.
- We propose a secure way for key distribution without any secure communication channels.
- Our scheme can achieve fine-grained access control, any authorized user can access cloud.

- We can protect the scheme from collusion attacks revoked users cannot access data.
- We need to provide backward and forward secrecy.

  -Backward Secrecy, a newly joined user cannot access previously shared files.

  -Forward Secrecy, a removed user should not be able to access the future shared files.

Issues Identified:

- Need for the secure communication channel for key distribution.
- Lacking protection against collusion attacks.
- There is more workload on the group manager since he have to calculate values for every file uploaded.
- Lack of support for multi-group systems.
- A client-server architecture so that the server can do the heavy lifting of saving the data related to the system.

Domain:

Parallel Computing, Cryptography, Cloud Computing.

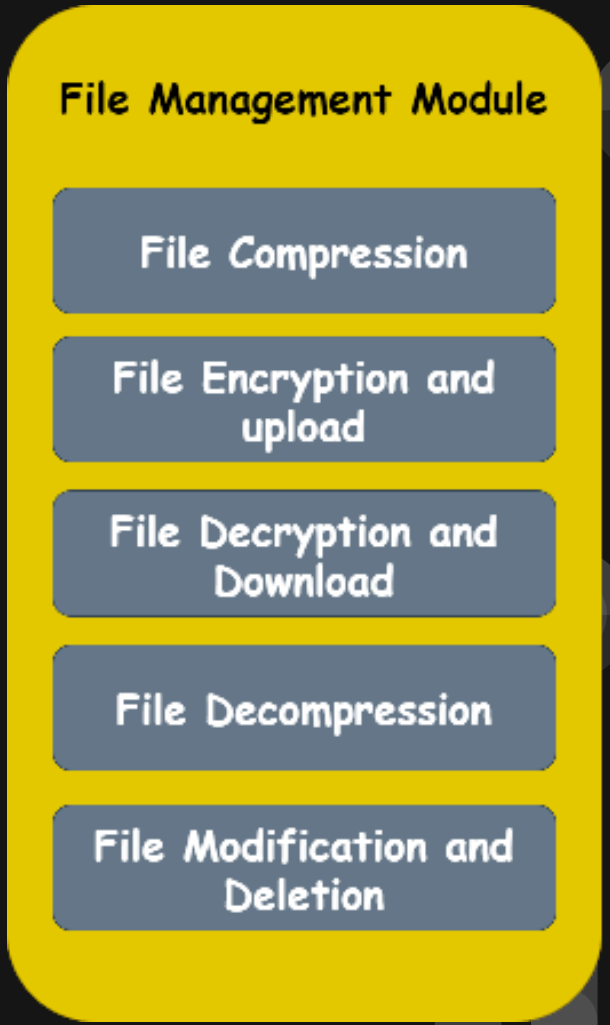
Approach:

- Parallel Computing is used in deciding the 256-bit prime numbers , since selecting a big prime number takes more computation time parallely computing it reduces computation time largely.
- Cryptography is used in adding users into the group and for file encryption.
- Cloud Computing is used for uploading and downloading files and folders.

Block Diagram

Sub Block Diagram

Sub Block Diagram



Modules and Deliverables:

Modules	Input	Output
System initialization module	Logged in Admin	Creating and deleting group , group manager monitoring and revocation
Group Manager Module	Logged in Manager	Registering new users , Revocation of members
Group Member Module	Logged in user	Request for joining in a group, file upload and download
File Management Module	File for sharing	File compression, encryption, decryption and decompression

Modules:

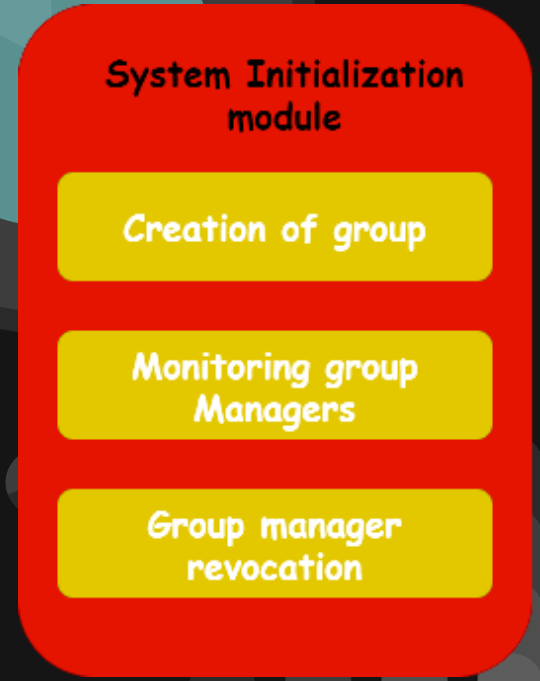
Module 1 : System Initialization Module

Input : Logged in Admin

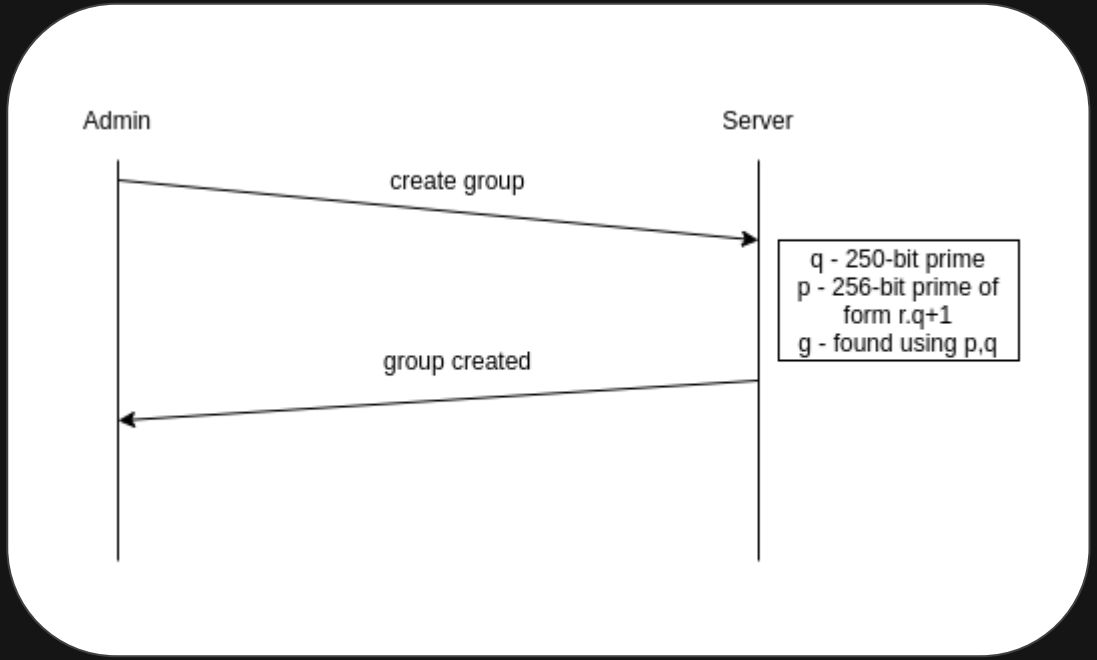
Output : Groups Management

In this module the logged in admin can do some group management tasks.

- Creation of a group on behalf of the group manager.
- Deletion of a group along with its data, members and managers.
- Viewing all existing group details.



Module 1 : System Initialization Module



The group is created by the admin. When the group is created the following order of tasks happens, the three values g p,q will be stored in the database as the group's public data.

Module 1 : System Initialization Module

```
ALGORITHM findPandQ
    q = getPrime(250)
    r = 2
    while (true)
        p = r * q + 1
        if (isPrime(p))
            if (bitLength(p) < 256)
                return { p, q }
            else
                q = getPrime(250)
                r = 1
                p = r * q + 1
        r+=1
```

P and Q are the basic requisites for a group.

```
ALGORITHM findG(p,q)
    let u = randomBetween(1,p)
    let g = u^((p-1)/q) mod p
    if (g == 1)
        findG(p, q)
    return g
```

g is the generator calculated by using p and q.

```
ALGORITHM DeleteGroup(groupId)
    Delete all files
    Delete all members
    Delete all requests
    Delete the manager
    Delete the group
```

Module 1 : System Initialization Module

S No	Input	Output
1	Creating a group Eg. create a group named 'group 1'	P,Q,G will be generated which are all randomly generated prime numbers which will be used to generate public and private keys. P: 4173268375… Q: 1682769506… G: 1286934615… All three are large prime numbers
2	Deleting a group Eg. Delete a group named 'group 1'	Delete the members list in the database and all the files associated with the group. Deletes the members of 'group 1' and all the files uploaded in 'group 1'.

Modules:

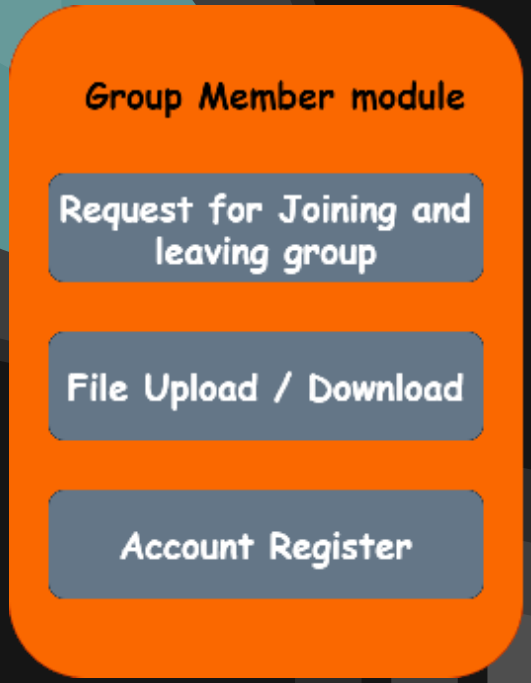
Module 2: Group Member Module

Input : Registered User

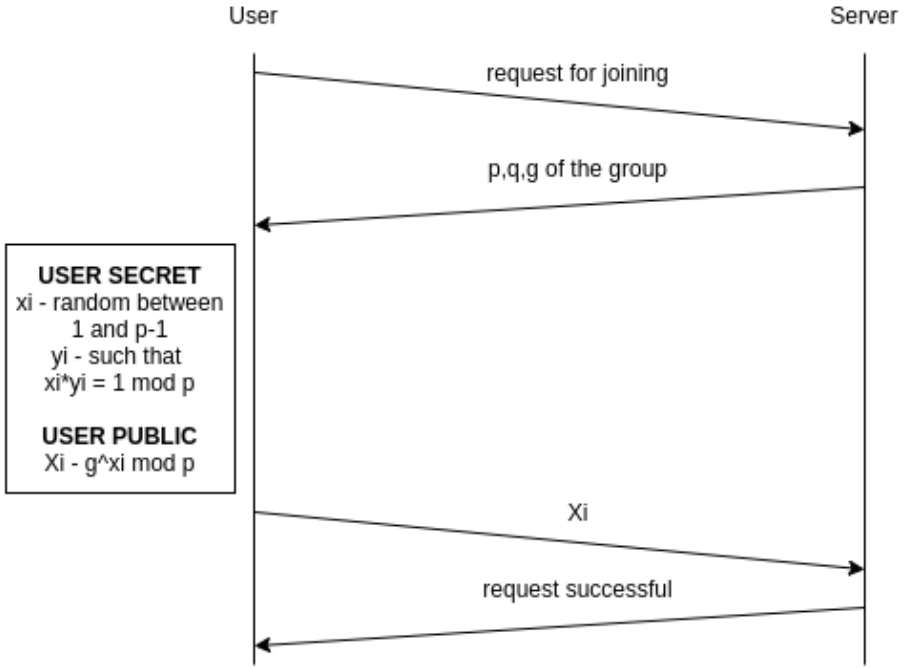
Output : Files sharing between other members

In this module the registered user can do the following tasks,

- Requesting to join a group.
- Uploading a file into the group.
- Downloading a file from the group.
- Deleting a file that the user has previously uploaded.



Module 2 : Group Member Module



User requesting to join the group

In order for the user to join the group the following tasks happens. The user's secret values are stored in the user device as private data and the user public value is sent to the server for storing the Xi => g^xi mod p value in the database.

Module 2 : Group Member Module

```
find_xi(p) {
      p1:= p - 1;
      xi := randNumberBetween 1 and p1
      while (gcd(xi, p1) !== 1)
            xi := randNumberBetween 1 and p1
      return xi
}
find_yi(xi, p) {
      let y := modInv(xi, p - 1);
      // thus xi*yi mod p is equivalent to 1
      return yi
}
find_Xi(g, xi, p) { return modPow(g, xi, p);
// returns g^xi mod p }
```

find_xi is one of the private keys generated by the user. This key is not known to any other users or even the group manager.

find_yi is another private key . It is computed in such a way that xi*yi mod p is equivalent to 1.

find_Xi : this is involved in the first step when a user asks to join in a group.

Module 2 : Group Member Module

S No	Input	Output
1	Signing up into the system Eg. User1 signs up into the system for the first time	Signing up creates public and private keys xi and yi in such a way that xi*yi mod p = 1 p: 4173268375… xi: 1892302414… Yi: 8910302134… Such that xi*yi mod p = 1
2	Requesting for addition and removal from the group.	This will be sent as a request to the group manager and he might accept or reject it, if he accepts the group key will be updated allowing this particular user not see anything before addition and anything after removal

Module 2 : Group Member Module - Output

All the requests made by the user

Details of the groups where you can examine your history of interactions with groups

User1's secret data is stored in clients machine


```
{
  "USER-1-1": {
    "p": "42144366793245396716227893492888489522912336514882657035214414289024566868499",
    "q": "1239540199801335197536114514449672028008565695632007814809454159670660490789 7",
    "g": "93149786338124380029522529841840645968167398210787698694301749081181047026 15",
    "xi": "5867536015003039856420220325384169088081215815218645187853899460520827789969",
    "yi": "42537298726034193092704375297156582359219089904715718533800205906221441874 51",
    "group_id": 1
  }
}
```


p,q,g the values specific to a group and xi, yi private keys getting stored in user's device

Modules:

Module 3: Group Manager Module

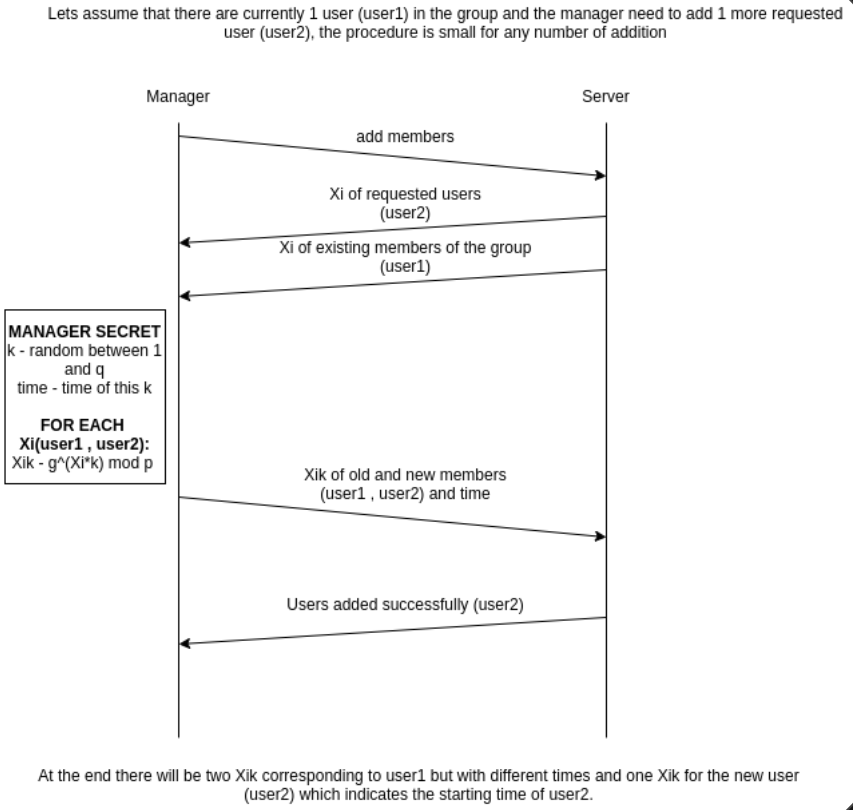
Input : Logged in Manager

Output : Group users and group files management

In this module the logged in manager can do group management tasks such as,

- Addition of a user to the group
- Addition of multiple users to the group at the same time.
- Removal of a member from the group.
- Removal of multiple members from the group at the same time.

Module 3: Group Manager Module

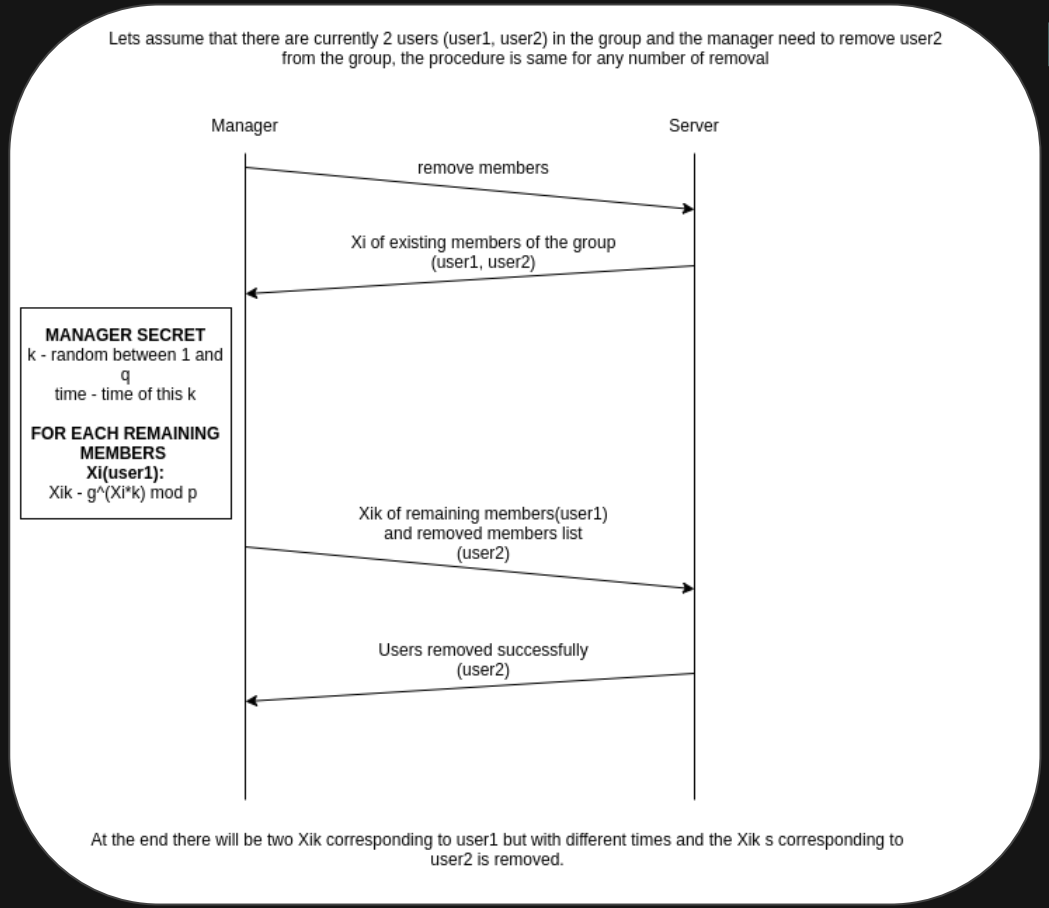


Addition of members to the group

In order for the addition of members to the group the following tasks happens in order, The manager's secret will be stored in the manager's local machine and Xik values of the old and new users are sent to the server for storing it in the database.

This feature also helps in adding multiple numbers of users into the group at the same time.

Module 3: Group Manager Module



Removal of members from the group

The removal of a member from the group involves the following tasks in order,The manager secret will be stored in the manager's local machine and Xik values of the old and new users are sent to the server for storing it in the database.

This feature also helps in removing multiple numbers of users into the group at the same time.

Module 3: Group Manager Module

```
find_k(q) {
        k := random Number between 1 and q
        while (gcd(k, q) !== 1)
                k = rand Number between 1 and q;
        return k;
}

find_Xik(Xi, k, p) {
        return modPow(Xi, k, p);
        // returns g^(xi*yi*k) mod p
        // g^(xi*k) mod p is Xi.
}
```

find_k : this is used by the group manager for the creation of a group key which will be shared to the user on successful joining and it will be later used to encrypt and decrypt the files.

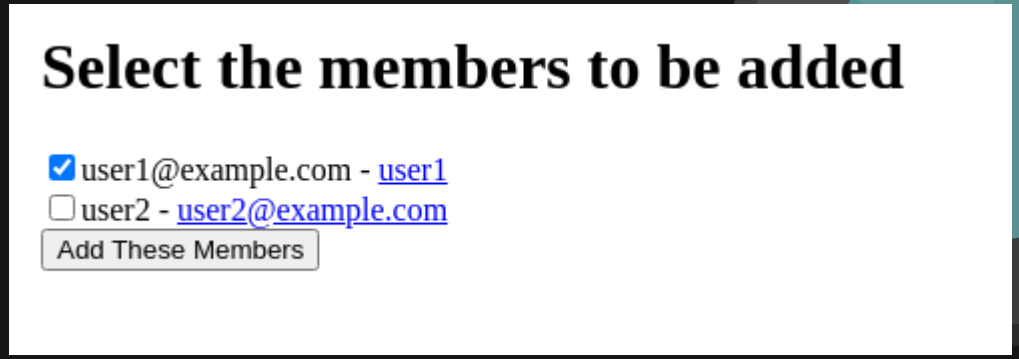
find_Xik : this is also used only by the group manager this computes g^(xi*k) mod p internally.

Module 3: Group Manager Module

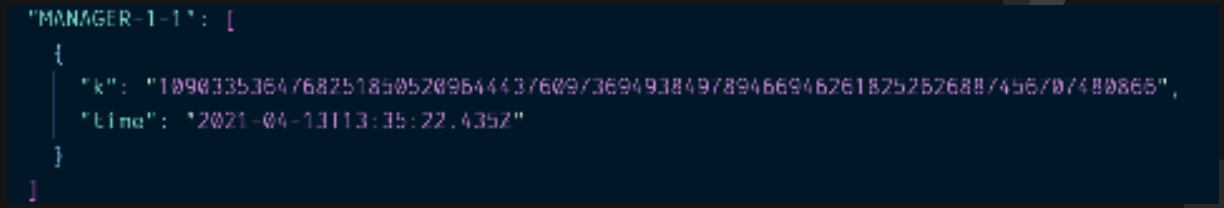
S No	Input	Output
1	A group addition request by an user Eg. user1 sending a request to 'group1' manager.	A key private to that user will be created and sent to the user, used for encryption and decryption. g^(xi*k) mod p will be generated and stored in the user's local storage.
2	Removing an user from the group or removal request to the manager. Eg. user1 sending a request for the removal from 'group1'	This deletes the entry in the database and updates the group key . After removal of a group user the key: k => 1459238...

Module 3: Group Manager Module - Output

- Manager adds the user1 to the system



- The following data will be added to the manager's local machine after any addition.



Modules:

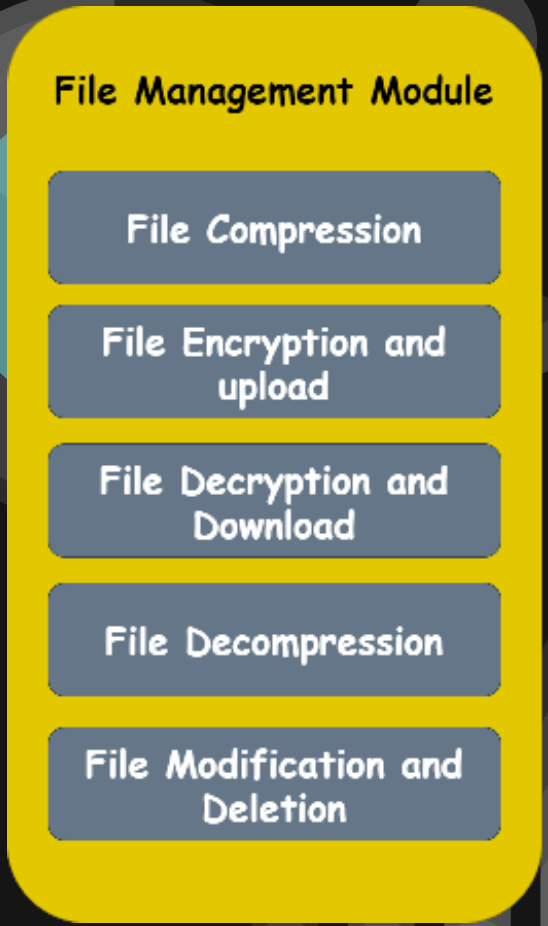
Module 4 : File Management Module

Input : Files for sharing

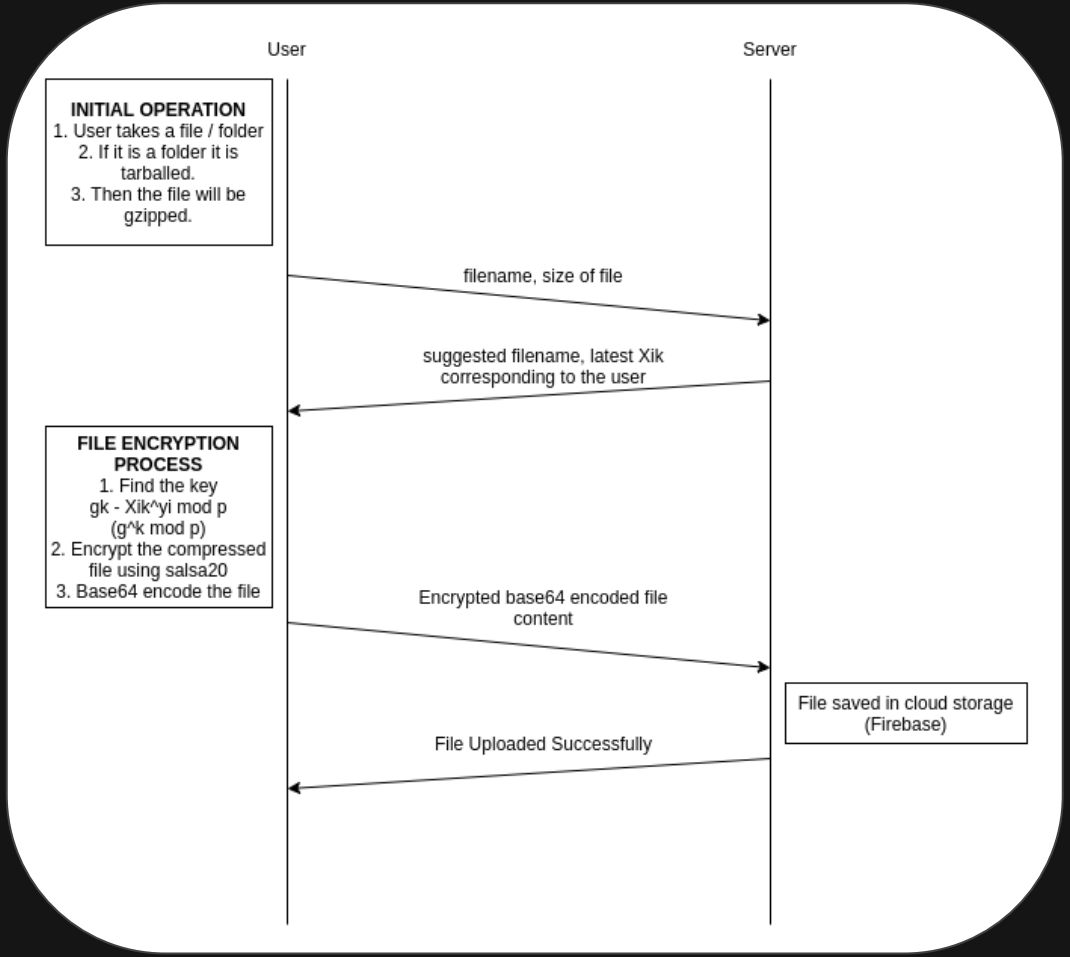
Output : Securing and storing the files

In this module the following tasks are accomplished such as,

- Files will be compressed / decompressed.
- Files will be encrypted / decrypted.
- Files will be uploaded to the cloud storage provider.
- Files will be downloaded from the cloud storage provider.



Module 4 : File Management Module

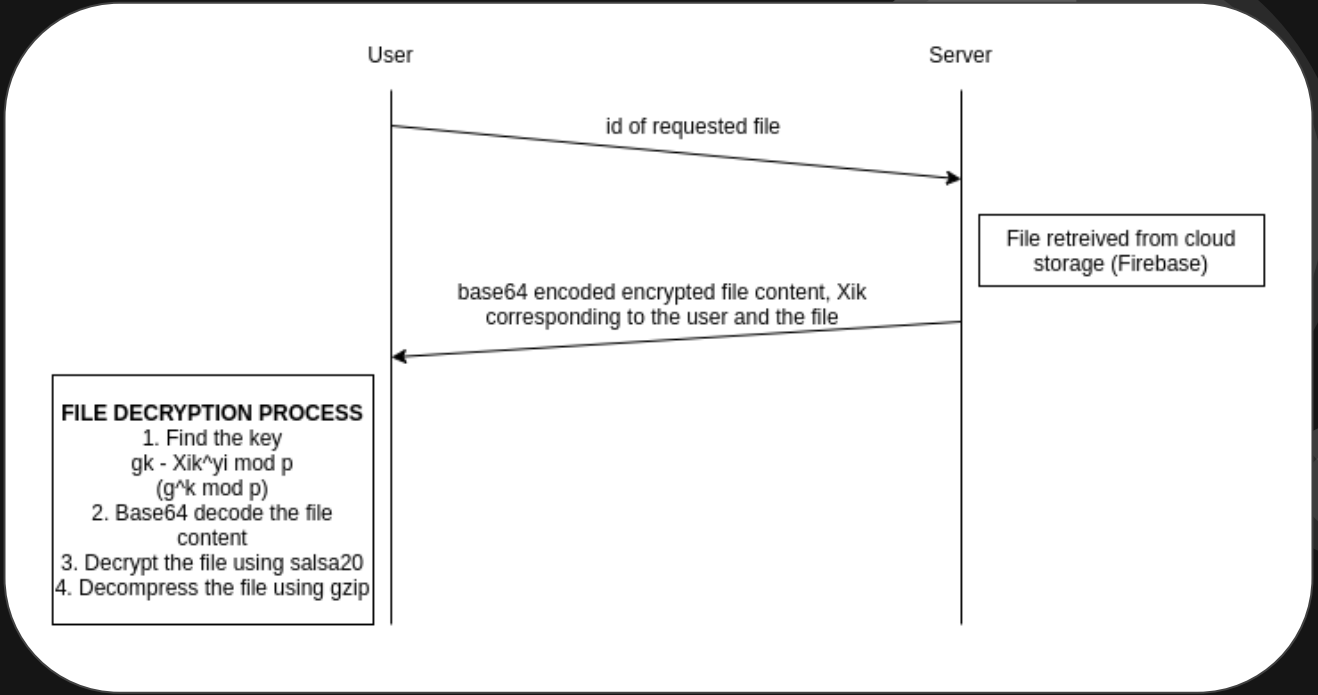


File Upload

The file uploading process involves the following tasks in order,The file is compressed using gzip before uploading. Then the filename and size is sent to the server.

The Xik value of the user is sent to the user and the user finding the group key using the user's secret value yi and the compressed file is encrypted and the data is base64 encoded and sent to the server and the server saves the file in the cloud storage system.

Module 4 : File Management Module



File Download

The file will be decrypted, decompressed and the original file is obtained by the user.

Module 4 : File Management Module

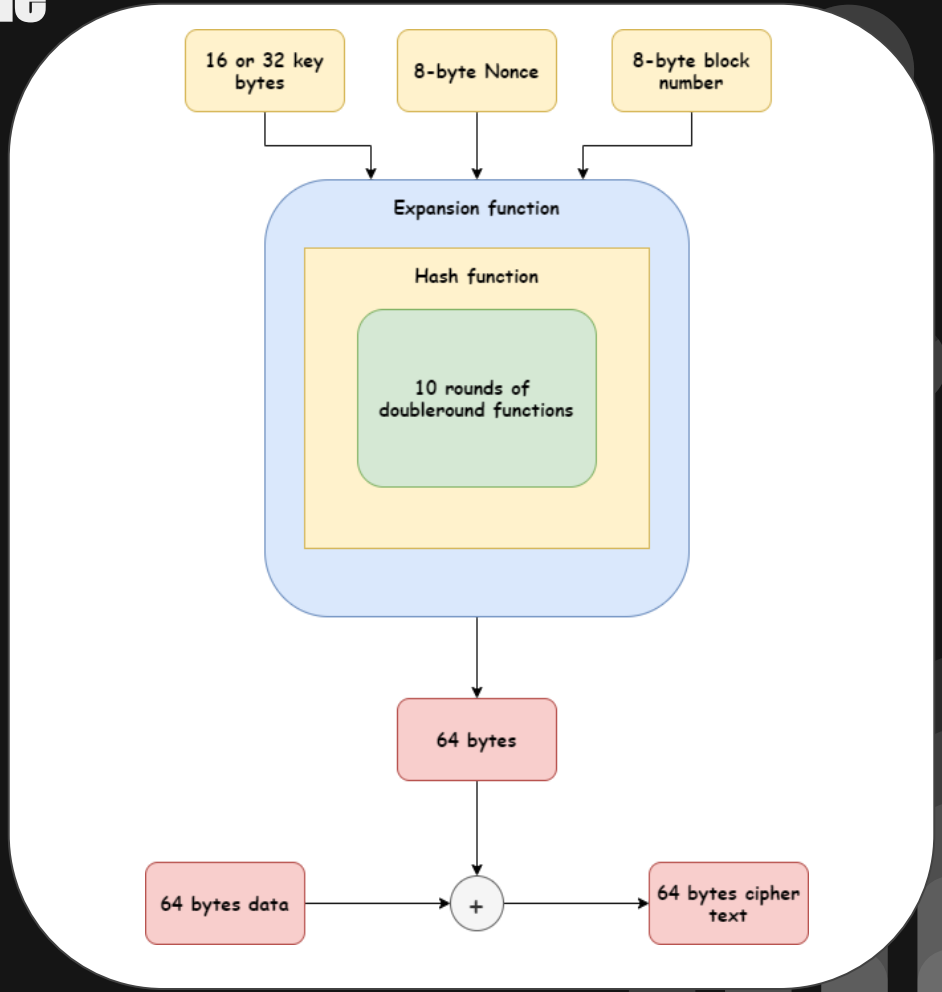
```
function CompressFile(file path) {
        zlib.gzipSync(read stream of the file);
        Create a gz file.
        Write the compressed file into the gz
        file using fs.writeFileSync('tar.gz
        path' ,compressed file)
}
```

The files uploaded by the users can be very large, encrypting the entire file might create an overhead problem increasing the uploading time. The files are compressed using zlib.

```
function CompressFolder(folder Path){
        tar.pack('folder path').pipe('create a output
        stream tar with utf-8 encoding')
                .on('close',()=>{
                    Gzip the tar file using zlib.gzipSync()
                        Create a tar.gz path
                        Write the compressed file into
                        tar.gz.  using fs.writeFileSync('tar.
                        gz path' ,compressed file)
                })
        )
}
```

Module 4 : File Management Module

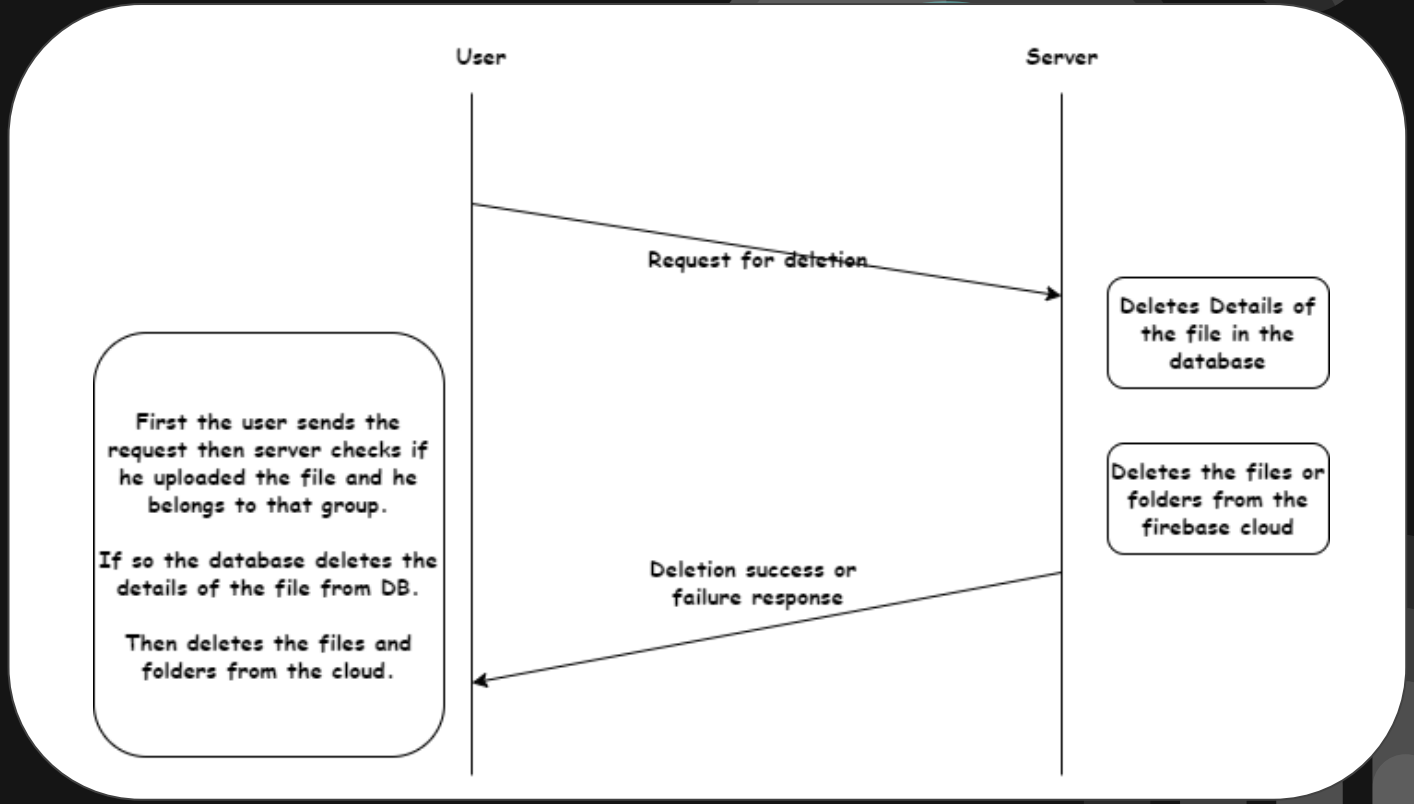
Salsa20 is considered to be a well-designed and efficient algorithm. There are two important functions: Expansion and Hash. Input to expansion is a 16 or 32 byte secret key , 8 byte nonce and 8 byte block number which increments for every function call. The same algorithm is used during the decryption. There are several important functions : sum of two words (a+b mod 32), xor of two words (a xor b), binary left rotation (2^a.w mod(2^32 - 1) => w<<<a) , littleendian , Quarterround, Rowround, columnround and the doubleround function.



Module 4 : File Management Module

File Deletion

First the credentials of the user is checked . Then the details of the file will be first deleted in the database. Followed by a deletion from the cloud.



Module 4 : File Management Module

```
Function delete(storageFileName){
        file := bucket.file(storageFileName);
        await file.delete() //deletes the file from the cloud
        Return the response .
}
Function deleteMany(storageFolderName){
        Delete the file which has the prefix 'storage
Folder Name'
        await bucket.deleteFiles({
                Prefix: 'storageFolderName'
        })
}
```

This function is used to delete a file from the cloud storage. Mostly this function is used by the group member to delete a file uploaded by him/her and the group manager to delete a single file uploaded in his/her group.

The function deleteMany is used to delete the files in a folder from the cloud storage. This function is used by the group manager to delete folders in his/her group, group members to delete the folders uploaded by them and by the admin to delete the folders while deleting the group.

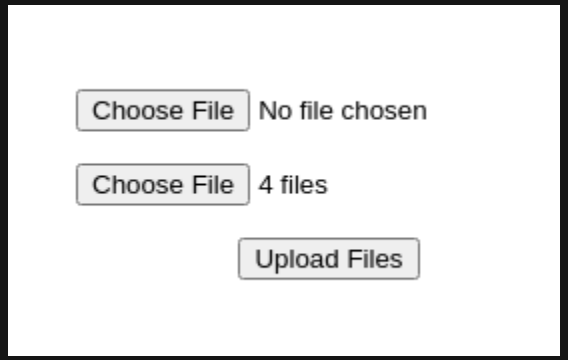
Module 4 : File Management Module

S No	Input	Output
1	File upload Eg.User1 uploads a file u1.txt User2 uploads a folder u2	File will be compressed and then encrypted with the group key and then finally uploaded into the cloud, metadata will be stored in the database. Stored as .gz.enc in the cloud. Stored as .tar.gz.enc in the cloud.
2	File download Eg. user1 downloads u1.txt User2 downloads u2	File will be decrypted using the group key used for encryption and then decompressed and then finally downloaded into the local storage. Downloads the original file u1.txt. Decrypts and decompress into u2. Note : the structure of u2 is maintained.

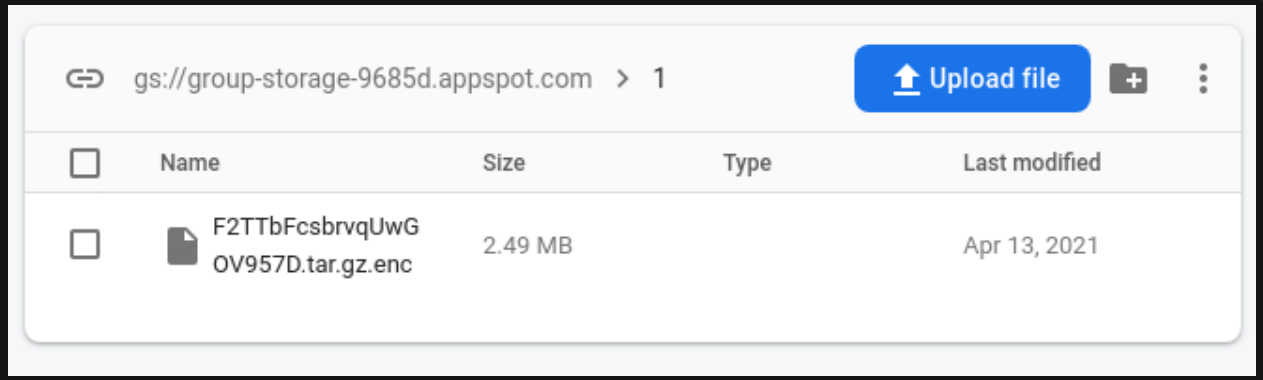
Module 4 : File Management Module

S No	Input	Output
3	File deletion	First the details of the file will be deleted in the database and then deletes the content from the cloud
	Eg. User1 tries to delete the file u1.txt	Deletes the metadata of u1.txt from db and deletes it permanently from the cloud.
	Eg. User2 tries to delete the folder u2	Deletes all the files in the folder u2.

Module 4 : File Management Module - Output



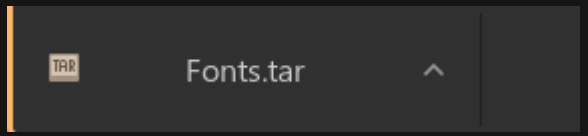
User uploads a folder consisting of 4 files



It is uploaded in the cloud as tar.gz.enc . 'tar.gz' indicates that it is a compressed folder and enc indicates that the compressed folder is encrypted.

Module 4 : File Management Module - Output

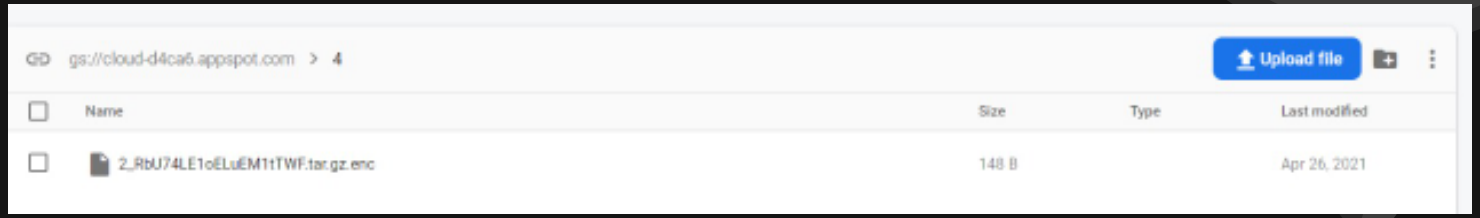
Other users in the group can view the files which can be downloaded.



If we click the download button it is downloaded as tar file, which indicates that it has decrypted and unzipped.

Module 4 : File Management Module - Output

There was two files in the cloud user1 a file uploaded by user1 and user2 a folder uploaded by user 2 . Manager deletes the file user1. And it is subsequently replicated in the database and cloud that only encrypted user1.txt is present.



In Firebase

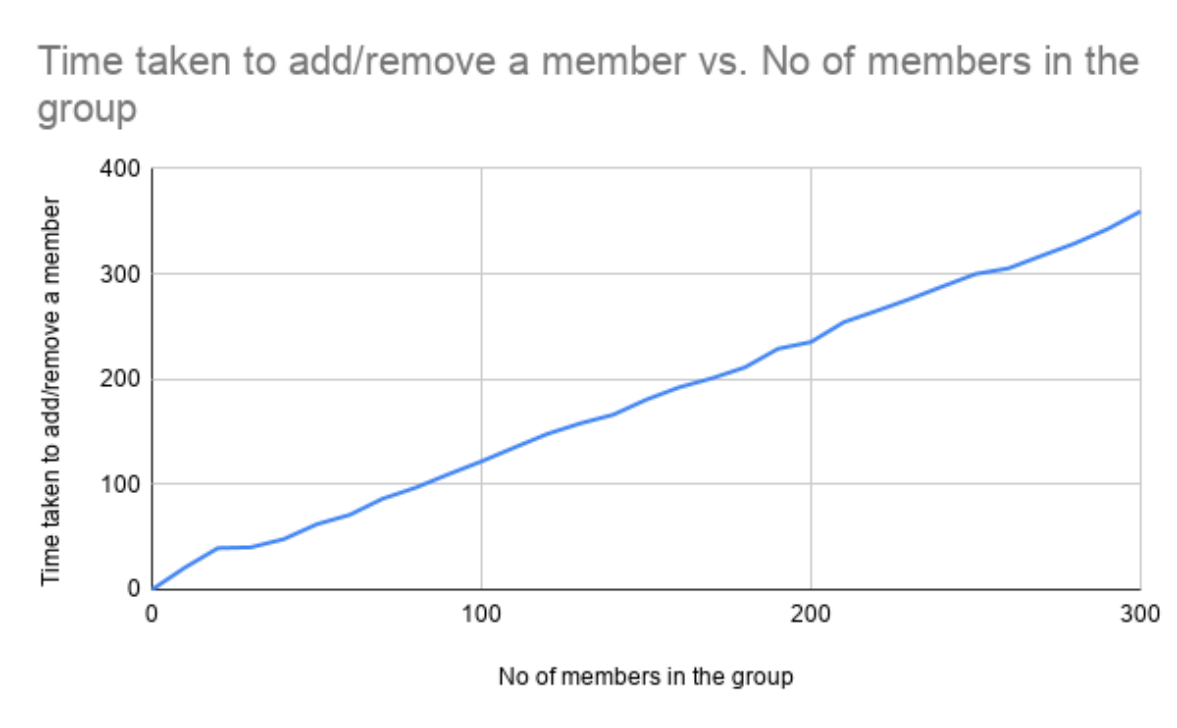
Data Output

	id [PK] integer	user_id integer	group_id integer	original_filename text	storage_filename text	file_time timestamp without time zone
1	11	7	4	user2.tar.gz	2_RbU74LE1oELuEM1tTWF....	2021-04-26 11:36:56.251

In the database

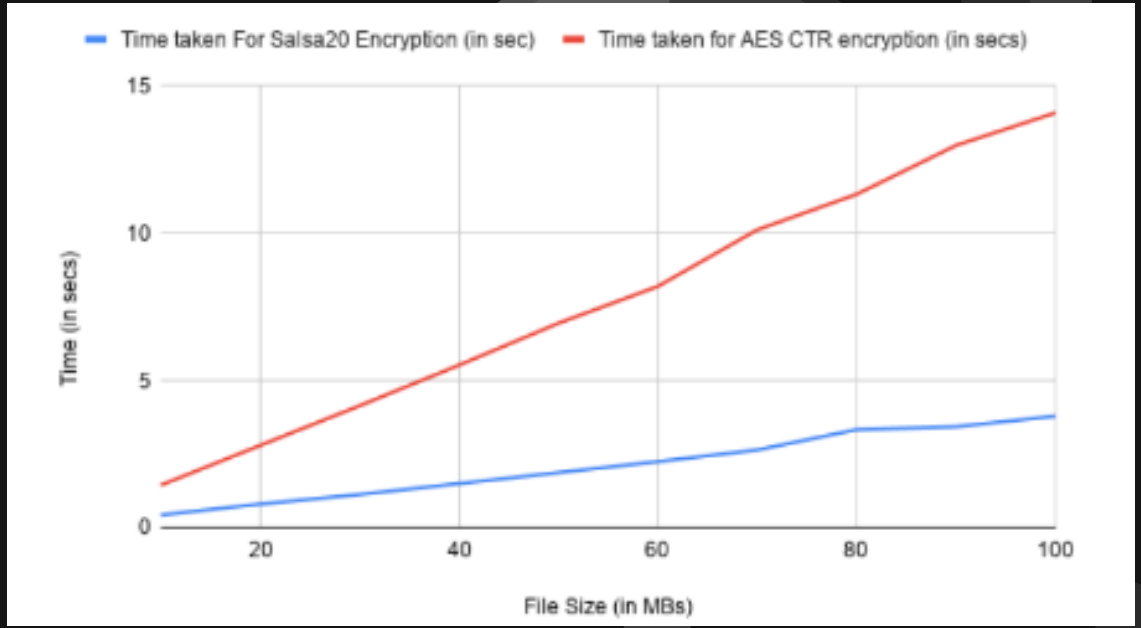
Performance Evaluation

The performance of addition or removal of a member from the group is based on the number of members currently in the group and the number of members added or removed. The members addition or removal takes a modular exponentiation operation for each user in the cumulative sum. The algorithm that is taken for modular exponentiation is based on the technique of repeated square and multiply.



Performance Evaluation

Similarly, time to encryption is greatly reduced by introducing salsa20 algorithm instead of standard AES algorithm. The reduction in time for various sizes of file is as follows. This is due to the fact that salsa20 algorithm works mosly on simple mathematical functions such as xor when compared to AES algorithm.



Conclusion and Future Direction:

The client-server architecture is achieved. The key distribution among the members of the group without the use of a secure communication channel is achieved. For effective bandwidth and storage size reduction, the file is compressed. Then the file data is encrypted with the help of salsa20 encryption algorithm. This algorithm is chosen for its speed and its security advantages. The workload on the manager side is reduced since all the manager needs to do is modular exponentiation for all the members of the group.

The limitation of this algorithm being that it takes more space and the time for member addition and removal depends on the number of users in the group. In the future, there can be independence developed between the number of users in the group and the users addition and optimal storage of the values and reducing the number of values stored in the database needs to be reduced.

References:

- M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable Secure File Sharing on Untrusted Storage," Proc. USENIX Conf. File and Storage Technologies, pp. 29-42, 2003.
- Shucheng Yu, Cong Wang, Kui Ren, and Weijing Lou, "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing," Proc. ACM Symp. Information, Computer and Comm. Security, pp. 282-292, 2010.
- R. Lu, X. Lin, X. Liang, and X. Shen, "Secure Provenance: The Essential of Bread and Butter of Data Forensics in Cloud Computing," Proc. ACM Symp. Information, Computer and Comm. Security, pp. 282-292, 2010.
- Lan Zhou, Vijay Varadharajan, and Michael Hitchens, "Achieving Secure Role-Based Access Control on Encrypted Data in Cloud Storage," IEEE Transactions on Information Forensics and Security, vol. 8, no. 12, pp. 1947-1960, December 2013.
- Xuefeng Liu, Yuqing Zhang, Boyang Wang, and Jingbo Yang, "Mona: Secure Multi-Owner Data Sharing for Dynamic Groups in the Cloud," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 6, pp. 1182-1191, June 2013.