International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Secure and Memory Efficient De-Duplication On Encrypted Data in Cloud Storage

Divya R, Arthi R, Indhumathi .M, Dr. U. V. Arivazhagu, M.E, M.B.A, Ph.D

Professor & Head of the Department **of** CSE, Kingston Engineering College

Abstract: ***Today most of the Attackers target to social media network data's, this happen in malicious Environment. User doesn't know if our sharing data may be misused by attacker. We propose novel concept "OTP" based data view, data owner can ease to identify malicious user, because consumer should be validate for receive any data from data owner. All stored data must be based on Data Lineage concept .Data Lineage means share one copy of data to all users and also maintain all accessed consumer information for that data. In this way we can avoid Duplications and easy to identify data leakage. Effectively manage Database memory. In this paper, we propose a novel server-side deduplication scheme for encrypted data. It allows the cloud server to control access to outsourced data even when the ownership changes dynamically by exploiting randomized convergent encryption and secure ownership group key distribution. This prevents data leakage not only to revoked users even though they previously owned that data, but also to an honest-but-curious cloud storage server. In addition, the proposed scheme guarantees data integrity against any tag inconsistency attack. Thus, security is enhanced in the proposed scheme. The efficiency analysis results demonstrate that the proposed scheme is almost as efficient as the previous schemes, while the additional computational overhead is negligible.***

Keywords: De-duplication, Cloud storage, Encryption, OTP, Revocation.

1. Introduction

Cloud computing provides scalable, low-cost, and location-independent online services ranging from simple backup services to cloud storage infrastructures. The fast growth of data volumes stored in the cloud storage has led to an increased demand for techniques for saving disk space and network bandwidth.

Most of the attacker easily steal the key from the cloud server and misuse it. In this paper we provide a security to key by using one time password and we avoid data duplication.

We provide a ownership to the data owner by using the proof of ownership. The hash function for the data will be generated using the MD5 algorithm. And the key, encryption, decryption are done using the DES algorithm.

Deduplication techniques can be categorized into two different approaches: deduplication over unencrypted data and deduplication over encrypted data. We propose a deduplication scheme over encrypted data. The proposed scheme ensures that only authorized access to the shared data is possible, which is considered to be the most important challenge for efficient and secure cloud storage services in the environment where ownership changes dynamically. It is achieved by exploiting a group key management mechanism in each ownership group. As compared to the previous deduplication schemes over encrypted data, the proposed scheme has the following advantages in terms of security and efficiency. First, dynamic ownership management guarantees the backward and forward secrecy of deduplicated data upon any ownership change. As opposed to the previous schemes, the data encryption key is updated and selectively distributed to valid owners upon any ownership change of the data through a stateless groupkey distribution mechanism using a binary tree. The ownership and key management for each user can be conducted by the semi-trusted cloud server deployed in the system.

However, previous deduplication systems cannot support differential authorization duplicate check, which is important in many applications. In such an authorized deduplication system, each user is issued a set of privileges during system initialization. Each file uploaded to the cloud is also bounded by a set of privileges to specify which kind of users is allowed to perform the duplicate check and access the files. Before submitting his duplicate check request for some file, the user needs to take this file and his own privileges as inputs. The user is able to find a duplicate for this file if and only if there is a copy of this file and a matched privilege stored in cloud.

2. Background Work

Xuexue Jin et al(2013) ,Explained that Cloud computing is viewed as the next generation architecture of IT companies. As promising as it is, cloud computing also brings forth many new security issues when users outsource sensitive data to cloud servers. To keep sensitive users' data confidential against untrusted servers, existing solutions usually apply cryptographic methods. With data encryption, the same file will become different from each other, thus deduplication which is widely adopted by cloud storage service providers meets some challenges. Current method to solve the problem is to make use of some information computed from the shared file to achieve deduplication of encrypted data, say convergent encryption. But this piece of information which is computable from the file via a deterministic public algorithm is not really meant to be secret. To this end, we propose a scheme to address the deduplication of encrypted data efficiently and securely with the help of **ensure** ng the ownership of the shared file, encrypting data using keys at user's will and realizing the anonymous store through the digital credential. We achieve this aims through proof of ownership (POW), proxy re-encryption (PRE) and digital credential..

Jong Hwan Park et al(2008) ,explained that Broadcast encryption allows a sender to securely distribute messages to a dynamically changing set of users over an insecure channel. In a public key broadcast encryption (PKBE) scheme, this encryption is performed in the public key setting, where the public key is stored in a user's device, or directly transmitted to the receivers along with ciphertexts. In this paper, we propose two PKBE schemes for stateless receivers which are transmission-efficient. A distinctive feature in our first construction is that, different than existing schemes in the literature, only a fraction of the public key related to the set of intended receivers is required in the decryption process. This feature results in the first PKBE scheme with O(r) transmission cost and O(1) user storage cost for r revoked users. Our second construction is a generalized version of the first one providing a tradeoff between ciphertext size and public key size. With appropriate parametrization, we obtain a PKBE scheme with (Oradicn) transmission cost and O(1) user storage cost for any large set of n users. The transmission cost of our second scheme is at least 30\% less than that of the recent result of Boneh et al.'s PKBE scheme, which is considered as being the current state-of-the-art. By combining the two proposed schemes, we suggest a PKBE scheme that achieves further shortened transmissions, while still maintaining O(1) user storage cost. The proposed schemes are secure against any number of colluders and do not require costly re-keying procedures followed by revocation of users.

Kazuhide Fukushima et al(2009) ,explained that Digital content distribution services require that 1) only valid user devices that has a valid key can decrypt the broadcasting content, 2) the keys can no longer be used to decrypt the content, if keys in a device are revealed, and 3) invalid users who illegally use keys in a device can be identified. This paper proposes a broadcast encryption scheme with traitor tracing based on the ternary tree structure. We design a new cover-finding algorithm and label assignment algorithm in order to achieve a coalition-resistant revocation and tracing schemes. In our scheme, the number of labels stored in a client device can be reduced by about 20.4 percent and the average header length by up to 15.0 percent in the case where the total number of devices is 65,536. The efficiency of the traitor tracing is the same as the complete subtree method, and its computational cost imposed on a client device stays within O(logn). Our scheme is an improvement of the complete subtree and difference subset methods.

Xiaofeng Chen et al(2015), explained that Data deduplication is a technique for eliminating duplicate copies of data, and has been widely used in cloud storage to reduce storage space and upload bandwidth. However, there is only one copy for each file stored in cloud even if such a file is owned by a huge number of users. As a result, deduplication system improves storage utilization while reducing reliability. Furthermore, the challenge of privacy for sensitive data also arises when they are outsourced by users to cloud. Aiming to address the above security challenges, this paper makes the first attempt to formalize the notion of distributed reliable deduplication system. We propose new distributed deduplication systems with higher reliability in which the data chunks are distributed across multiple cloud servers. The security requirements of data confidentiality and tag consistency are also achieved by introducing a deterministic secret sharing scheme in distributed storage systems, instead of using convergent encryption as in previous deduplication systems. Security analysis demonstrates that our deduplication systems are secure in terms of the definitions specified in the proposed security model. As a proof of concept, we implement the proposed systems and demonstrate that the incurred overhead is very limited in realistic environments.

Dharani p et al(2015) explained that which is widely used in cloud to reduce storage space and increase bandwidth. Convergent encryption has been extensively adopted for secure deduplication, in order to use efficiently and reliably manage a huge number of convergent keys. A baseline approach named as Dekey is used to distribute the convergent key which would be shared across multiple servers. But implementation of Dekey using the Ramp secret sharing scheme has some limitations; a heavy computational cost is required to make n shares and recover the secret as a solution to this problem. Hence a new (k, L, n)-threshold ramp scheme (extension of existing ramp scheme) is proposed which is perfect, idle and faster secret sharing scheme, every combination of k or more participants can recover the secret, but every group of less than k participants cannot obtain any information about the secret.

3. System Architecture

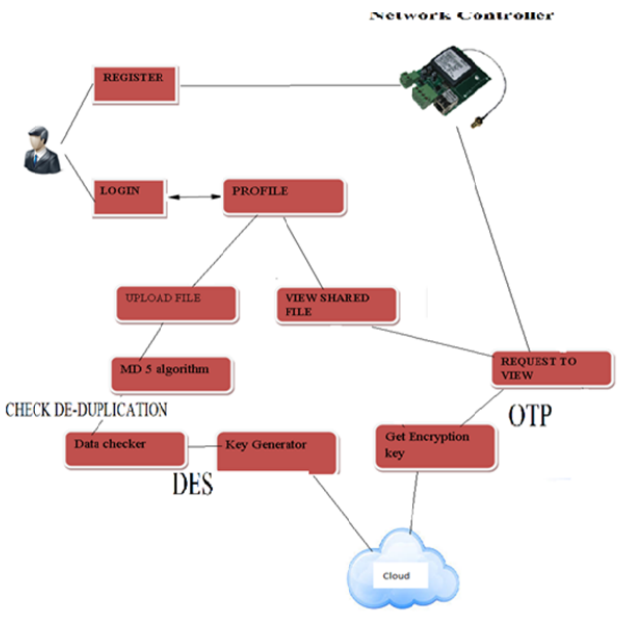


Figure 1: System architecture

In this section we describe a detailed description of the architecture. At first the user login in to the cloud and creates a profile. User first uploads the file. The file is sent to the data checker for giving the ownership for the user.

It is checked using the MD5 algorithm. MD5 algorithm generates the hash function. Message-Digest (Fingerprint) algorithms are special functions which transform input of (usually) arbitrary length into output (so-called "fingerprint" or "message digest") of constant length. Message-Digest algorithms serve in digital signature applications for guaranteeing consistency (integrity) of data. Commonly used model is as follows (message-digest in cooperation with asymmetric cryptography): And if the file already exists the data checker can easily identify using hash function. If the file does not exist already then the key will be generated and stored in the cloud. The ownership will be given to data owner. The key will be generated using the DES algorithm. This key performs the binary rotation operation. It encrypts and decrypt the given message that includes key generation algorithm also. It performs binary rotation process for the security of the key. The uploaded file will be in encrypted form if other users want to view the file they give request to view the file. After receiving the request from the other user the one time password will be sent to their mail. Other user gets the link and one time password to view the file. They must view the link and enter into the link to view the uploaded file. By this way that data can be kept secured and other has no authorization to forward those files to some other. If the same file is uploaded again the data checker finds and does not accepts the file and does not give the ownership to that particular user. It is stored in the cloud.

4. Deduplication on Encrypted Data

- DeDuplication using MD5.
- Data set creation and finding duplication.
- Shared Dataset.
- Security and key generator

5. Deduplication using MD5

Data owner can upload data's, that data are split into multipart data then send to trusted data checker, job of the data checker is to generate signature key from MD5 and compare with previous keys, if mismatch then that data send to Key generator Server, Job of the key generator are generate encryption key as user specified algorithm ,finally encrypt then store in Database.

The deduplication algorithm should guarantee tag consistency against any poison attacks. That is, the deduplication algorithm should allow the valid owners to verify that the data downloaded from the cloud storage have not been altered. It sends to trusted data checker, job of the data checker is to generate signature key from MD5 A data owner encrypts the data and outsources it to the cloud storage.

This is a client who owns data, and wishes to upload it into the cloud storage to save costs. A data owner encrypts the data and outsources it to the cloud storage with its index information, that is, a tag. If a data owner uploads data that do not already exist in the cloud storage, he is called an initial uploader; if the data already exist, called a subsequent uploader since this implies that other owners may have uploaded the same data previously, he is called a subsequent uploader.

Dataset Creation and Finding Deduplication

In this Module We create data owner dataset, this dataset only map owner with our upload data's , we maintain common database for effectively find duplications. The files will be uploading only once. If another data owner going to upload the same file in database means they will get the notification (the data is already uploaded in database).So data owner can save cost and time.

In this Module We create data owner dataset, this dataset only map owner with our upload data's .we maintain common database for effectively find duplications.

The files will be uploading only once. So data owner can save cost and time. This is an entity that provides cloud storage services. It consists of a cloud server and cloud storage.

Duplication may done in 2 ways. They are;

- Duplication over encrypted data
- Duplication over non encrypted data

In order to preserve data privacy against inside cloud server as well as outside adversaries, users may want their data encrypted. In the context of deduplication, backward secrecy means that any user should be prevented from accessing the plaintext of the outsourced data before uploading the data. Conversely, forward secrecy means that any user who deletes or modifies the data in the cloud storage should be prevented from accessing the outsourced data after its deletion or modification.

6. Shared Dataset

Share Dataset is an light weight dataset that only contain mapping file metadata information, in our project we maintain one common big data database instead of unique because efficiently find duplication and memory management, if data owner share our data to client that data not replicate instead map client name.

This is an entity that provides cloud storage services. It consists of a cloud server and cloud storage. The cloud server deduplicates the outsourced data from users if necessary and stores the deduplicated data in the cloud storage. The cloud server maintains ownership lists for stored data, which are composed of a tag for the stored data and the identities of its owners. The cloud server controls access to the stored data based on the ownership lists and manages (e.g., issues, revokes, and updates) group keys for each ownership group as a group key authority. The cloud server is assumed to be honest-but-curious. That is, it will honestly execute the assigned tasks in the system; however, it would like to learn as much information about the encrypted contents as possible. Thus, it should be deterred from accessing the plaintext of the encrypted data even if it is honest.

7. Security and Key Generator

We are implementing "Dynamic Encryption key Generation". It means all shared data only view with data owner permission, so we can avoid from unknown access. Unauthorized users who cannot prove ownerships should not be able to decrypt the ciphertext stored in the cloud storage. Additionally, the cloud server is no longer fully trusted in the system. Thus, unauthorized access from the cloud server to the plaintext of the encrypted data in the cloud storage should be prevented.

Social users are group members they can only view and share the data. If want show the data mean they need to get permission to data owner then data owner will send Encryption key after they can view the data. If data owner does not provide the KEY mean user cannot view the file Key will be generator using the DES key generating algorithm. The key is kept safe by using the rotation shift. Des performs binary rotation .so the key is kept safe.

8. MD5 Algorithm

MD5 is an algorithm that is used to verify data integrity through the creation of a 128-bit message digest from data input (which may be a message of any length) that is claimed to be as unique to that specific data as a fingerprint is to the specific individual.

```
 Step 1. Append Padding Bits
 Step 2. Append Length
 Step 3. Initialize MD Buffer
 Step 4. Process Message in 16-Word Blocks
 Step 5. Output
Procedure PLR,GLR(f,x)
 If $|\Delta Hp_t - \Delta Hp_{t-1}| < |\Delta Hp_{t-1} - \Delta Hp_{t-2}|$ then
 //promote exploration
 if lattConf = 2(n/2 ×n) then
 lattConf =4(n/2×n/2);
 else if lattConf=4(n/2×n/2) then
 lattConf = (n×n);
 else if lattConf=(n×n) then
 lattConf =n(1×n/2);
 else
 lattConf =n(1×n/2);
 end if
 else if $|\Delta Hp_t - \Delta Hp_{t-1}| > |\Delta Hp_{t-1} - \Delta Hp_{t-2}|$ then
 //promote exploration
 if lattConf = n(1 ×n/2) then
 lattConf =(n×n);
 else if lattConf=(n×n) then
 lattConf =4 (n/2×n/2);
 else if lattConf=4(n/2×n/2) then
 lattConf =2(n/2×n);
 else
 lattConf =2(n/2×n);
 end if
else
end if
end procedure
```

9. DES Algorithm

The **Data Encryption Standard** is a block cipher, meaning a cryptographic key and algorithm are applied to a block of data simultaneously rather than one bit at a time. To encrypt a plaintext message, **DES** groups it into 64-bit blocks.

Step 1: Create 16 subkeys, each of which is 48-bits long. The 64-bit key is permuted according to the following table, PC-1. ...
Step 2: Encode each 64-bit block of data.

```
Cipher (plainBlock[64], RoundKeys[16, 48],
cipherBlock[64])
{ permute (64, 64, plainBlock, inBlock,
InitialPermutationTable)
 split (64, 32, inBlock, leftBlock, rightBlock)
for (round = 1 to 16) { mixer (leftBlock, rightBlock,
RoundKeys[round])
if (round!=16) swapper (leftBlock, rightBlock)
 }
combine (32, 64, leftBlock, rightBlock, outBlock)
permute (64, 64, outBlock, cipherBlock,
FinalPermutationTable)
 }
mixer (leftBlock[48], rightBlock[48], RoundKey[48])
{ copy (32, rightBlock, T1)
function (T1, RoundKey, T2)
 exclusiveOr (32, leftBlock, T2, T3)
copy (32, T3, rightBlock)
 }
 swapper (leftBlock[32], rigthBlock[32])
 {
copy (32, leftBlock, T)
 copy (32, rightBlock, leftBlock)
 copy (32, T, rightBlock)
}
 function (inBlock[32], RoundKey[48], outBlock[32])
{
 permute (32, 48, inBlock, T1,
ExpansionPermutationTable)
exclusiveOr (48, T1, RoundKey, T2)
 substitute (T2, T3, SubstituteTables)
permute (32, 32, T3, outBlock, StraightPermutationTable)
}
 substitute (inBlock[32], outBlock[48],
SubstitutionTables[8, 4, 16])
{
 for (i = 1 to 8)
{
row ←2 *inBlock[i* 6 + 1] + inBlock [i *6 + 6]
col ←8 * inBlock[i *6 + 2] + 4 * inBlock[i *6 + 3] + 2 *
inBlock[i *6 + 4] + inBlock[i *6 + 5]
value = SubstitutionTables [i][row][col]
 outBlock[[i *4 + 1] ←value / 8;
 value ←value mod 8 outBlock[[i *4 + 2] ←value / 4;
value ←value mod 4 outBlock[[i * 4 + 3]← value / 2;
value ← value mod 2 outBlock[[i * 4 + 4]← value } }
```

Key generation:

```
Key_Generator (keyWithParities[64], RoundKeys[16, 48], ShiftTable[16])
{
 permute (64, 56, keyWithParities, cipherKey, ParityDropTable)
 split (56, 28, cipherKey, leftKey, rightKey)
for (round = 1 to 16)
{
 shiftLeft (leftKey, ShiftTable[round])
shiftLeft (rightKey, ShiftTable[round])
combine (28, 56, leftKey, rightKey, preRoundKey)
 permute (56, 48, preRoundKey, RoundKeys[round], KeyCompressionTable)
 }
 }
 shiftLeft (block[28], numOfShifts)
{
 for (i = 1 to numOfShifts)
 {
 T← block[1]
for (j = 2 to 28)
 {
 block [j−1] ← block [j]
}
 block[28]← T
}
 }
```

Comparison using the existing project:

Existing Project

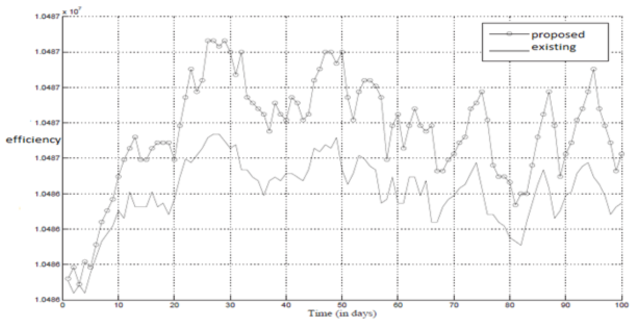
1. The existing system may not effectively find data duplication because that only effect data encryption under same key so it lead to security flaw.
2. The threat now extends to our personal lives: plethora of personal information is available to social networks and smart phone providers and is indirectly transferred to untrustworthy third party and fourth party applications.
3. The data will stored in cloud and no separate multipart data is created.

Proposed Concepts:

1. The proposed concepts uses one time password so the key keep on changing ("Dynamically Generate Encryption key ") and by this the security increases
2. Here the proof of ownership is given only to data owners who first upload the file and only the data owners can share or transfer file to the user so by this way the messages will not be unwontedly leaked.
3. That data is sent into multi part and generate a signature key for each part, key based on part content so we can easy to identify data duplication.

10. Efficiency

The comparative results for the theoretical efficiency of the scheme are summarized in below graph the analysis results of each scheme in terms of the communication and storage overhead are shown. For communication overhead, "upload message size" represents the communication cost required for the data outsourcing process; "download message size" represent the communication cost required for ciphertext downloading and tag checking processes, and "rekeying message size" represents the communication cost required for rekeying the data encryption key. For storage overhead, "key size" and "tag size" represent the size of the keys and tag information that each owner needs to store, respectively. For the upload and download message sizes, the proposed scheme is the same as the basic RCE scheme. In LR, the communication overhead for verifying PoW is additionally included in the download message. In the scheme, the PoW verification and tag checking processes are done during the data upload the phase by the data owner.

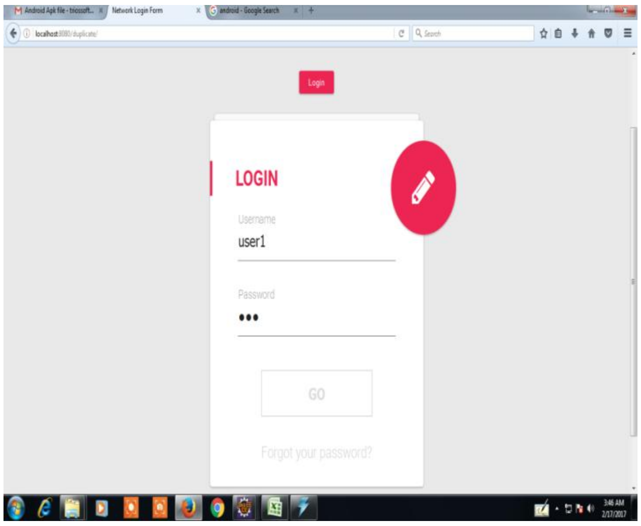


However, they can be executed during the data download phase without loss of functionality and efficiency. Thus, we suppose they are executed during the download phase as in and the proposed scheme for the sake of fair comparison. With regard to the rekeying message size, only the proposed scheme supports key updates upon ownership changes for data. In the proposed scheme, the rekeying message size (i.e., size of C3 i) would be (n−m)log n n−m Ck. This additional message plays an important role in enhancing the backward and forward secrecy, and enforces fine-grained user access control to the outsourced data in contrast to the other schemes. Whereas, the encryption key is determined by the message itself, it is selected by the initial uploader and never updated during the lifetime of the data in the system. Thus, even if the other schemes do not need the additional rekeying messages, they cannot guarantee the data privacy during the windows of vulnerability in the practical cloud environment where the ownership changes dynamically as time elapses. The above graph clearly shows the variation between the before paper and the current paper.

11. Results and Discussion

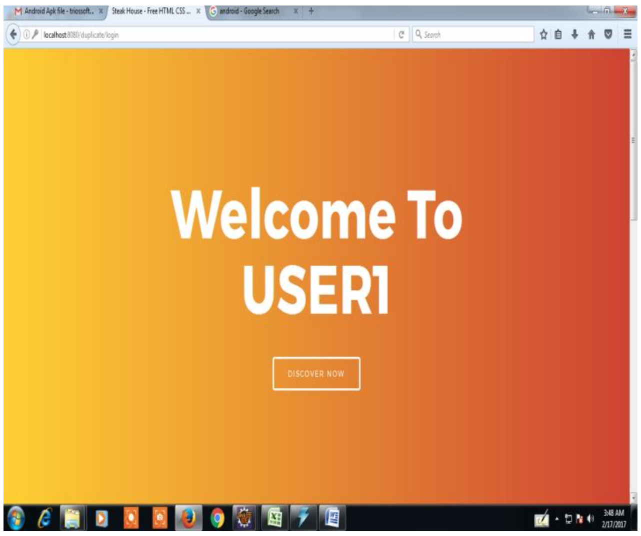
The goal of the proposed system is to increase security to the data and avoid duplication. Here we use a message digest technique to avoid the duplication of the file. The message digest technique generates a hash function by using that hash function the data checker can easily find the duplicate data because for the same file the generated hash function will also be same. So it is easy to find the duplication. By finding the duplication we can increase the storage and efficiency. We use a data encryption standard for encrypting, decrypting and key generation. The encryption and decryption process uses a 64 bit key so if the attacker gets the file the key cannot be identified and the data will be kept safe. And for the key generation it performs the binary rotation operation. This binary rotation keeps on moving the key so the key cannot be hacked. The data owner provides will provide the one time password to the other user. So the other users are requested to view the file only once and they don't have authorization to share that particular file to third party applications. So therefore our project is more efficient and secure.

User 1 login form:



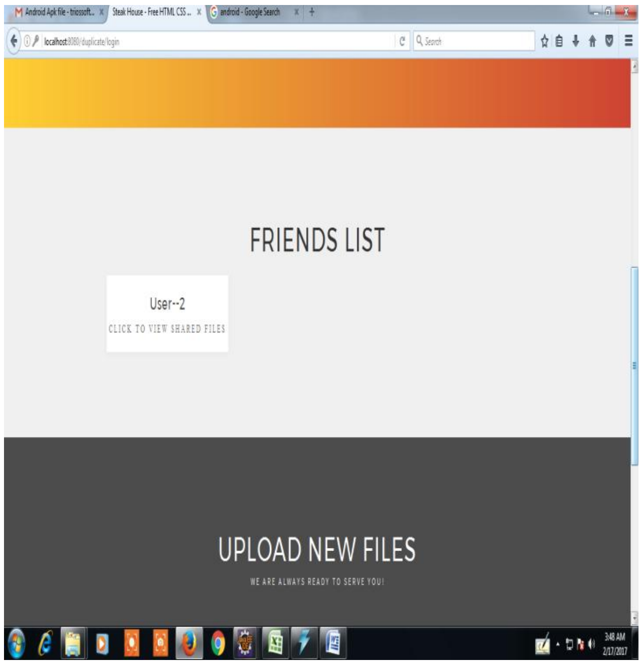
Here the user1 enters into the form and creates the profile.

User1 profile:



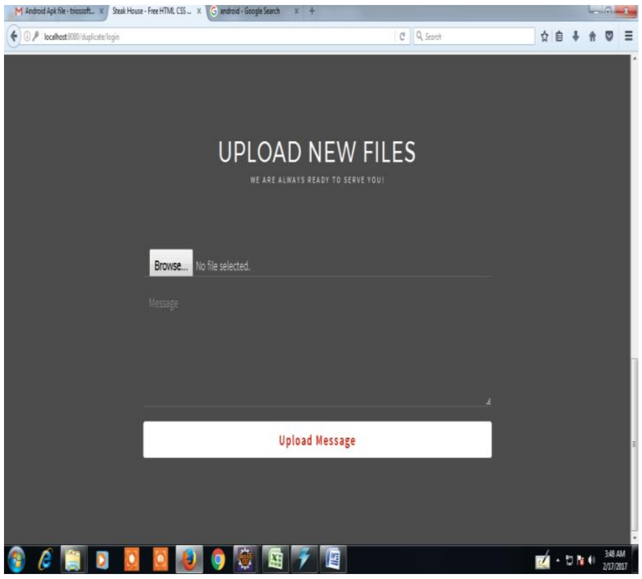
In this profile the user can either upload a files and he can view the other users uploaded file

User1 view others shared file



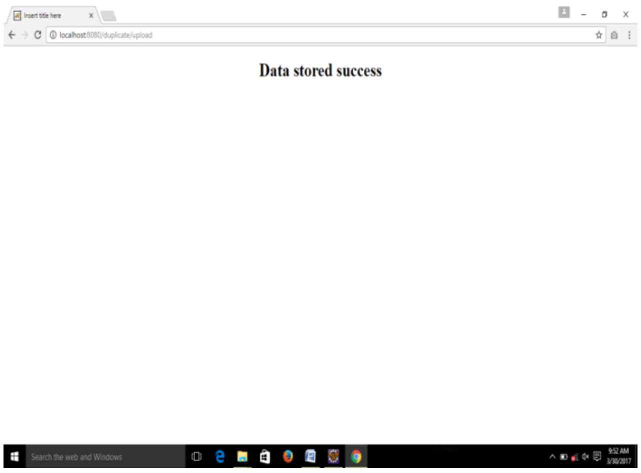
In this the user1 can view the other users shared file that will be in the encrypted form

User1 upload profile


UPLOAD NEW FILES
WE ARE ALWAYS READY TO SERVE YOU!
Browse... No file selected.
Message
Upload Message

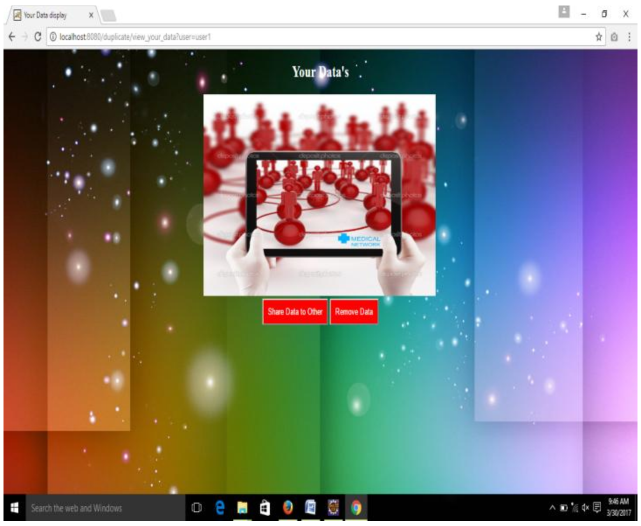
Here user1 should upload his file that includes a message and the file will be in the form of a image

User 1 storage details



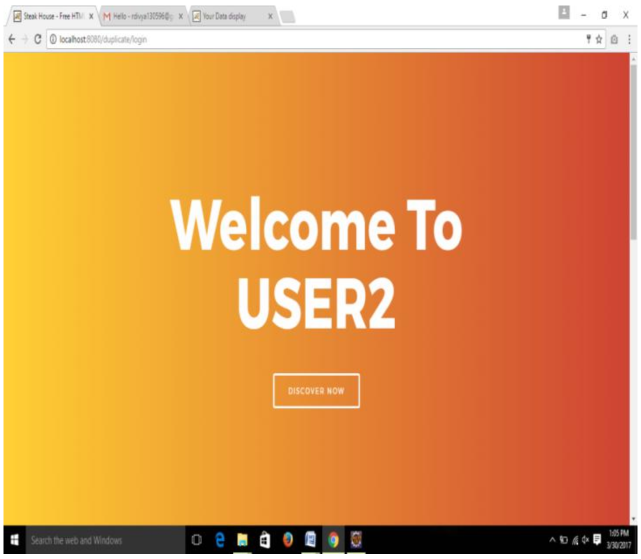
Here the data checker checks the file and verify the duplication. If the duplication does not exists then the data will be stored successfully

Viewing user1's upload file



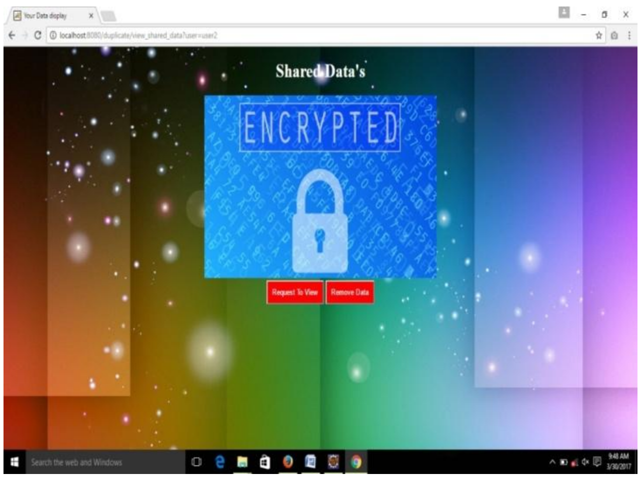
Here user1 will view his uploaded files and shares the file.

User2 profile



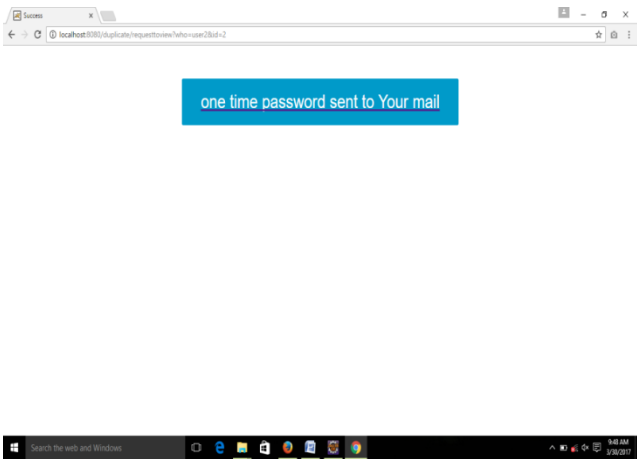
In this profile the user2 can either upload a files and he can view the other users uploaded file

User2 viewing the encrypted data



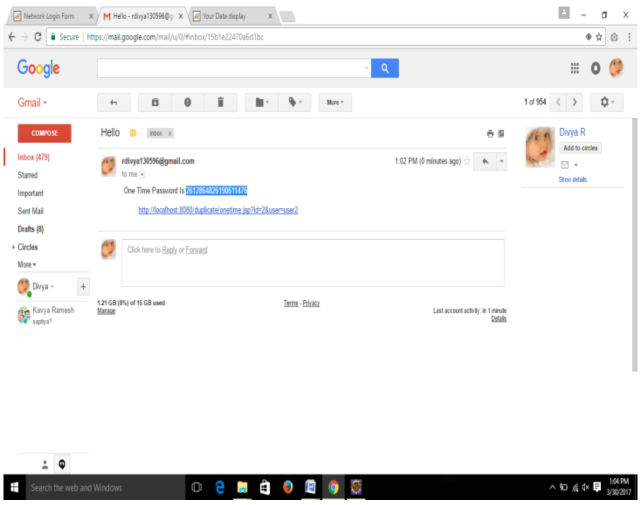
Here the user2 will view the encrypted data and send the request to view the file.

OTP request


localhost:8080/duplicate/requesttoview?who=user2&id=2
one time password sent to Your mail

By clicking the link the otp will be sent to the user that is used to view the uploaded file.

Viewing OTP



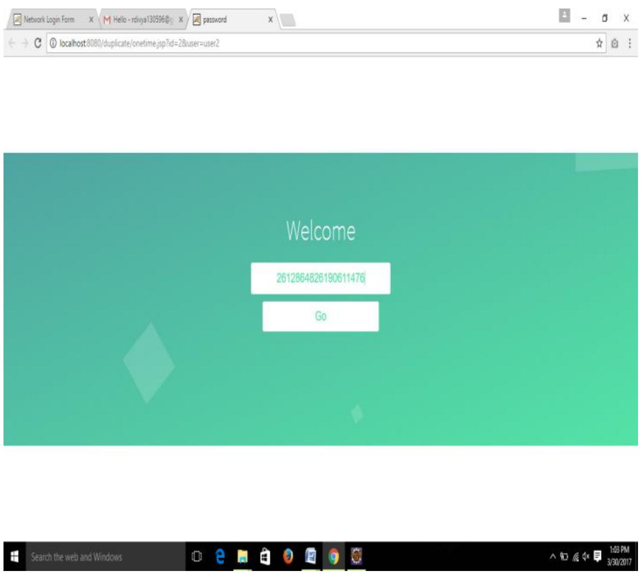
rdivya130596@gmail.com
to me
1:02 PM (0 minutes ago)

One Time Password Is:2612864826190611476

http://localhost:8080/duplicate/onetime.jsp?id=2&user=user2

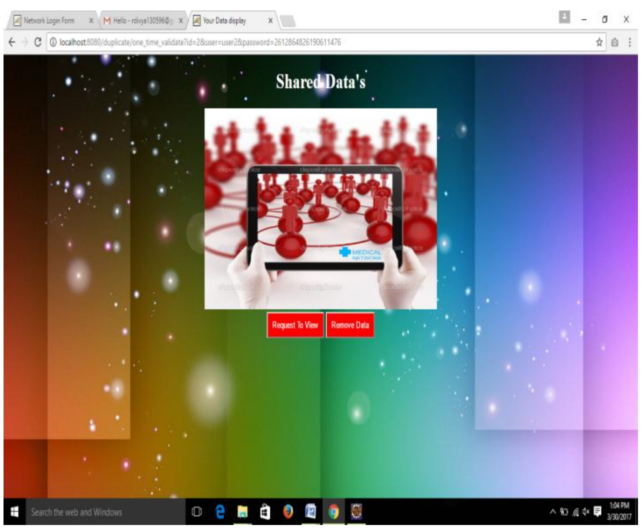
Here the user will view the one time password and by clicking the link they can view the message

OTP profile



Here the user will enter into the OTP profile and view the file.

User2 viewing the encrypted file


localhost:8080/duplicate/one_time_validate?id=2&user=user2&password=2612864826190611476
Shared Data's
Request To View
Remove Data

In this the user2 views the correct file from the encrypted form using the one time password and this user can only view the file and cannot share the files to others.

12. Conclusion

Dynamic ownership management is an important and challenging issue in secure deduplication overencrypted data in cloud storage. In this study, we proposed a novel secure data deduplication scheme to enhance a fine-grained ownership management by exploiting the characteristic of the cloud data management system. The proposed scheme features a reencryption technique that enables dynamic updates upon any ownership changes in the cloud storage. Whenever an ownership change occurs in the ownership group of outsourced data, the data are reencrypted with an immediately updated ownership group key, which is securely delivered only to the valid owners. Thus, the proposed scheme enhances data privacy and confidentiality in cloud storage against any users who do not have valid ownership of the data, as well as against an honest-but-curious cloudserver. Tag consistency is also guaranteed, while the scheme allows full advantage to be taken of efficient data deduplication over encrypted data. In terms of the communication cost, the proposed scheme is more efficient than the previous schemes, while in terms of the computation cost, taking additional 0.1−0.2 ms compared to the RCE scheme, which is negligible in practice. Therefore, the proposed scheme achieves more secure and fine-grained ownership management in cloud storage for secure and efficient data deduplication.

Reference

[1] J. Xu, E. Chang, and J. Zhou, “Leakage-resilient client-side deduplication of encrypted data in cloud storage,” ePrint, IACR, http://eprint.iacr.org/2011/538.
[2] M. Bellare, S. Keelveedhi, and T. Ristenpart, “Message-locked encryption and secure deduplication,” Proc. Eurocrypt 2013, LNCS 7881, pp. 296–312, 2013 Cryptology ePrint Archive, Report 2012/631, 2012
[3] S. Halevi, D, Harnik, B. Pinkas, and A. Shulman-Peleg, “Proofs of ownership in remote storage systems,” Proc. ACM Conference on Computer and Communications Security, pp. 491–500, 2011
[4] M. Mulazzani, S. Schrittwieser, M. Leithner, and M. Huber, “Dark clouds on the horizon: using cloud storage as attack vector and online slack space,” Proc. USENIX Conference on Security, 2011
[5] A. Juels, and B. S. Kaliski, “PORs: Proofs of retrievability for large files,” Proc. ACM Conference on Computer and Communications Security, pp. 584–597, 2007.
[6] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable data possession at untrusted stores,” Proc. ACM Conference on Computer and Communications Security, pp. 598–609, 2007