A Hybrid Cloud Approach for Deduplication with Attribute-Based Encryption

Supriya Kalkote, Jibi Abraham

Abstract*: Data deduplication is a special type of data compression technique to eliminate duplicate copies of iterating data in storage which saves storage space and bandwidth. In traditional deduplication system, hybrid cloud is introduced for implementing authorized deduplication. As public cloud cannot be fully trusted, the data is protected by Convergent Encryption. Only the user having a hash of the data can decrypt data from the cloud in a traditional system. In this paper, we present a deduplication scheme combined with Attribute-Based Encryption. In this technique, the data from the server can be accessed by the legitimate users if out of n, any k attributes related to the data are correct. We provide an implementation of our system and give performance measurements of the traditional system with respect to our proposed system.

Index Terms: Attribute-Based Encryption, Convergent Encryption, Deduplication.

I. INTRODUCTION

As cloud computing becomes predominant, one serious challenge of the cloud storage services is the management of the increasing volume of the data. Instead of storing multiple data copies with the same content, deduplication eliminates duplicate data by keeping only one physical copy. In a system without deduplication mechanism, a complete data copy is sent to the public cloud and then after, duplication is checked which results in network bandwidth consumption which is directly related to the size of the data copy. Instead of that in deduplication technique only hash of the data copy is sent to the public cloud which reduces the number of bytes that have to be sent to the public cloud. Hence deduplication saves storage space and also network bandwidth. Privacy and security concerns arise because users' sensitive data which is out-sourced to the public cloud are susceptible to both insider and outsider attacks. Traditional encryption techniques require different users to encrypt same data with unique private keys, which will lead to in data copies of different users forming different ciphertexts making deduplication almost impossible. The work proposed in [5] uses a technique named Convergent Encryption(*CE*) in which the encryption key is formed from the data contents after applying the cryptographic hash function, which will result in producing the same ciphertext every time for the same data contents. In this technique, the data can be decrypted only by the users who possess the encryption key, which is nothing but the hash of the data contents. Hence, in the models of applications where the owner of the data encrypts and stores the ciphertext on a cloud, the encryption key can be shared with the data users in two ways. Firstly, the owner stores the encryption key and securely shares the key with the data users whenever required. Secondly, the data owner is aware of the list of data users and shares the encryption key immediately with them. In both cases, there is a high possibility of data leakage with respect to storage and sharing. Our work focuses on a model of applications in which after deduplication validation, the data is stored on a cloud in encrypted form. To perform the encryption-decryption, our work makes use of Attribute-Based Encryption(*ABE*), which focuses on a list of attributes of the data [10]. The data owner uses these attributes to encrypt the data and any legitimate user who has knowledge about a few of these attributes can able to decrypt the data. Experimentation is conducted with different sizes of data to compare various performance parameters between the work in [5] which uses *CE* and our work which uses *ABE*. The rest of the paper is organized as follows. Section 2 gives a brief overview of related works. Section 3 introduces our proposed scheme. Section 4 explains a detailed implementation of our scheme, followed by performance evaluation in Section 5. Finally, a conclusion is presented in the last section.

II. LITERATURE REVIEW

Dropbox [1], Google and many more service providers perform deduplication which saves space by storing only one copy of each file uploaded. In traditional encryption [12], a key used for encryption is different per user. This results in completely different encrypted data though the contents are same, hence making deduplication impossible. Message-Locked Encryption(*MLE*) [2] uses concept named as Convergent Encryption(*CE*) which was introduced by Douceur et al. [3] and [4]. In the implementation of *CE* let *M* be a file's data. Cryptographic hash function *H* is applied to the data *M*. The result of this function is an encryption key $K = H(M)$. With this key, ciphertext is computed $C \leftarrow (K, M)$ via a deterministic symmetric encryption scheme. When another user comes with the same file for storing, the resultant cryptographic hash will be the same which produces the same encryption key, hence, making deduplication possible. However, *CE* has limitation to offline brute-force dictionary attacks [6] and [7]. If a user is not a legitimate user, but if he has the hash of the file, then, attack of manipulation of data identifier is possible. To avoid this problem, Proofs of Ownership(*PoW*) practical implementation is first introduced by Halevi et al. [7]. This uses Merkle tree for deduplication, which checks deduplication at client-side. Only when all Merkle tree paths are valid, the verifier accepts the proof. Meye et al. proposed another scheme to adopt two servers for intra-user deduplication and inter-deduplication [8].

Revised Manuscript Received on May 28, 2019.

Supriya Kalkote, Dept. Of Computer engineering, College Of Engineering, Pune, Maharashtra, India. kalkote.supriya@gmail.com

Jibi Abraham, Dept. Of Computer engineering, College Of Engineering, Pune, Maharashtra, India. ja.comp@coep.ac.in

The user key is used to encrypt the ciphertext *C* of *CE* and then transferred to the servers ClouDedup [9] tries to cope with the security exposures of *CE* using data deletion. But data deletion has its certain issues and ClouDedup [9] cannot address these issues. Like a data holder who removed the data from the cloud can still access the data because the encryption key is known. If data is present in the cloud then with encryption key he can decrypt it.

There can be many internal and external attacks on public cloud and hence the data owners may not trust the public cloud. "Hybrid Cloud Approach for Secure Authorized Deduplication [5]" scheme introduces a new entity named as the private cloud which acts as an execution interface between data owner and public cloud to provide an access control to a user. A token is formed from the hash of a user access privileges together [8] with the data to be uploaded by the owner and the data along with token is stored in the public cloud. When another owner with the same access privileges tries to upload the same file, results in generating the same file token which leads to deduplication. The token generation and user access privileges are managed by the private cloud.

III. PROPOSED SYSTEM

To device, an application with secure deduplication features, a file server implementation is considered in which every file is unique with respect to its contents, is encrypted based on the file attributes and which can be decrypted by a user having appropriate minimum number of attributes with valid access control.

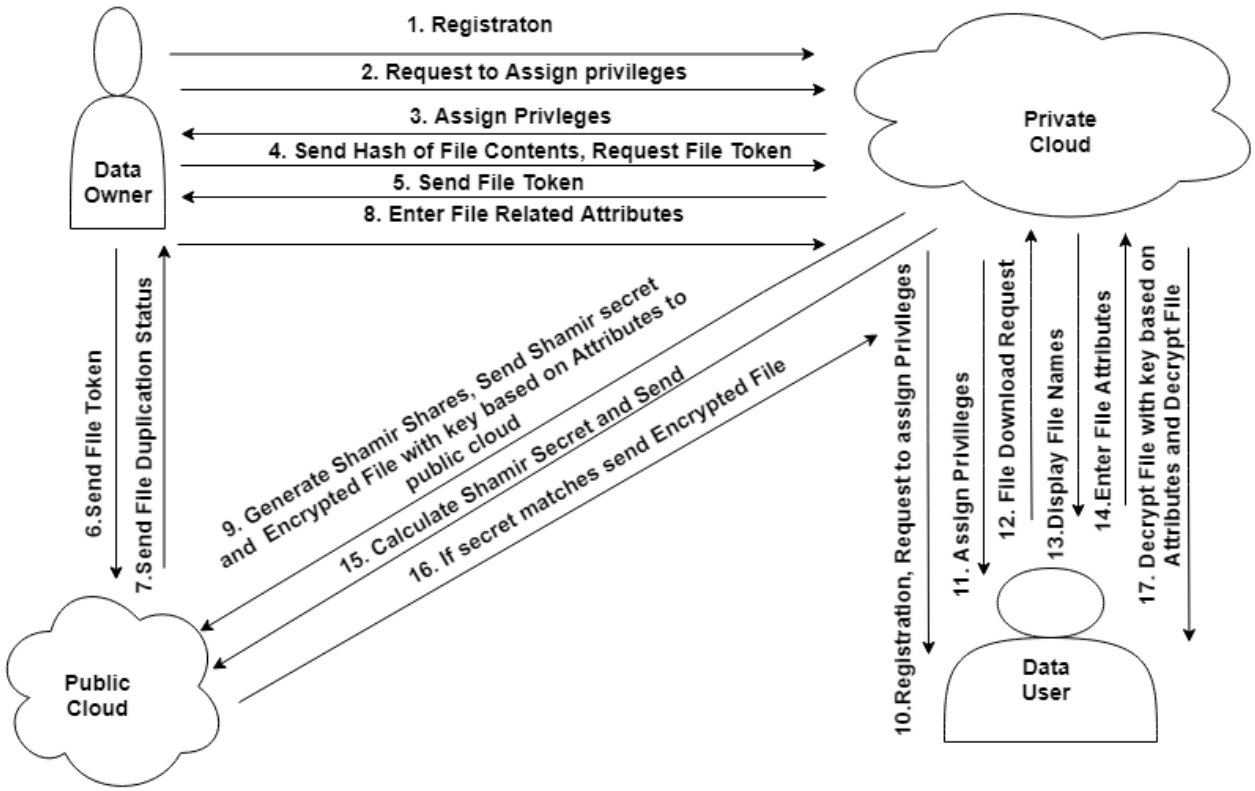

Fig.1 System overview.

Other applications like student record administration system, Audio-video Download System can also be thought of in similar lines. The *ABE* based system permits any authorized user to decrypt the data if he has *k* correct attributes out of *n* correct attributes.

As an example, student record administration system can be considered where unique student's reports are stored in a public cloud in an encrypted form. In this case, student's name, roll number, date of birth and so forth can be utilized as attributes to encode the student's reports.

Public cloud provides service of file server management by using cost effective encrypted file contents storage with deduplication.

Private cloud acts as an interface between a user and the public cloud since public cloud is not fully trusted. It deals with all transactions in a secure manner.

Data owner is a user, having assigned a set of access privileges. Each file is outsourced to the public cloud by encrypting Attribute-Based Encryption.

Data User is authorized user who can decrypt and download file contents from the public cloud by providing sufficient attributes.

There are 4 major entities present in the proposed system which is shown in Fig.1. In line with [5], steps 1-7 from Fig.1 perform registration and duplication check. Data owner and data user are enrolled in a private cloud and private cloud doles out access rights to them. Each access right has been assigned a randomly generated key named as Privilege key. File token is formed by *SHA256*(File contents, Privilege key) and it enables the deduplication check. Initially, data owner needs to get a file token from the private cloud. In the next step, public cloud receives a file token from the data owner, verifies it and sends duplication status to the data owner.

After deduplication validation, an encryption key is formed from the attributes of the file and *AES* encrypted file is outsourced to the public cloud(Step 7-17) from Fig.1.

Private cloud implements *ABE* which uses Shamir secret sharing scheme where a secret which is nothing but a randomly generated number is divided into parts, giving each attribute its own unique part. Some of the parts or all of them are needed in order to reconstruct the secret.

To download a file, say *file1.txt*, a data user is required to provide *k* correct attributes out of *n*. Based on these attributes, a decryption key, Shamir secret say *s*, is generated dynamically by the private cloud. Private cloud passes secret *s* along with a filename to the public cloud. Upon download request, an encrypted file *file1.txt* is sent to the private cloud provided *s* sent by the private cloud matches secret stored at public cloud for *file1.txt*. Private cloud provides decrypted file to the data user.

A. Implementation

The proposed model is implemented in Java which uses JDK and Eclipse for execution of the code and Amazon EC2 Cloud for the private and public cloud setup. We have used, Amazon Simple Notification Service(*SNS*) for inter-cloud communication. Amazon SNS is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients. Any SQL server can be used for attributes storage.

Method of implementation: In the Data user/owner registration module, *generateNextID*() method generates an auto-incremented unique id for the data owner/user. Inter- cloud communication is done with method *SNS.send*(*String arg0*, *String arg1*, *String arg2*, *String arg3*, *String arg4*) and *SNS.Receive*().

Polynomial initialization: *ABE* uses Shamir secret sharing scheme where out of *n*, any *k* attributes are required for data decryption. For dividing secret into *n* parts with the threshold of k parts, the polynomial degree will be *k - 1*. Initialization of polynomial in equation 1 is shown in algorithm 1.

$$f(x) = (a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_{k-1} x^{k-1}) \qquad (1)$$

Algorithm 1 Polynomial initialization for secret shares generation
Input Parameters: BigInteger *secret*, threshold *k*, total share created *n*, BigInteger *prime*, random number *r*
Output: $Coeff[]$ an array of polynomial coefficient

$Coeff[0] \leftarrow \text{sec}\,ret$
for *i* =0 to *k* - 1
r = *random*();
if(*r* > 0 and *r* < *prime*)
$Coeff[i] \leftarrow r$
else
break;

Secret shares calculation: Secret shares corresponding to each attribute are calculated with polynomial in equation (1) as shown in Algorithm 2.

Encryption key generation: A file outsourced to a public cloud has to be *AES* encrypted for which 128 bits encryption-decryption key is generated based on attributes as shown in algorithm 3. For file download, the file attributes are provided by the data user and the corresponding Shamir shares say (x_0, y_0), (x_1, y_1) and so on are fetched from SQL server. Private cloud computes, a Lagrange basis polynomial in equation (2)

$$l_j(x) = \prod_{0 \le m \le k} \frac{x - x_m}{x_j - x_m} \qquad (2)$$

by using shares (x_0, y_0),(x_1, y_1) ...(x_k, y_k). Value of $l_j(x)$ is used, to compute a secret by using equation (3).

$$f(x) = \sum_{j=0}^{k} y_j . l_j(x) \qquad (3)$$

Algorithm 2 Shamir secret shares generation.
Input Parameters*:* BigInteger *secret*, threshold *k*, total share created *n*, BigInteger *prime*, random number *r*
Output: Shamir secret shares

for *i* = 0 to *n*
$Coeff[0] \leftarrow \text{sec}\,rete$
for *j* = 0 to *k* - 1
x = x.*add*(*coeff*[*k*].i^k)%*prime*
return *shares*;

Algorithm 3 Encryption-decryption key generation
Input Parameters: *S*[] which is the set of attributes for File.
Output: Encryption-decryption key.

1. User provides the *m* attributes *a* about the file.
2. for each attribute

$$P = \sum_{k=0}^{n} \text{concatenate } a_k \text{ with } P \qquad (4)$$

3. Convert array *P* into 64 bit String encoder text.

B. Performance Measurements

The system is thoroughly tested to evaluate the performance of *CE* and *ABE*. File size ranging from 100kb to 800kb is taken as a dataset. File upload and download operation times are measured in milliseconds for *CE* and *ABE* respectively. We have observed that *ABE* has 41 % extra overhead over *CE* in a file upload time and 49 % extra overhead over *CE* in a file download time as shown in Fig.2 and Fig.3 respectively. We have also observed that time required for a file upload and download operations increases with more number of attributes as shown in Fig.4.

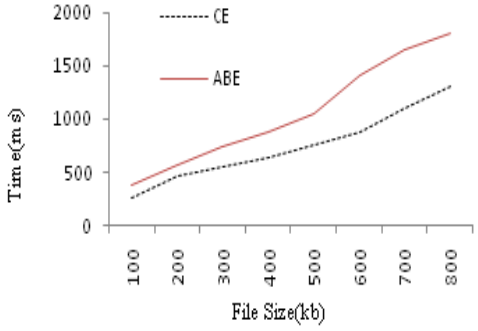

Fig.2 *CE* and *ABE* file upload time comparison.

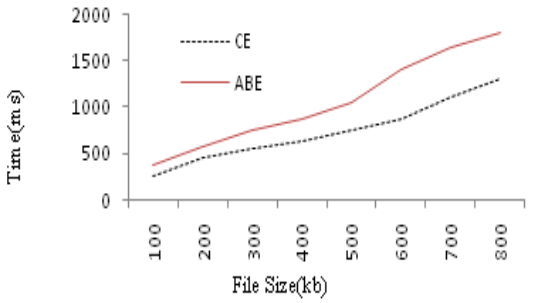

Fig.3 *CE* and *ABE* file download time comparison.

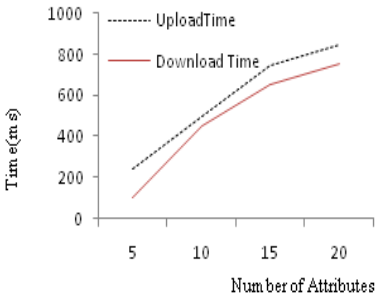

Fig.4 File upload and download time with varying number of attributes.

Though the *ABE* has cost overhead over *CE*, but we eliminated the drawback from traditional system that file decryption can only be done by the data users having hash of the file contents. In our system, a file download can be done by any data user having known *k* out of *n* attributes hence achieves one to many decryption, with some additional overheads.

IV. CONCLUSION

Data deduplication saves storage space and network bandwidth. Also for the data security over the cloud, data needs to be stored in an encrypted format. In this work, the notion of authorized data deduplication with Attribute-Based Encryption is proposed. As a proof of concept, we implemented our proposed authorized duplicate check scheme with *ABE* and evaluated performance of *ABE* over *CE*. We showed that our scheme eliminated drawbacks of *CE* with some extra overhead.

REFERENCES

1. Dropbox, A file storage and sharing service(2016). LNCS, vol. 9999, pp. 113. Springer, Heidelberg (2016). http://www.dropbox.com.
2. J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, Reclaiming space from duplicate files in a serverless distributed file system, in Proceeding IEEE International Conference Distributed Computing System, 2002, pp.617624, doi:10.1109/ICDCS.2002.1022312. Author, F.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 12. Publisher, Location (2010).
3. G. Wallace, et al., Characteristics of backup workloads in production systems, in Proc. USENIX Conf. File Storage Technol., 2012, pp. 116.
4. Z.O.Wilcox,Convergentencryptionreconsidered,2011. http://www.mailarchive.com/cryptography@metzdowd.com/msg08949.html
5. Jin Li, Yan Kit Li, Xiaofeng Chen, Patrick P.C. Lee, and Wenjing Lou A Hybrid Cloud Approach for Secure Authorized Deduplication IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 26, NO. 5, MAY 2015. Pp-1206-1216
6. S. Keelveedhi, T. Risterpart and M.Bellare "Message-locked encryption and secure de-duplication in proceeding of Eurocrypt 2013", pp. 296-312, doi: 10.1007 n 978-3-642-38348-9 18.
7. A. Shulman-Peleg, B. Pinkas, D. Harnik and S. Halevi "Proofs of ownership in remote storage systems", in proceeding 18th ACM Conference Computer Commun. Secur., 2011, pp. 491500, doi:10.1145/ 2046707.2046765.
8. P. Raipin,P. Meye, F. Tronel, and E. Anceaume, "A secure two phase data deduplication scheme", pp. 802809, doi:10.1109/HPCC.2014.134 in proceeding HPCC/CSS/ICESS, 2014.
9. M. Onen, S. Loureiro, P. Puzio, R. Molva, "ClouDedup: Secure deduplication with encrypted data for cloud storage", pp. 363370, doi:10.1109/CloudCom.2013.54 in proceeding IEEE International Conference Cloud Computing Technology Science, 2013.
10. B. Waters, J. Bethencourt, A. Sahai, "Ciphertext-Policy Attribute-Based En-cryption" Pp: 321-334, Washington, DC, USA, 2007. IEEE Computer Society. In proceeding of the 2007 IEEE Symposium on Security and Privacy SP '07.
11. P.Samarati , S.Foresti, D.Vimercati, S.Jajodia, S. Parabosch and "Over-encryption: management of access control evolution on outsourced data" Pp: 123-134 in VLDB 2007
12. D. Richard Kuhn, David F. Ferraiolo "Role-Based Access Controls" Reprinted from 15th National Computer Security Conference (1992), Baltimore MD pp. 554563

AUTHORS PROFILE

Supriya Kalkote
Qualifications:
M-Tech (Computer Engineering)
B-Tech (Computer Science & Engineering)
Industry Experience: 3 years
Work area- Data warehouse.

Dr. Jibi Abraham
Qualifications:
PhD (Computer & Information Sciences Engineering)
MS (Software Systems)
B.Tech (Computer Science & Engineering)

Research work:
International Journals - 11
Book Chapters - 04
International Conferences held abroad - 07
International Conferences held in India - 37
National Conferences - 05