Secure and low Cost Encrypted Data Sharing in Cloud

[1]Ms. Minimol Mathew, [2]Mrs. Sumathi D., [3]Ms. Ranjima P., [4]Ms. Anisha Viswan

[1,3,4] PG Scholar,[2]Assistant Professor,
[1,2,3,4] Computer Science and Engineering, PPG Institute of Technology, Coimbatore, 641035, India

***ABSTRACT*-Cloud computing is used to store and share data by anyone from anywhere in the world. It provides scalable and on demand self services. The cloud data needs maximum security at rest state as well as at transit state. Cloud service providers provide security to the data in the cloud and the data owner also uses some cryptographic techniques for data security. Owners may use symmetric cryptography or asymmetric cryptography. In the symmetric key cryptographic technique, both data encryption and decryption use same key. But in the asymmetric key cryptography, use receiver's public key for encryption and private key for decryption. If data owner use same cryptographic key for all data files, then it cannot share the partial amount of data files. To overcome this problem, the owner using separate encryption/decryption keys for each file in the cloud. During the data sharing, secure key transformation is a difficult process. To share 'n' number of files, 'n' number of key should be transferred to partner. It will increase the network traffic and reduce security. To overcome these, the data owner can aggregate all decryption keys which want to share with partner, into a single key known as aggregate key. This aggregate key can decrypt subset of delegated ciphertext files.**

***Index terms*- Cloud storage, Security, Data encryption, Aggregate-key**

I. INTRODUCTION

Cloud computing is used to provide services rather than products such as resources, software, information. Cloud resources are allocated for multiple users simultaneously per demand. The National Institute of Standards and Technology's definition of cloud computing identifies five characteristics such as on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. Also, cloud service is provided in three models, namely, private, public, and hybrid. In public model, the cloud resource is available for every user but in private model the resources can access by only authenticated people. Hybrid cloud is the combination of private and public cloud.

Data storage and sharing is an important functionality of cloud computing. Data from different users are stored in different virtual machines, but reside on single physical machine. The virtual machine data may be attacked by an intruder [2]. To avoid this, data owners are using some cryptographic techniques. For example [3], with proven security relied on number-theoretic assumptions is more desirable, whenever the user is not perfectly trusting the security of the VM or the honesty of the technical staff. These users are encrypting their data with their own keys before uploading them to the server. It is shown in fig 1. For example, the data owner can upload encrypted data files which include both personnel and official files. The main issue is how to share the portion of encrypted files. One method is, owner can download file from storage, decrypt them and send to the partner, but it loses the value of cloud storage. Another method is, data owner provides the access rights to others and they can directly access data from cloud.

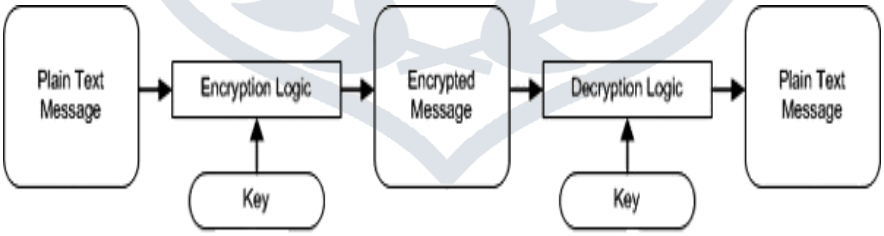

Fig 1: Data encryption

Let us assume Alice store her private files in cloud. She doesn't want to expose her files to everyone. Due to various security problems, Alice encrypts her files by using her own keys before uploading. Bob, Alice's friend, asks to share some of the files, which are kept in cloud. Alice can then use the share function of cloud provider, but the problem now is how to delegate the decryption rights for these files to Bob. The possible way is securely share the keys with Bob. Naturally there are two ways to encrypt the files,

- Alice can encrypt all files with single encryption key.
- Alice can encrypt each file with separate encryption keys.

Consider the case in which Alice is using single key for all files and then giving that key to Bob. Obviously, this method is inadequate since all private data can be view by Bob. For the second method, owner use number of keys for encryption and share all these keys with Bob. If Alice shares thousand files then, she has to send thousand keys. It will increase the network traffic and causes security problems.

The best solution for the above problem is that, Alice encrypts file with distinct encryption keys and only share single decryption key known as Aggregate key. This single decryption key can be sent via a secure channel and kept in secret. This small size single key is desirable. Usually secret key are stored in tamper - proof memory which is very expensive. This research mainly focuses on minimizing communication requirements like aggregate signature [4].

II. LITERATURE SURVEY

Cloud storage is used to store and share data effectively. Several methods are used for this purpose. Most common method is one to one key sharing method. In this method, data owner will generate distinct keys for each uploading files. These distinct keys will send to the partner based on the requirement. For example, the data owner having thousand files and partner requires hundred files. Then the owner should send corresponding hundred keys. During the key transformation, some security problems may arise.

The cryptographic key assignment schemes ([5], [6]) mainly focused on reducing the expense of storing and sharing the cryptographic keys. Consider tree structure for the key assignment, where, a key in a branch can be used to derive the keys of its decedent nodes. In this case, granting the parent key implicitly grants all the keys of its descendant nodes. Sandhu [7] proposed a scheme for generating a tree hierarchy of symmetric keys by using repeated evaluation of block ciphers. This can be generalized from tree structure to graph structure. Most of the modern key assignment schemes support both acyclic and cyclic graphs ([8], [9]). These schemes mainly generate keys for symmetric key cryptosystem. The public key cryptosystem is more expensive than symmetric key cryptosystem when the key assignment schemes used.

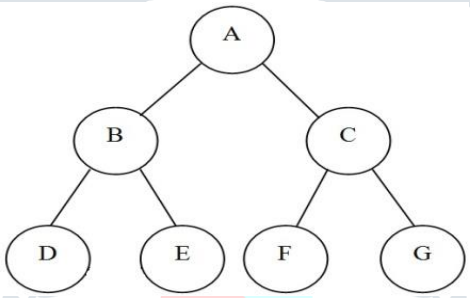

Fig 2: Key assignment for tree structure

Let us consider an example, Alice categorize the ciphertext classes based on some subjects. In fig 2, each node in the tree represents a secret key and the leaf nodes represent keys for each ciphertext classes. Every non leaf nodes derive keys for its descendant nodes. If Alice wants to share all files under category “B” then she only grant key for node B which automatically grant keys for all descendant nodes such as D and E. But this method is not an efficient method.

Identity based encryption (IBE) is a type of public key encryption. In this method the public key is considered as the identity of the user ([10], [11]). Here a trusted third party hold the master secret key and distribute private key to each cloud user with respect to the user identity. Guo tried to build IBE with key aggregation. These schemes assume random oracles [12]. Key aggregation means aggregation of different keys that are coming from different identity divisions. The key aggregation of secret keys and public keys at the expense of O(n) sizes for both ciphertexts and the public parameter, where n is the number of secret keys. In fuzzy IBE, one single aggregate key can decrypt ciphertext which are encrypted under many identities [13].

In many situations, data owner use separate access control for different users. Here they are using number of cryptographic keys. To provide appropriate keys to each user, owner generates some special schemes. These are low cost methods. Each data owner has a master key and all cryptographic keys are generating from this master key. Modular exponentiation is used to generate this scheme. To avoid attacks via master key, the factors of master key kept in secret.

III. KEY – AGGREGATE CRYPTOSYSTEM

In modern cryptography, a fundamental problem is about leveraging the secrecy of a small piece of knowledge into the ability to perform cryptographic functions multiple times. Here we study, how to make a decryption key more powerful in the sense that it allows decryption of multiple ciphertexts, without increasing its size. The problem statement is

“To design an efficient symmetric key encryption scheme which supports flexible delegation ie; a subset of the cipher texts (produced by the encryption scheme) is decryptable by a single decryption key.”

To solve this problem, we can introduce a new symmetric key encryption technique known as Key Aggregate Cryptosystem (KAC). In KAC, the data owner encrypts files with symmetric keys. Each file is having distinct key for encryption. During file sharing, the data owner will generate Aggregate Key and share that key with the partner. Aggregate key can be generated by aggregating all symmetric keys that the owner wants to share with partner, into a single key. This aggregate key is same size of an encryption key but, aggregates the power of many such keys, i.e., the decryption power for any subset of ciphertext.

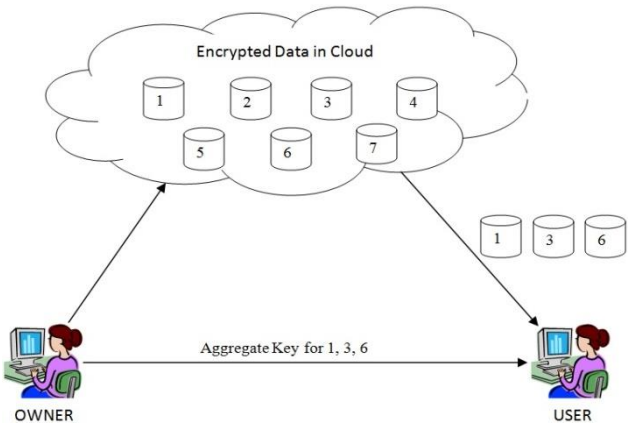

Fig 3: Owner shares files 1, 3, and 6 with partner by sending him a single aggregate key.

Data owner can send aggregate key to partner via a secure way. Partner can download the required files from cloud and decrypt this by using aggregate key. This is depicted in Fig 3.The size of encryption key, ciphertext and aggregate key are constant size. And also the size of encryption key and aggregate key are same.

A. Frame Work for Key Aggregate Cryptosystem

Aggregate key generation consist of four phases. Data owner can generate symmetric keys in ***KeyGen*** phase. Data can be encrypted by using these symmetric keys in ***Encrypt*** phase. Aggregate key can be generated by using n number of secret keys in ***Extract*** phase. In the ***Decrypt*** phase, the recipient can decrypt data by using aggregate key.

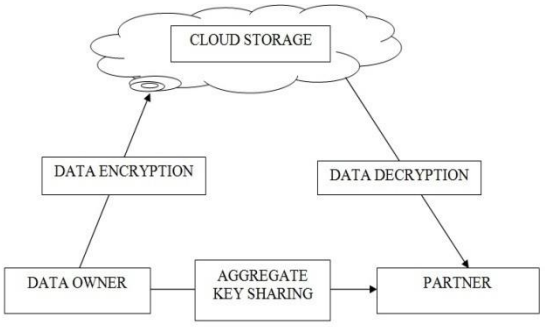

Fig 4: Architecture of Key Aggregate Cryptosystem

KeyGen: It is executed by data owner to randomly generate symmetric key *sk*.
Encrypt(sk,m): This phase is also executed by owner. Here inputs are symmetric key **sk** and message **m**. Its output is ciphertext C.
Extract(sk_i,i,S): In this phase, the input is **n** number of symmetric keys and set of indices of ciphertext **S**. Here the output is Aggregate key $\mathbf{K_S}$.
Decrypt(K_S,C,i,S):This phase is executed by delegatee who receives the aggregate key which is generated in Extract phase. By using the aggregate key, recipient can decrypt the data. Data decryption can be performed using the aggregate key

There are two functional requirements:

- **Correctness:** For any integer n, any set $S \subseteq \{1,2,\ldots n\}$, any index $i \in S$ and any message m,$Pr[Decrypt(K_S,C,i,S)=m, C \leftarrow Encrypt(sk,m), (sk_i) \leftarrow KeyGen, K_S \leftarrow Extract(sk_i,i,S)]=1$
- **Compactness:** For any integers n, any set S , any index $i \in S$ and any message m; $(sk_i) \leftarrow KeyGen$, $C \leftarrow Encrypt(sk,m)$, and $K_S \leftarrow Extract(sk_i,i,S)$; $|Ks|$, $|C|$ independent of number of ciphertext n.

IV. SYSTEM MODEL

Symmetric key cryptosystem is constructed for data sharing to the users in the cloud storage against unexpected privilege escalation. But the sharing of n number of keys is not desirable. It increases the transmission cost and network traffic. To avoid this situation, use a new technique known as Key-Aggregate Cryptosystem. In this technique, aggregate all keys that has to be shared with a partner into a single key. By using this key, partner can decrypt data. Aggregate key is also sent through a secured way. This key can be attacked by third party. Data owner wants to get assurance that the aggregate key can only be accessed by the original partner. For this purpose, Diffie-Hellman key exchange method can be used.

A. Creation of Cloud Environment

In cloud storage, user can upload, encrypt and also share data. The cloud storage is responsible for keeping the data available and accessible. Considering data privacy, a traditional way to ensure it is to rely on the server to enforce the access control after authentication. An automatic key generator distributes distinct keys for each uploaded data.

B. Designing Symmetric Key Encryption

To encrypt data, key generator provides distinct keys for each file. These encrypted files can only be decrypted by applying the same algorithm, and by using the matching private key. Both the sender and receiver share the same key to perform decryption. Symmetric key ciphers are implemented as either block ciphers or stream ciphers. The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are block cipher designs. In this phase, AES technique are used to encrypt messages.

AES is based on a design principle known as a substitution-permutation network. It is the combination of both substitution and permutation, and is fast in both software and hardware. Unlike DES, AES does not use a Feistel network. AES is a type of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. In contrast, the Rijndael works with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits. The key size specifies the number of repetitions of transformation rounds which convert the input, known as plaintext, into the cipher text.

C. Designing of Aggregate Key

Key aggregation is used for efficient data retrieval and to reduce network traffic. Data owner use the separate encryption/ decryption keys for each file in the cloud. Here to provide security and avoid network burden, the data owner can aggregate all decryption keys into a single key. Merkle-Damgard construction can be used to generate aggregate key. This aggregate key can decrypt any subset of delegated cipher text blocks.

Data owner can send single aggregate key instead of sending all secret keys via a secure method. Partner can decrypt shared files by using this single key. When using this method, both the network traffic and storage cost can be reduced. It is an efficient method to share the n number of secret keys.

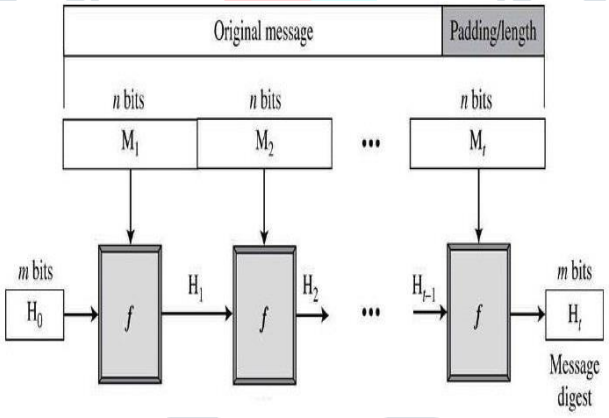

Fig 5: Merkle-Damgard construction

D. Secure Sharing of Aggregate Key

Diffie–Hellman key exchange (D–H) is a specific method of securely exchanging cryptographic keys over a public channel. It is an asymmetric key cryptographic method. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can be used to encrypt subsequent communications using a symmetric key cipher.

Diffie–Hellman establishes a shared secret that can be used for secret communications while exchanging data over a public network. The crucial part of the process is that sender and receiver exchange their secret value in a mix only. Finally this generates an identical key that is computationally difficult (impossible for modern supercomputers to do in a reasonable amount of time) to reverse for third party. Sender and receiver now use this common secret to encrypt and decrypt their sent and received data. Note that the starting prime number and base values are arbitrary, but is agreed on in advance by sender and receiver. These values are assumed to be known to any third party.

V. PERFORMANCE ANALYSIS

In the tree based key assignment, it supports 2^h key assignment when height of tree is h. That means, one key can assign 2^h keys. Consider *r* is the *delegation ratio*, which is the ratio of the delegated ciphertext classes to the total classes. n_a is the

number of keys assigned to access the ciphertext. If r=0, n_a also be zero ie; there is no ciphertext access. If r=1 then the ciphertext access is 100%.

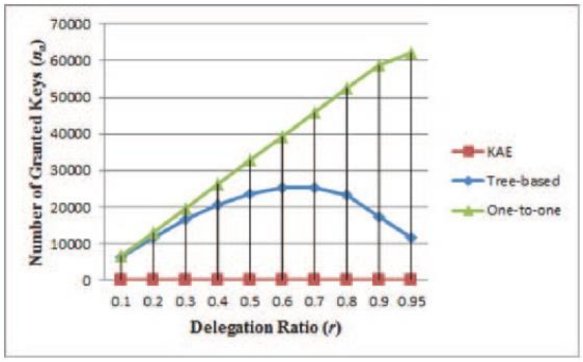

Fig 6: Key distribution comparison

Consider comparison between key assignments of the three methods such as one to one, tree based key assignment and Key Aggregate Cryptosystem in Fig 6. Here we can see that, the number of keys granted is equal to the number of delegated ciphertext in one to one key assignment. In tree structure scheme, we can reduce the number of granted keys according to the delegation ratio. But in the KAC system, the number of granted key is constant for any delegation ratio.

VI. CONCLUSION AND FUTURE WORK

Data owners in cloud storage use different methods to provide data security such as cryptographic techniques. In symmetric key cryptographic technique, effective and secure key transformation is difficult. So a new method, key- aggregate cryptosystem is used to provide effective key transformation. This method proposes the aggregation of any set of secret keys into a single key and the same is shared with the partner. By using this key, delegated set of ciphertext files can be decrypted but the other encrypted files outside the set remain confidential. This compact key can be conveniently sent to others or be stored in a smart card with very limited secure storage. In future, we can check the authenticity and integrity of data.

REFERENCES

[1] Cheng-Kang Chu, Sherman S.M. Chow, Wen-Guey Tzeng, Jianying Zhou, and Robert H. Deng, "Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage," in *Parallel And Distributed Systems*, VOL. 25, NO. 2, pp. 468-477, 2014.

[2] L. Hardesty, "Secure Computers Aren't so Secure". MIT press, http://www.physorg.com/ news176107396.html, 2009.

[3] S.S.M. Chow,C.-K. Chu,X. Huang,J. Zhou, and R.H. Deng, "Dynamic Secure Cloud Storage with Provenance," Cryptography and Security, pp. 442-464, Springer, 2012.

[4] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," Proc. 22nd Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '03), pp. 416-432, 2003.

[5] S.G. Akl and P.D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," ACM Trans. Computer Systems, vol. 1, no. 3, pp. 239-248, 1983.

[6] G.C. Chick and S.E. Tavares, "Flexible Access Control with Master Keys," Proc. Advances in Cryptology (CRYPTO '89), vol. 435, pp. 316-322, 1989.

[7] R.S. Sandhu, "Cryptographic Implementation of a Tree Hierarchy for Access Control," Information Processing Letters, vol. 27, no. 2, pp. 95-98, 1988.

[8] M.J. Atallah, M. Blanton, N. Fazio, and K.B. Frikken, "Dynamic and Efficient Key Management for Access Hierarchies," ACM Trans. Information and System Security, vol. 12, no. 3, pp. 18:1-18:43, 2009.

[9] Q. Zhang and Y. Wang,"A Centralized Key Management Scheme for Hierarchical Access Control," Proc. IEEE Global Telecomm. Conf. (GLOBECOM'04), pp. 2067-2071,2004.

[10] D. Boneh and M.K. Franklin, "Identity-Based Encryption from the Weil Pairing," Proc. Advances in Cryptology (CRYPTO '01), vol. 2139, pp. 213-229, 2001.

[11] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," Proc. 22nd Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '05), vol. 3494, pp. 457-473, 2005.

[12] S.S.M.Chow, Y. Dodis, Y. Rouselakis, and B. Waters, "Practical Leakage-Resilient Identity-Based Encryption from Simple Assumptions," Proc. ACM Conf. Computer and Comm. Security, pp. 152-161, 2010.

[13] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," Proc. 22nd Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '05), vol. 3494, pp. 457-473, 2005.

[14] T. Okamoto and K. Takashima, "Achieving Short Ciphertexts or Short Secret-Keys for Adaptively Secure General Inner-Product Encryption," Proc. 10th Int'l Conf. Cryptology and Network Security (CANS '11),pp.138-159, 2011.

[15] S.S.M. Chow, Y.J. He, L.C.K. Hui, and S.-M. Yiu, "SPICE – Simple Privacy - Preserving Identity-Management for Cloud Environment," Proc. 10th Int'l Conf. Applied Cryptography and Network Security (ACNS), vol. 7341, pp. 526-543, 2012.

[16] M. Chase and S.S.M. Chow, "Improving Privacy and Security in Multi-Authority Attribute-Based Encryption," Proc. ACM Conf. Computer and Comm. Security, pp. 121-130. 2009,
[17] T.H. Yuen, S.S.M. Chow, Y. Zhang, and S.M. Yiu, "Identity-Based Encryption Resilient to Continual Auxiliary Leakage," Proc. Advances in Cryptology Conf. (EUROCRYPT '12), vol. 7237, pp. 117-134, 2012.
[18] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," Proc. 22nd Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '03), pp. 416-432, 2003.
[19] C. Wang, S.S.M. Chow, Q. Wang, K. Ren, and W. Lou,"Privacy-Preserving Public Auditing for Secure Cloud Storage," IEEE Trans. Computers, vol. 62, no. 2, pp.362-375, Feb. 2013.
[20] B. Wang, M. Li, S.S. Chow, and H. Li, "Computing Encrypted Cloud Data Efficiently under Multiple Keys," Proc. IEEE Conf. Comm. and Network Security (CNS '13), pp. 90-99, 2013.