ISSN XXXX XXXX © 2016 IJESC

Research Article Volume 6 Issue No. 12

# Public-Key Encryption for Data Sharing over a Cloud by Implementing Key-Aggregate Cryptosystem

Prithviraj[1], Deekshith. K[2]
Assistant Professor[1, 2]
Department of CSE, SDMIT, India

**Abstract:**
Using cloud storage users can remotely store their data and enjoy the high quality services and applications from computing resources. Cloud provides a suitable infrastructure for storing and accessing our sensitive data. It also facilitates the service for storing and managing the data remotely through internet. The data can be effectively shared with other users over the cloud. The public-key cryptosystem is uses asymmetric key cryptography and also provides a better way of sharing of files. The secret keys can be aggregated to form a single key, called as aggregate key. We can say that, the data owner can release the constant-size aggregate key for the set of cipher text classes. User can have access to those set of files by using a single aggregate key. The files, which are outside the set, will remain confidential.

**Keywords**- Data sharing, public-key cryptosystem, Aggregate key, master secret key

## I. INTRODUCTION

One of the fundamental services offered by the cloud is data storage. Cloud storage allows the user to store their data online so that they can access the data from any location through internet. It also allows the user to store the data efficiently without worrying about the hardware management. Cloud storage is used as a core technology for many online services. With the wireless technology having more bandwidth user can access the data anywhere anytime they want. Data privacy is one of the important functionality in a cloud concept. There will be lack of security for the user's data over the cloud. Different client's data have been stored in different virtual machines, within a single physical machine. Data from a target virtual machine could be stolen easily by compromising the other virtual machine which is co resident with the target machine. The third party auditor (TPA), is the one who has expertise and capabilities that cloud to assess the cloud storage service security on behalf of the user upon request. TPA checks for the availability and confidentiality of user data. TPA audits the cloud data storage without local copy of data. However, allowing the integrity check without requiring the entire data is one more challenge in cloud computing. The verifier who verifies the data may have to download the entire data, but it is not the real case, since there will be huge amount of wastage in computation and communication resource. One approach is to have a security-mediator (SEM). This SEM should not learn anything about the uploaded data. A cloud user cannot fully depend on the cloud in terms of data security. They would not be happy with the security of the VM or the honesty of the technical staff. For these reasons, every cloud users depends on data encryption. The encrypted data will be stored and maintained on a server. The data owners can let their clients to view some portion of the data. Sharing these encrypted files with other users without leaking confidential data is one of the challenging tasks.
If a data owner wants to share some number of files with the others, then either he can encrypt all files with single key and gives corresponding key to respected user or he can encrypt the files with different keys and provide corresponding number secret keys to the data user. Both techniques are inefficient since in first method, the user can also be able to see some unwanted files. In the second method the number of keys will be equal to the number of encrypted file and it also requires large and expensive storage space for the keys. One solution is that the data owner encrypts the files with different public keys but send the user a single decryption key of constant-size (fig 3.1). By using this single key the user can access only the requested files.

## II. Literature Survey

This KAC system is developed by considering some of the drawbacks occurred in previous work.

**2.1 Fully functional identity-based encryption scheme (IBE)-** This scheme has chosen ciphertext security in the random oracle model assuming an elliptic curve variant of the computational Diffie Hellman problem. It gives precise definitions for secure identity based encryption schemes and gives several applications for such systems. In this paper it proposes a fully functional identity-based encryption scheme. The performance of this system is comparable to the performance of ElGamal encryption in Fp. The security of the system is based on a natural analogue of the computational Diffie-Hellman assumption on elliptic curves. Based on this assumption it shows that the new system has chosen ciphertext security in the random oracle model. This system can be built from any bilinear map e : G1 x G1 → G2 between two groups G1, G2 as long as a variant of the Computational Diffie Hellman problem in G1 is hard.

**2.2 Attribute-Based Encryption for Fine-Grained Access Control (ABE)-** The user data is encrypted and stored in the cloud by third party sites. But the disadvantage of encrypting the data is that, it can be shared at coarse-grained level by giving private key to another party. The Key-Policy Attribute-Based

Encryption (KP-ABE) technique or a cryptosystem have been developed for fine-grained sharing. The two authors, Sahai and Waters introduced initial steps to solve this problem of fine-grained level by using a concept called Attributed-Based Encryption (ABE). A much richer type of attribute based cryptosystem scheme called Key-Policy Attribute-Based Encryption (KP-ABE) has developed. In this system a cipher text is denoted by a set of descriptive attributes. Every private key will have its own access structure that specifies which cipher text the key can decrypt.

**2.3 Scalable Hierarchical Access Control in Secure Group Communications-** This technique describes the group communication systems containing multiple data streams and users with different access privileges Let {r1, r2, …}denote the set of resources in the system. In the group communication scenario, each resource corresponds to a data stream that is transmitted in one multicast session. Each multicast session is associated with a multicast address and a multicast routing tree. The routing trees for different multicast sessions can be jointly constructed. From the data transmission points of views, the users belonging to the same multicast session form a Data Group (DG). That is, one DG contains the users that can access to a particular resource. It is clear that the DGs can have overlapped membership because users may subscribe multiple resources. The users are also divided into non-overlapping Service Groups (SG) according to access privilege. One SG contains the users that are authorized to access the exactly same set of resources.

## III. Existing System

The data owner classifies the cipher text classes according to the subject. The nodes in the tree represent a secret key, leaf nodes represents keys for each and every cipher text classes

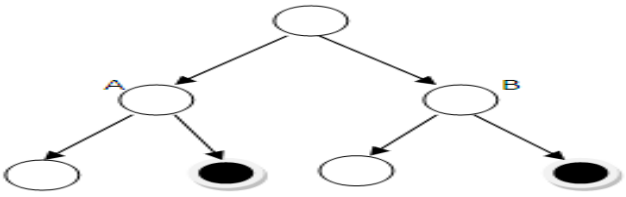


**Figure.1. 1 a**

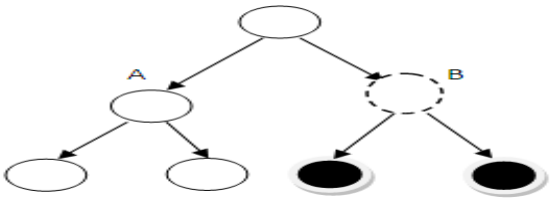


**Figure.2. 1 b**

The circles with dotted lines represent the key to be granted. If a data owner [fig(1a)] wants to share all the files under the node personal with the user then he has to provide the key for that node, it automatically grants the key of its descendant nodes. In case of, fig (1b) if the data owner wants to share some of the files under the different nodes then the number of secret keys will be increases as the number of files or classes. For every files owner has to create separate keys and send them to user.This is not an efficient method. This technique is suitable where the owner wants to share all his files under certain branches. The existing scheme is an inadequate technique since it increases the total number of keys. It requires secure channel to transmit and storage space to store large number of keys.

## IV. Proposed System

In the proposed scheme, it represents Key Aggregate Cryptosystem (KAC) under public key encryption. Here the messages are encrypted under the identifier of cipher text called class. Every cipher texts are grouped into different number of classes. The master secret key is used to extract the aggregate key. This aggregate key is a unique key is used for a single class. The Key aggregation property is especially useful when we expect the delegation to be efficient and flexible. The schemes enable a content provider to share her data in a confidential and selective way, with a fixed and small ciphertext expansion. Data owner provides this aggregate key to the respected user; the user decrypts the files for that particular class or set. The files out of the set remain confidential. The cipher text, public key, master secret key and aggregate keys are all of constant size.

### 3.1 KAC

The problem statement can be given like this, Making a decryption keys more powerful to make it to decrypt many ciphertexts, without increasing the key size. The Key Aggregate Cryptosystem (KAC) is one of the efficient techniques under public-key encryption. Here the messages are encrypted under the identifier of cipher text called class. Every cipher texts are grouped into different number of classes. The master secret key is used to extract the aggregate key. This aggregate key is a unique key is used for a single class. Data owner provides this aggregate key to the respected user; the user decrypts the files for that particular class or set. The files out of the set remain confidential. The cipher text, public key, master secret key and aggregate keys are all of constant size. The cipher text classes are represented in a hierarchical manner.

#### a. Advantages

By having this method, at the same time multiple files can be extracted by using a single constant-size key. The number of key size will be reduced. It does not require much storage space to store this aggregate key. The files which are out of the requested set of data will remain confidential. The overview of the project is given below. The files will be stored in encrypted form in a cloud.

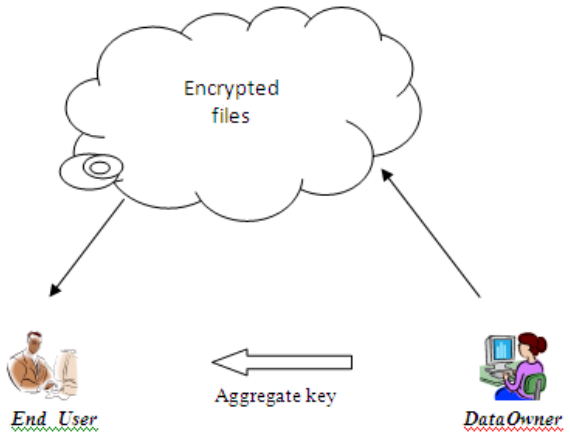


**Figure. 3. Encrypted files**

### 3.3 Problem Statement

The problem statement is to designing the public-key encryption scheme in which it supports that any set of cipher texts can be decrypt able by a constant- size decryption key called an aggregate key.

### 3.4 Architecture

This shows the architecture of how the user retrieves the aggregate key

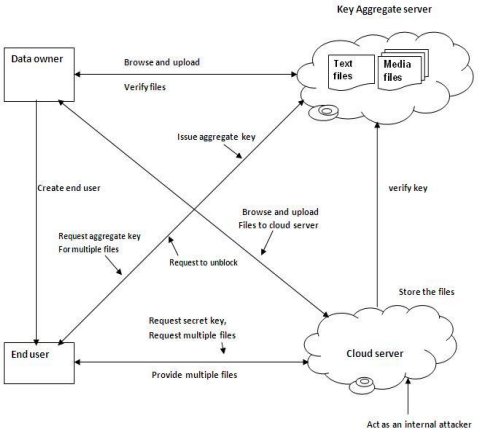


**Figure.4. Architecture**

In this architecture diagram, there are four different modules namely data owner, key aggregate server, cloud server and end user. The arrows indicates the interaction between the modules.

## V. Techniques

There are five polynomial algorithms.

**Setup($1^{\lambda}$,n) :** This is executed by the data owner to setup an account on a server. On a security level parameter $1^{\lambda}$ and the number of cipher text class n, it generate public system parameter.

**KeyGen Phase:** It is executed by the data owner to generate public or master secret key pair.

**Encrypt(pk,i,m):** This can be executed by anyone who wants to encrypt the data. This take input as public key pk, an index i, and a message m, then it provides ciphertext C.

**Extract(msk,S):** It is executed by the data owner. Input the master secret key msk, ans a set S for different classes, it generates aggregate key $K_S$ for set S.

**Decrypt($K_S$,S,i,C):** Executed by users who want to decrypt the file. It takes the input as $K_S$, the set S, an index i, and ciphertext C, then it outputs plain message m. The aggregate key formed by combination of different master secret keys of the set together, with the index of the cipher text classes.

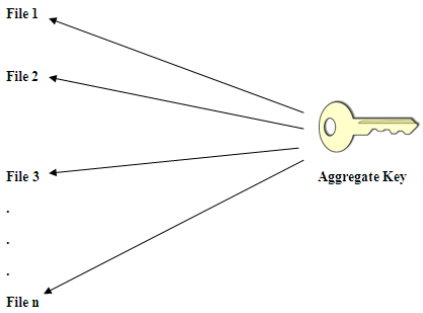


**Figure.5.Aggregate key**

In fig 4, one aggregate key is generated for the number of files requested by the user.

### 4.1 Algorithms

1. AES - File Encryption and decryption

2. SHA1 - Secured Hash Algorithm - for generating Digital Signature

3. Aggregate Key Generation algorithm - To generate an unique key for multiple file request

4. RSA - To generate Public Key for accessing file

## VI. Cloud Storage

In cloud storage, data is stored in an encrypted form. We can store text files, audio files, video files and image files. This is a secured storage for cloud users. Data owner can see the files stored by him.

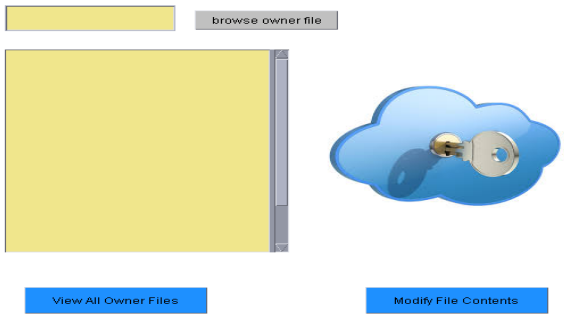


**Figure. 5. browse owener file**

The figure shows the cloud storage where data owner can browse and modify the contents of the file.

## VII. RESULT

This is flexible and efficient method to share the data with other users. The files can be easily uploaded to the cloud. The amount of time taken to upload the files and to get the confirmation message from the cloud is given by the graph below (fig 4.3).

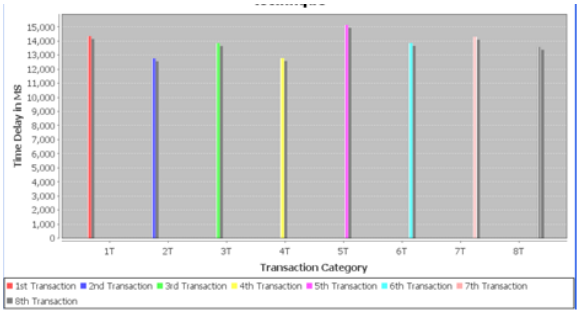


**Figure.6.graph**

This technique will generate a single aggregate key for different files, and extraction of multiple files can be done by using a constant size aggregate key. The aggregate key can be extracted by providing the different file names. This also reduces the memory space for storing number of secret keys by producing a single key.

## VII. CONCLUSION

This approach is more flexible than hierarchical key assignment which can only save spaces if all key-holders share a similar set of privileges. secret keys for different ciphertext classes in cloud storage. No matter which one among the power set of classes, the delegatee can always get an aggregate key of constant size.

## VIII. REFERENCE

[1] Cheng-Kang Chu, Sherman S. M. Chow, Wen-GueyTzeng, Jianying Zhou, and Robert H. Deng, "Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage" IEEE Transactions on Parallel and Distributed Cloud Computing Systems,Volume:25, Issue:2,Issue Date:Feb.2014

[2] 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, 2013.

[3] B. Wang, S. S. M. Chow, M. Li, and H. Li, "Storing Shared Data on the Cloud via Security-Mediator," in International Conference on Distributed Computing Systems - ICDCS 2013. IEEE, 2013.

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

[5] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, "Dynamic and Efficient Key Management for Access Hierarchies," ACM Transactions on Information and System Security (TISSEC), vol. 12,no. 3, 2009.

[6] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted data,"

[7] D. Boneh and M. K. Franklin, "Identity-Based Encryption from the Weil Pairing," in Proceedings of Advances in Cryptology – CRYPTO '01, ser. LNCS, vol. 2139. Springer, 2001, pp. 213–229.

[8] R. Canetti and S. Hohenberger, "Chosen-Ciphertext Secure Proxy Re-Encryption," in Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07).

[9] M. Chase and S. S. M. Chow, "Improving Privacy and Security in Multi-Authority Attribute-Based Encryption," in ACM Conference on Computer and Communications Security, 2009, pp. 121–130.]

[10] Y. Sun and K. J. R. Liu, "Scalable Hierarchical Access Control in Secure Group Communications," in Proceedings of the 23th IEEE International Conference on Computer Communications