Volume 5, Issue 6, June 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Compact Key Generation for Scalable Data Sharing in Cloud Storage

R Muni Prashanthi*, P Rajasekhar
Dept of CSE, SEAT, JNTU
Anantapur, Andhra Pradesh, India

Abstract— Sharing of data is an important functionality in cloud storage. It is important to consider the privacy of data which is stored on the cloud. An attempt was made to share data securely, efficiently, and flexibly with others in cloud storage. A new approach was proposed that uses a public-key cryptosystem which generates constant-size cipher texts. The idea behind this approach is that it can aggregate a number of secret keys and make them as compact as a single key which encompasses the power of all the individual keys. This compact aggregate key can be conveniently sent to others or be stored in a smart card with very limited secure storage. Since cloud is not an open source users may have different access rights according to their usage for example paid users and normal users.

Keywords— cloud storage, data sharing, key aggregate, access rights.

I. INTRODUCTION

Cloud storage is gaining its popularity day-to-day. In real life there is a rise in demand for data outsourcing, which assists in the strategic management of corporate data. It is also used as a core technology behind many online services for personal applications. Nowadays it is easy to create free accounts for email; photo album; file sharing and/or remote access, with storage size more than 25 GB. Together with the current wireless technology, users can access almost all of their files and emails by mobile phone in any corner of the world.

Considering data privacy, it is better to rely on the server to enforce the access control only after authentication [1]. Un-authorized access may lead to expose of all data. This becomes more danger in a shared cloud computing environment.

Data from different clients can be hosted on different virtual machines (VMs) which reside on a single physical machine. Data stored on the target VM can be stolen by instantiating another VM which is co resident with the target machine [2]. In order to check availability of files on the server in the presence of data owner without leaking the data [3], or without compromising data owner's anonymity [4] there are a series of cryptographic schemes which checks the availability of files by considering a third- party auditor. Likewise, cloud users probably will not hold the strong belief that the cloud server is doing a good job in terms of confidentiality.

A cryptographic solution [5] relies on number theoretic assumptions are more desirable whenever the user is not satisfiable by trusting the security of the VM or the honesty of cloud staff. These users are motivated to encrypt their data with their own keys before uploading them to the server.

Data sharing is an important functionality in cloud storage. For example, blogs are created by a group of people, upload pictures, documents related to them and enable the group members to view and download data; in an enterprise a large amount of sensitive data is placed on the server and it grants rights to its employees to access a small portion of data. The challenging problem is how to effectively share encrypted data. Of course users can download the encrypted data from the storage, decrypt them, then send them to others for sharing, but it loses the value of cloud storage. Users should be able to delegate the access rights of the sharing data to others so that they can access these data from the server directly. However, finding an efficient and secure way to share partial data in cloud storage is not trivial. Below we consider Dropbox as an example for illustration.

Assume that Alice puts all her private photos on Dropbox, and she does not want to expose her photos to everyone. Due to various data leakage possibility Alice does not feel relaxed by just relying on the privacy protection mechanisms provided by Dropbox, so she encrypts all the photos using her own keys before uploading. One day, Alice's friend, Bob, asks her to share the photos taken over all these years in which Bob has appeared. Alice can then use the share function of Dropbox, but the problem now is how to delegate the decryption rights for these photos to Bob. A possible option Alice can choose is to securely send Bob the secret keys involved. There are two possible ways for her under the traditional encryption paradigm:

- Alice encrypts all files with a single encryption key and gives Bob the corresponding secret key directly.
- Alice encrypts files with distinct keys and sends Bob the corresponding secret keys.

Obviously, the first method is inadequate since all un-chosen data may also be leaked to Bob as the secret key for all other photos is same. For the second method, there are practical concerns on efficiency. The number of such keys is as many as the number of the shared photos, say, a thousand. Transferring these secret keys inherently requires a secure channel, and storing these keys requires expensive secure storage. The costs and complexities involved generally increase with the number of the decryption keys to be shared. In short, it is very heavy and costly to do that. Even though

the keys and encrypted data are transferred there will be a confusion regarding for which photo or file a particular key belongs to. In order to avoid this confusion a new approach of key aggregation is proposed.

Cryptography is process of encryption and decryption. Encryption and Decryption can be done in two ways by using symmetric and asymmetric algorithms. Symmetric key algorithms include DES and AES. Asymmetric algorithms provide key distribution and secrecy (Diffie -Hellman key exchange), some provide digital signatures (DSA-Digital signature Algorithm), and some provide both such as RSA. The keys generated are of two flavors—symmetric key or asymmetric (public) key. Using symmetric encryption, when Alice wants the data to be originated from a third party, she has to give the encryptor her secret key; obviously, this is not always desirable. By contrast, the encryption key and decryption key are different in public key encryption. The use of public-key encryption gives more flexibility for our applications. For example, in enterprise settings, every employee can upload encrypted data on the cloud storage server without the knowledge of the company's master-secret key.

Therefore, the best solution for the above problem is that Alice encrypts files with distinct public-keys, but only sends Bob a single (constant-size) decryption key. In cloud storage we may find various kinds of files such as doc, pdf, jpg, gif, txt, etc. For each kind of file format a unique decryption key of constant size is generated and is sent to the delegate who prefers to decrypt the file and view the actual data. Since the decryption key should be sent via a secure channel and kept secret, small key size is always desirable. For example, we cannot expect large storage for decryption keys in the resource-constraint devices like smart phones, smart cards, or wireless sensor nodes. Especially, these secret keys are usually stored in the tamper-proof memory, which is relatively expensive. The present research study uses symmetric key algorithm AES (Advanced Encryption Standard) algorithm and uses the concept of key-aggregation for a number of files of same format which mainly focuses on minimizing the communication requirements (such as bandwidth, rounds of communication) like aggregate signature.

II. LITERATURE SURVEY

This section briefly reviews the state-of-the-art of proving security to the data stored on the cloud storage. Different approaches were proposed by many scholars and few of them are mentioned below.

A method was proposed by [6] to generate a tree hierarchy of symmetric-keys by using repeated evaluations of pseudorandom function/block-cipher on a fixed secret. The concept can be generalized from a tree to graph. In another system [7] data owner encrypts the data, public key, data index and then uploads to the cloud server. Data owner generates aggregate decryption key using its master-secret key, and shares the data to other users by sending its ADK to users via a secure E-mail on the other side the data user decrypts the data in this process steganography was used.

Another technique [8] introduces a special type of encryption called as key-aggregate cryptosystem which allows user to share their data partially across cloud and produces constant-size cipher text. In this technique user provide a constant-size aggregate key for different cipher text classes in cloud storage, but the other encrypted files outside the class remain confidential

III. PROPOSED SYSTEM

Considering the advantages and disadvantages of the existing system a new approach called key-aggregate cryptosystem (KAC) [10] was proposed. The cloud storage is repository of data where files of different formats such as doc, txt, jpg, gif, or any other format is stored. A group of people pertaining to enterprise or institution may use this repository to perform a task. This people will have the ability to perform cryptographic functions such as encryption and authentication multiple times. An assumption was made before performing any cryptographic function. The files on the cloud storage must be categorized into different classes depending on their file type and size. Once the data is categorized it must be labeled by a unique class number, which is further used for generating an aggregate key. After this assumption the actual implementation is explained below.

In KAC, users encrypt a message under a public-key and a cipher text identifier called a class. This means the cipher texts are in turn categorized into different class classes. The owner holds a key called a master-secret key, which can be used to extract secret keys for different classes. The extracted key is an aggregate key which is as compact as a secret key used for a single specific class, but encompasses the power of many keys which is decryption power for a subset of cipher text classes. In KAC approach the size of ciphertext, public-key, master-secret key, and aggregate key are of constant size. With this solution, Alice can simply send Bob a single aggregate key via secure channel; Bob can download the encrypted photos from Alice's Dropbox and then use the aggregate key to decrypt these encrypted photos as shown in figure 1.

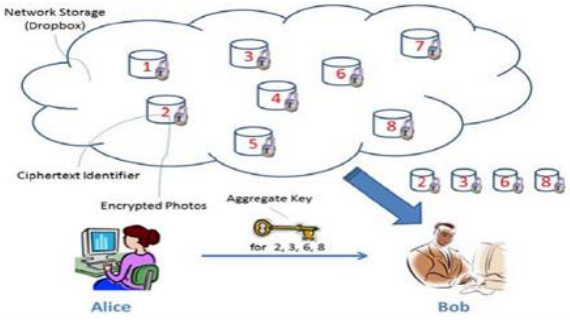

Figure1: Alice shares files with identifiers 2, 3, 6, and 8 with Bob by sending him single compact key.

How this aggregation of keys goes? Is explained below. The concept of key-aggregate cryptosystem comprises of five different algorithms as stated below:

1. Initialization phase: executed by data owner to setup an account on an untrusted server. On input a security level parameter S_l and the number of ciphertext classes C_n, it outputs the public system parameter *param.*

2. KeyPairGen: executed by the data owner to randomly generate a public/master-secret key pair *(pk,msk)*.

3. Encryption *(pk,C$_i$,mes)*: executed by anyone who wants to encrypt data. On input a public-key *pk*, an index C_i denoting the ciphertext class, and a message *mes*, it outputs a ciphertext *Cipher.*

4. Extraction *(msk,S)*: executed by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegate. On input the master-secret key *msk* and a set *S* of indices corresponding to different classes, it outputs the aggregate key for set *S* denoted by $A_{ks.}$

5. Decryption *(A_{ks}., S, C$_i$, Cipher):* executed by a delegate who received an aggregate key A_{ks} generated by Extract. On input $A_{ks,}$ the set *S*, an index C_i denoting the ciphertext class the ciphertext *Cipher* belongs to, and *Cipher*, it outputs the decrypted result *mes* if C_i *belongs to S*.

The key generated using this process is efficient, time saving and does not require more amount of memory. This approach is efficient in terms of cost and bandwidth.

IV. PERFORMANCE OF PROPOSED SYSTEM

KAC approach allow the compression factor F to be a tunable parameter, at the cost of O(Cn) sized system parameter. KAC includes both encryption and decryption, encryption can be done in constant time, while decryption can be done in O(|S|) group multiplications with two pairing operations where S is denotes the set of cipher text classes decryptable by the granted aggregate key and |S|<=Cn.

When KAC was implemented in C by using pairing-based cryptography (PCB) library [9], was proved that KAC shows optimal results. For our experiment we consider the number of cipher text classes Cn=2^{16}=65536. The experimental results are as shown in the following table1.

Table1: Performance of KAC approach for different delegation ratio r.

r	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Init	8.4								
Extraction	2	4	5	7	8	9	10	10	11
Decryption	4	6	9	12	14	15	16	18	20

The table shows the execution times of initialization phase, extraction and decryption phases. It can be observed that all the three phases has taken different times to perform its function and are independent of delegation ratio r. the running time complexities of Extraction and Decryption increase linearly with delegation ratio r. It can be seen from the above table that the running time of Decryption is double that of Encryption. This PCB library uses Type-A curves and accompanies only 65536 cipher text classes. For applications where the number of cipher text classes is large but the non-confidential storage is limited, one should deploy KAC with PCB which uses Type-D curves.

V. CONCLUSION

The central tendency of cloud storage is "how to protect users' data privacy?" many approaches have come into the field but failed in terms of efficiency and cost considerations. Cryptographic schemes are getting more versatile and involve multiple keys for a single application. Our approach was considered in all these aspects and generated a novel approach to "compress" secret keys in public-key cryptosystems which support delegation of different cipher text classes in a cloud storage. The delegate can decrypt the encrypted data by obtaining the aggregate key and need not worry about to which class the data belong to. Our approach is more flexible than hierarchical key assignment which can only save spaces if all key-holders share a similar set of privileges. In addition to this we can provide different access rights to different users.

REFERENCES

[1] 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.

[2] L. Hardesty, **Secure Computers Aren't so Secure,** MIT press, 2009.

[3] 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.

[4] B. Wang, S.S.M. Chow, M. Li, and H. Li, **Storing Shared Data on the Cloud via Security-Mediator,** Proc. IEEE 33rd (ICDCS), 2013.

[5] 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.

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

[7] S.Prasanna and S.Ramay, **Implementation of Key Aggregate Cryptography with Stegnography for Secured Data Sharing in Cloud Computing**.

[8] Rashmi Khawale and Roshani Ade **Patient Controlled Encryption using KeyAggregation,** Interantional Journal of ComputerApplications-2015.

[9] http://crypto.standard.edu/pbc

[10] 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.** IEEE transactions on Parallel and Distributed Systems, Vol.25. No.2 February,2014.

