ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

IJARCCE

nCORETech

LBS College of Engineering, Kasaragod

Vol. 5, Special Issue 1, Feburary 2016

Key Aggregate Cryptosystem for Scalable Data Sharing with Policy Based File Assured Deletion for Secure Data Storage in Cloud

Dhanya K K[1], Sindhu S[2]

PG Student, Dept. of CSE, NSSCE, Palakkad, India [1]

Associate Professor, Dept. of CSE, NSSCE, Palakkad, India [2]

Abstract: Data sharing is an important functionality in cloud storage. Cloud computing technology is widely used so that the data can be outsourced on cloud can be accessed easily. But the user has no control over the outsourced data. A new public key encryption method is based on aggregate key is discuss here. Instead of sending large no of decryption keys, Sender can release a constant size aggregate key that can send via e-mail securely. In policy-based file assured deletion scheme that reliably deletes files with regard to revoked file access policies. The encrypted files can be securely deleted and remain permanently inaccessible after a predefined duration. The main idea is that a file is encrypted with a data key, and this data key is further encrypted with a control key that is maintained by a separate key manager service. The control key is time-based, meaning that it will be completely removed by the key manager, when an expiration time is reached. Without the control key, the data key and hence the data file remain encrypted and are deemed to be inaccessible. Thus, the main security property of file assured deletion is that even if a cloud provider does not remove expired file copies from its storage, those files remain encrypted and unrecoverable.

Keywords: Virtual machine, Key aggregate encryption, ciphertext , Attribute based Encryption, Aggregate key.

I. INTRODUCTION

Cloud has become very important in internet world. Cloud provides storages, platforms which improves the functionality. Cloud storage shows how securely and flexibly we can store and share our data. The shared data in cloud servers, however, usually contains user's sensitive information such as personal profile, financial data, health records, etc. are needs to be well protected. As the ownership of the data is separated from the administration of them, the cloud servers may migrate users data to other cloud servers in outsourcing or share them in cloud searching. Therefore, it becomes a big challenge to protect the privacy of those shared data in cloud.

A special type of encryption called key-aggregate cryptosystem which allows user to share their data partially across cloud and which produces constant size ciphertext. In this technique user provide a constant-size aggregate key for different ciphertext classes in cloud storage, but the other encrypted files outside the class remain confidential. The importance is that one can aggregate any set of secret keys and make them as compact as a single key, but encompassing the power of all the keys being aggregated.

Suppose Alice put all data on DropBox.com and she does not want to expose her data to everyone. Due to data leakage possibilities she does not trust on privacy mechanism provided by DropBox.com, so she encrypt all data before uploading to the server. If Bob ask her to share some data then Alice use share function of DropBox.com. But problem now is that how to share encrypted data.
There are two severe ways: 1. Alice encrypt data with single secret key and share that secret key directly with the Bob. 2. Alice can encrypt data with distinct keys and send Bob corresponding keys to Bob via secure channel. In first approach, unwanted data also get expose to the Bob, which is inadequate. In second approach, no. of keys is as many as no. of shared files, which may be hundred or thousand as well as transferring these keys require secure channel and storage space which can be expensive[1].

In KAC, the encryption is done through an identifier of ciphertext known as class, with public key. The classes are formed by classifying the ciphertext. The key owner has the master secret key which is helpful for extracting secret key. So in above scenario now Alice can send an aggregate key to bob through an email and the encrypted data is downloaded from dropbox through the aggregate key [1].

II. RELATED WORKS

Symmetric-Key Encryption with Compact Key

In symmetric key encryption, the derivation of the key for a set of classes (which is a subset of all possible cipher text classes) is as follows. A composite modulus is chosen where p and q are two large random primes [8]. A master secret key is chosen at random. Each class is associated with a distinct prime. All these prime numbers can be put in the public system parameter. A constant-size key for set can be generated. For those who have been delegated the access rights for S' can be generated. The content provider needs to get the corresponding secret keys to encrypt data which is not suitable for many applications [8].

Attribute-Based Encryption

Attribute based encryption is a type of public key encryption in which the secret key of a user and the ciphertext are dependent upon attributes (e.g. the country he lives, or the kind of subscription he has) [5]. In such a system, the decryption of a ciphertext is possible only if the set of attributes of the user key matches the attributes of the ciphertext. Although ABE concept is very powerful and a promising mechanism, but confronted with key escrow problem [5]. Attribute set belonging to one user is usually monitored by different authorities in this era of collaboration. ABE is collusion-resistance but not the compactness of secret keys. Indeed, the size of the key often increases linearly with the number of attributes it encompasses, or the ciphertext-size is not constant (e.g., [4]).

Identity-based encryption (IBE**)**

IDE is a vital primary thing of identity based cryptography. The public key of user contains distinct information of user's identity. The key can be textual value or domain name, etc. IDE is used to deploy the public key infrastructure. The identity of the user is used as identity string for public key encryption [1]. A trusted party called private key generator (PKG) in IBE which has the master secret key and gives secret key to users according to the user identity. The data owner collaborate the public value and the identity of user to encrypt the data. The cipher text is decrypted using secret key [8].The main drawback is that the ciphertext produced by encryption algorithm is not constant [8].

Different Schemes	**Ciphertext size**	**Decryption key size**	**Encryption type**
Key assignment schemes	Constant	Non-constant	Symmetric or public-key
Symmetric-key encryption with compact key	Constant	Constant	Symmetric key
IBE with compact key	Non-constant	Constant	Public key
Attribute based encryption	Constant	Non-constant	Public key
KAC	Constant	constant	Public key

III.OVERVIEW

In key-aggregate cryptosystem (KAC), users encrypt a message not only under a public-key, but also under an identifier of ciphertext called class. That means the ciphertexts are further categorized into different classes. The key owner holds a master-secret called master-secret key, which can be used to extract secret keys for different classes. More importantly, the extracted key have can be an aggregate key which is as compact as a secret key for a single class, but aggregates the power of many such keys, i.e., the decryption power for any subset of ciphertext classes [1].With our example, Alice can send Bob a single aggregate key through a secure e-mail. Bob can download the encrypted photos from Alice's Box.com space and then use this aggregate key to decrypt these encrypted data. The sizes of ciphertext, public-key, and master secret Key and aggregate key in KAC schemes are all of constant size. The public system parameter has size linear in the number of ciphertext classes, but only a small part of it is needed each time and it can be fetched on demand from large(but non-confidential) cloud storage.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
IJARCCE
nCORETech
LBS College of Engineering, Kasaragod
Vol. 5, Special Issue 1, Feburary 2016

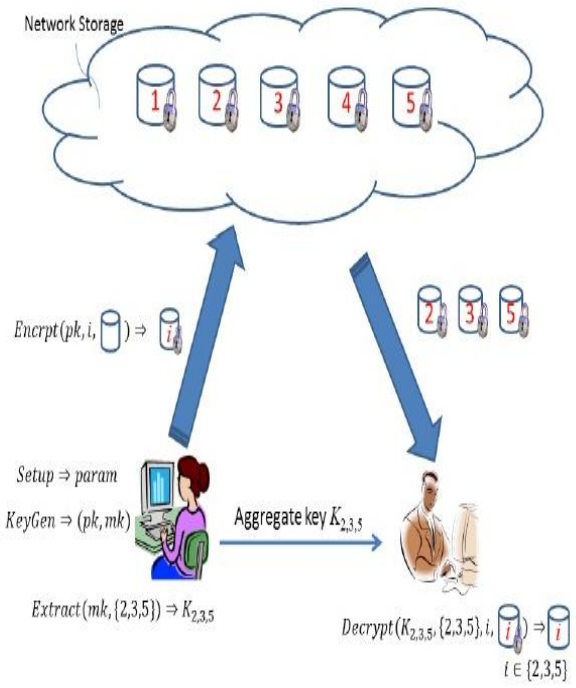
Network Storage
1
2
3
4
5
Encrpt(pk, i,) ⇒ i
2
3
5
Setup ⇒ param
KeyGen ⇒ (pk, mk)
Aggregate key $K_{2,3,5}$
Extract(mk, {2,3,5}) ⇒ $K_{2,3,5}$
Decrypt($K_{2,3,5}$, {2,3,5}, i, i) ⇒ i
i ∈ {2,3,5}

Fig. 1 Encryption and decryption in KAC.

IV. IMPLEMENTATION

A key-aggregate encryption scheme consists of five polynomial-time algorithms as follows.

1. Setup Phase

The data owner executes the setup phase for an account on server which is not trusted. The setup algorithm only takes implicit security parameter.

2. KeyGen Phase

This phase is executed by data owner to generate the public or the master key pair (pk, msk)

3. Encrypt Phase

This phase is executed by anyone who wants to send the encrypted data. Encrypt (pk, m, i), the encryption algorithm takes input as public parameters pk, a message m, and i denoting ciphertext class. The algorithm encrypts message m and produces a ciphertext C such that only a user that has a set
of attributes that satisfies the access structure is able to decrypt the message.

- Encrypt (pk, i, m) : It is executed by data owner and for message m and index i ,it computes the ciphertext as C.

4. Extract Phase

This is executed by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegate.

- Extract (msk, S): It is executed by data owner for delegating the decrypting power for a certain set of ciphertext classes and it outputs the aggregate key for set S denoted by Ks.

5. Decrypt Phase

Decrypt (Ks, S, I, C): It is executed by a delegate who received, an aggregate key Ks generated by Extract. On input Ks, set S, an index i denoting the ciphertext class ciphertext C belongs to and output is the original message m.
In policy based file assured deletion means that files are reliably deleted and remain permanently unrecoverable and inaccessible by any party. It decouples the management of encrypted data and encryption keys, such that encrypted data remains on third-party (untrusted) cloud storage providers, while encryption keys are independently maintained by a key manager service. It generalizes time-based file assured deletion (i.e., files are assuredly deleted upon time expiration) into a more fine-grained approach called policy based file assured deletion, in which files are associated

with more flexible file access policies (e.g., time expiration, read/write permissions of authorized users) and are assuredly deleted when the associated file access policies are revoked and become obsolete.
Time-based file assured deletion, which means that files can be securely deleted and remain permanently inaccessible after a predefined duration. The main idea is that a file is encrypted with a data key, and this data key is further encrypted with a control key that is maintained by a separate key manager service. The control key is time-based, meaning that it will be completely removed by the key manager when an expiration time is reached, where the expiration time is specified when the file is first declared. Without the control key, the data key and hence the data file remain encrypted and are deemed to be inaccessible. Thus, the main security property of file assured deletion is that even if a cloud provider does not remove expired file copies from its storage, those files remain encrypted and unrecoverable[2].

Suppose that we have enforced the condition of policy renewal. A straightforward approach of implementing policy renewal is to combine the file upload and download operations, but without retrieving the encrypted file from the cloud. The procedures can be summarized as follows: (i) download all encrypted keys from the storage cloud, (ii) send them to the key manager for decryption, (iii) recover the data key, (iv) re- encrypt the data key with the control keys of the new policies, and finally (v) send the newly encrypted keys back to the cloud.
It is still important to improve the robustness of the key manager service to minimize its chance of being compromised. To achieve this, we can use a quorum of key managers [2], in which we create n key shares for a key, such that any $k < n$ of the key shares can be used to recover the key. While the quorum scheme increases the storage overhead of keys, this is justified as keys are of much smaller size than data files.

V. CONCLUSION

Cloud storage offers an abstraction of infinite storage space for clients to host data, in a pay-as-you-go manner.KAC is an efficient public-key encryption scheme which supports flexible delegation in the sense that any subset of the ciphertexts (produced by the encryption scheme) is decryptable by a constant-size decryption key (generated by the owner of the master-secret key). The receiver gets securely an aggregate key of constant size. In cloud storage, the number of cipher texts usually grows rapidly without any restrictions. So we have to reserve enough cipher text classes for the future extension. Sharing of decryption keys in secure way plays important role. Public-key cryptosystems provides delegation of secret keys for different ciphertext classes in cloud storage. In policy based file assured deletion files are reliably deleted and remain permanently unrecoverable and inaccessible by any party.

ACKNOWLEDGMENT

First and foremost I thank the almighty god for helping me to fulfil this work. I also extend my sincere thanks from the depth of my heart to my guide Mrs. Sindhu S, Associate Professor, Department of Computer Science, NSSCE Palakkad for enlightening me through her valuable guidance, help and suggestions. I would also like to extend my gratitude to all my friends, dears and nears for helping me to make this work a success.

REFERENCES

[1] Cheng-Kang Chu ,Chow, S.S.M, 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. Volume: 25, Issue: 2. Year:2014.
[2] FADE: Secure Overlay Cloud Storage with File Assured Deletion by Yang Tang†, Patrick P. C. Lee†, John C. S. Lui†, and Radia Perlman‡The Chinese University of Hong Kong intel Labs, IEEE TRANSACTIONS DEPENDABLE ON SECURE COMPUTING VOL:9 NO:6 YEAR 2012
[3] A Secure Data Self-Destructing Scheme in Cloud Computing By Jinbo Xiong, Student Member, IEEE, Ximeng Liu, Student Member, IEEE, Zhiqiang Yao, Jianfeng Ma, Qi Li, Kui Geng, and Patrick S. Chen, IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 2, NO. 4, OCTOBER-DECEMBER 2014
[4] Improving Privacy and Security in Multi-Authority Attribute-Based Encryption ByMelissa Chase Microsoft Research1 Microsoft Way Redmond, WA 98052, USA melissac@microsoft.com Sherman S.M. Chow∗ Department of Computer Science Courant Institute of Mathematical Sciences New York University, NY 10012, USA schow@cs.nyu.edu
[5] Control Cloud Data Access Privilege and Anonymity With Fully Anonymous Attribute-Based Encryption By Taeho Jung, Xiang-Yang Li, Senior Member, IEEE, Zhiguo Wan, and Meng Wan, Member, IEEE,IEEETRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 10, NO. 1, JANUARY 2015
[6] 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.
[7] T. Okamoto and K. Takashima, ―Achieving Short Ciphertexts or Short Secret-Keys for Adaptively Secure General Inner-Product Encryption,‖ in Cryptology and Network Security (CANS '11), 2011, pp. 138–159.
[8] 4S. S. M. Chow, Y. J. He, L. C. K. Hui, and S.-M. Yiu, "SPICE - Simple Privacy-Preserving IdentityManagement for Cloud Environment," in Applied Cryptography and Network Security – ACNS 2012, ser. LNCS, vol. 7341. Springer, 2012, pp. 526–543.