Efficient Data Sharing with Unbounded Ciphertext Classes in Cloud Storage using Single key

P.Kiruthika, S.Rajeevan, M.Manjunath
Department of Computer Science & Engineering,
SNS College of Technology,
Coimbatore, INDIA

N.Geethanjali
Assistant professor, Department of Computer Science & Engineering,
SNS College of Technology,
Coimbatore, INDIA

Abstract— Cloud computing is an emerging paradigm in which resources of the computing infrastructure can be able to share over the internet. As this paradigm also have many challenges for data security and access control when users outsource their sensitive data in cloud servers. Secure and Efficient Data sharing in Cloud with single key size uses Key Aggregate cryptosystem (KAC) which is a public-key cryptosystem that produce constant-size ciphertexts for any set of secret key and make them compatible as single key as other data in cloud remains confidential. The limitation is that the number of ciphertext classes and the constant key size that makes the secret key predictable and data is unsecure. In this paper, the KAC-UC (KAC with Unbounded Ciphertext classes) scheme is proposed, in which the number of ciphertext classes in cloud storage is unbounded and the ciphertext classes can be extended by expanding the public and master key pair. The Variable size key is been used to securely share data that avoids the problem of being attacked and provides complexity for guessing the secret key. In addition, the ordered search is included to search the required data that is decrypted at the receiver side.

Keywords- Cloud data sharing, Public Key Cryptosystem, Ciphertext set, Variable-size key, ordered search.

I.INTRODUCTION

Cloud computing provide a way to store and share data as service over the internet. It enables several data sharing capabilities and provides an abundant of benefits to the cloud users. There is an increase in the usage of cloud in several organizations to increase their data sharing efforts and to reduce their maintenance costs. Data sharing is an important functionality in cloud storage. So it is necessary that sharing of data between the users should be efficient and secure. It is been accepted that data encryption provides a better solution for this problem. This can be implemented in cloud system by several cryptosystem schemes that use encryption and decryption for data to be shared securely. A cryptosystem is a pair of algorithms which has several encryption and decryption algorithms.

Encryption is a process of converting plaintext into ciphertext using an encryption key.

Decryption is a process of converting ciphertext into the plaintext using decryption key. There are two types of cryptosystems available for sharing of data in cloud computing, they are

1. Symmetric Key Cryptosystem
2. Asymmetric Key Cryptosystem

Symmetric Key Cryptosystem uses single key for both encryption and decryption. In this when the data need to be originated, the secret key need to share to third party. This is always not desirable.

Asymmetric Key Cryptosystem also known as Public key cryptosystem is the solution for the above mentioned problem as it uses different keys for encryption and decryption. This gives more flexibility for the data sharing.

For example, if Alice wants to share some of her photos with Bob but other should remain confidential. She have already encrypted all her data before uploading to cloud so it is possible to send the secret key for all the encrypted data or key for selected distinct file. Obviously sending key for the distinct set of files is the best solution.

However, as considered the key size is also important for sharing the data in cloud. To share the set of data with a single key, Key Aggregate Cryptosystem is used [1]. The Key Aggregate Cryptosystem (shown in figure 1) uses a single aggregate key to share the selected set of data and others remain confidential. The set of data are selected from the number of created ciphertext classes (the categorized set of encrypted data and it is identified

using the class identifier i which is less than or equal to n) by the data owner. A single aggregate key is generated for a selected set of data. This key is send via a secure email to the receiver to access the shared data.

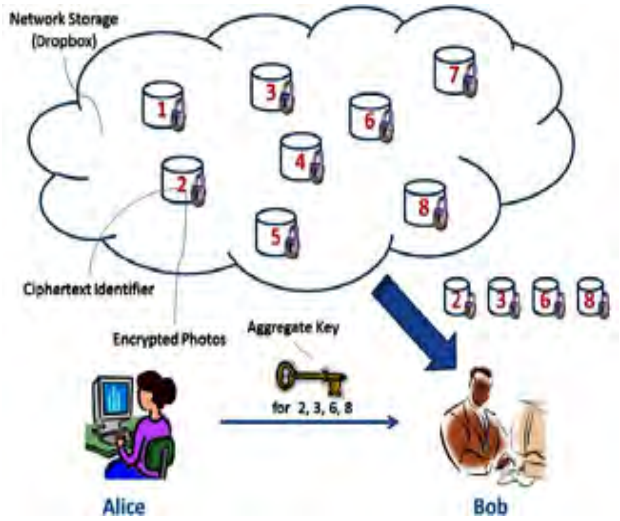

Fig.1 KAC for data sharing in cloud storage.

II.OUR CONTRIBUTIONS

In cryptosystem the major concern is the secrecy of data in the system that able to perform encryption, decryption and authentication several times. In this paper we used a decryption scheme that allows decryption of multiple ciphertexts to provide a variable size single key.

We introduce a new type of public-key encryption with unbounded ciphertext classes with a variable size key. In KAC-UC the data is encrypted with public key and a unique identifier of ciphertext class. The master-secret key is maintained for each ciphertext classes and used for extraction of key.

In previous work the ciphertext classes are limited and the key size remain same as easy for attackers to crack the key. And other schemes like ABE that increases the cost for storing in increase in attribute, in IBE key aggregation is forced that all keys must from different identity divisions and proxy Re-Encryption needs proxy to re-encrypt the data with secret key in which proxy is not trusted.

We propose KAC-UC scheme that provides a single key for aggregate set of data and also includes a extend module that gives a new secret and master key pair for the extension of ciphertext classes. Variable size key is used for providing a secure secret key, i.e., it gives different sized key for each set of different encryption in the cloud.

III.KEY ENCRYPTION

A key encryption scheme has five polynomial-time algorithms as follows.

Setup(1^{μ}, n): executed by the data owner to setup an account on an untrusted server. On input a security level parameter μ and the number of ciphertext classes n (i.e., class index should be an integer ranges from 1 to n), it outputs the public system parameter, which is neglected from the input of the other algorithms for shortness.

KeyGenerate: executed by the data owner to randomly generate a public and master-secret key pair (pk, msk).

Encrypt(pk, i, m): executed by anybody who wants to encrypt data**.** With inputs pk (public-key), i(index) denotes the ciphertext class, and along with m(message), it computes the output ciphertext C.

Extract(msk, S): executed by the data owner for giving the decrypting power for a certain set of ciphertext classes to a data consumer. On giving the input master-secret key (msk) and set of indices(S) corresponds to different classes, it provides the single aggregate key for set S denoted by K_s.

Decrypt(K_s, S, i, C): executed by a data consumer who received an aggregate key K_s generated by Extract. On input K_s, the set S, an index i indicating the ciphertext class the ciphertext C belongs to, and C, it outputs the decrypted message m if i belongs to S.

The drawback in the KAC scheme is that in the Setup phase the number of ciphertext classes is a limited one which is fixed as n. This will limits the extension of ciphertext classes if the data owner wants to extend his ciphertext classes in future. So KAC-UC is proposed to solve this problem, which provides unbounded ciphertext classes. The ciphertext classes can be extended by expanding the public and master key pair. Also the constant size key can be hacked by brute force attack [10] and so a variable size key is a better solution to escape from this attack.

An Ordered searched can be performed at the receiver side to decrypt the most downloaded data. This search is performed by the system itself after verifying the aggregate key at the receiver side. For example, consider this scenario, if a data owner shares the friend class (set of files to share with his friends alone) to two of his friends, if one person had downloaded two files from friend class, then that two files are displayed first to the second person when he downloads the files from friend class. This search will provide the recently and mostly downloaded data.

IV. RELATED WORKS

The integrity (data should not be changed) for the outsourced cloud data will be provided by the third party auditor [2]. This auditing process which is introduced in [2] does not bring new vulnerabilities towards user data privacy, and does not introduce any additional online burden to user. To have a trust the cloud storage, it is necessary a confidential data sharing mechanism [3]. The confidentiality for cloud data is also given by digital signatures. The round of communication is reduced by aggregate signatures [4] which provide a single short to verify a group of signatures**.** This Aggregate signature will cause verifiably encrypted signatures. It enables the verifier to test that a given ciphertext C is the encryption of a signature on a given message M.

The Key Cryptosystems similar to our paper are as follows

1. Key Aggregate Cryptosystem (KAC)
2. Attribute Based Encryption (ABE)
3. Pre-defined Hierarchy
4. Identity Based Encryption (IBE)
5. Proxy Re-Encryption (PRE)

3.1 Key Aggregate Cryptosystem

Key Aggregate Cryptosystem (KAC) will provide a constant key size for a selected set of ciphertext classes in the cloud storage [1]. A single aggregate key will decrypt the set of ciphertext classes only when the aggregate key satisfies the data owner's selected set and the other ciphertext classes which not selected for sharing remains confidential. If the data owner wants to extend ciphertext classes, he can't extend because the number of ciphertext classes in KAC algorithm is predefined and fixed.

3.2 Attribute Based Encryption

Attribute-based Encryption (ABE) is a type of public-key encryption in which the secret key of a user and the ciphertext are dependent upon policy of attributes [5]. In such a system, the decryption of a ciphertext is possible only if the set of attributes of the user key matches the policy of attributes of the ciphertext. For example, with the secret key for the policy (1 v 5 v 7 v 8), one can decrypt ciphertext tagged with class 1, 5, 7, or 8. The problem in ABE scheme is that the size of the key often increases linearly with the number of attributes it contains, or the ciphertext-size is variable.

3.3 Pre-defined Hierarchy

Using tree based structure [11], key for a given branch can be used to derive the keys for its descendant nodes. Just granting the parent key implicitly grants all the keys of its descendant nodes. Also the number of keys increases with the number of branches in the structure.

3.4 Identity Based Encryption

Identity-based Encryption is a type of public-key cryptography in which a publicly known string representing an individual or organization is used as a public key [6]. The public string could include an email address, domain name, or a physical IP address. A trusted third party, called the private key generator (PKG), generates the corresponding private keys. To operate, the Private Key Generator first gives the master public key, and retains its corresponding master private key.

In [8], [7], they tried to build IBE with key aggregation. In their schemes, key aggregation is forced in the manner that all keys that are to be aggregated must be from different Identity Divisions, while the identities and secret keys are of exponential number, but only some of them can be aggregated. Most probably, their key-aggregation [8], [7] comes at the cost of O(n) sizes for both ciphertexts and the public parameter, where n is the no. of secret keys which could be aggregated into a constant size one. Thus increases the costs of storing and transmitting ciphertexts, which is unrealistic in many situations such as shared cloud storage.

3.5 Proxy Re-Encryption

Proxy re-encryption allows the proxy to decrypt some of the ciphertexts without sending the secret key. It also allows a proxy to transform a ciphertext computed under Alice's public key into one that can be opened by Bob's secret key [9]. There are many useful applications of this type of encryption. For instance, Alice might wish to temporarily forward encrypted email to her coworker Bob, without providing him the secret key of her own. In such case, Alice could choose a proxy to re-encrypt her incoming mail in such a format that Bob can decrypt those shared files using his own secret key. In this Alice simply needs to provide her secret key to the proxy, but this requires an impractical level of trust in the proxy.

V. PROPOSED SYSTEM

In this paper we propose the KAC-UC scheme with different security levels and extension of ciphertext classes.

i)Extension of ciphertext classes

ii)Variable size aggregate key as secret key and Ordered search.

i)The ciphertext classes can be extended by expanding the public and master key pair(shown in figure 2). If a user needs to categorize his ciphertexts into more number of ciphertexts than n classes, then he could register for additional key pairs (pk_2, msk_2), . . . , (pk_l, msk_l). Each class now is indexed by a two-level index in {i, j} where $1 \leq i \leq n$ and $1 \leq j \leq l$ and the number of classes is increased by n for each additional key.

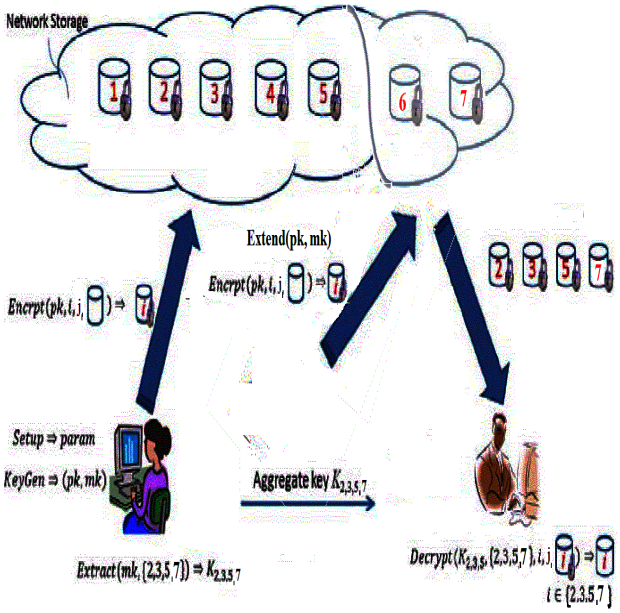

Fig.2 Using KAC-UC to extend ciphertext classes.

In KAC-UC scheme, six polynomial time algorithms are used.

First algorithm is Setup which is executed by the data owner to establish public system parameter.

Next KeyGenerate is executed by the data owner to generate public and master key pair. If the data owner wants to extend his ciphertext classes, he can extend it by Extend algorithm. It will execute KeyGenerate() to get extended public and mater-secret key pair. Then the files or message can be encrypted by Encrypt algorithm and this encrypted file will be uploaded in the cloud storage.

A single aggregate key is generated by Extract algorithm for a selected set of ciphertext classes. Then at the receiver side the encrypted data is decrypted by Decrypt.

Setup(1^{μ}, n): executed by the data owner to setup an account on an untrusted server. On input a security level parameter μ and the number of ciphertext classes n (i.e., class index should be an integer ranges from 1 to n), it outputs the public system parameter, which is neglected from the input of the other algorithms for shortness.

KeyGenerate: executed by the data owner to randomly generate a public and master-secret key pair (pk, msk).

Extend(pk_l, msk_l): executed by the data owner to get extended public and master secret key pair.

Encrypt(pk_l, (i, j), m): executed by the data owner to encrypt the message. Here, the message will be encrypted with two index values i and j where $1 \leq i \leq n$(number of ciphertext classes) and $1 \leq j \leq l$(number of extended public and master key pair).

Extract(msk_l, S_l): executed by the data owner to get the aggregate key for a set of ciphertext classes.

Decrypt(K_s, S_l, (i, j), C): executed by the data consumer to decrypt the set of data using the received aggregate key.

ii) Variable size key is used to avoid Brute force attack. Brute force attack is a random guessing of possible password combinations until getting a right one [10]. The hacker uses an automated tool that can make thousands of requests per minute with credentials generated from a large list of possible values. If a data owner uses a constant size aggregate key, it may be hacked by randomly generating possible combinations of key.

For example, if the aggregate key size is fixed as 4 digits, the hacker will use the automated tool to generate the possible combinations of digits, by that, he can easily get the valid aggregate key. In proposed work, a variable size key is used, so the hacker can't guess the key size.

An Ordered search is included as a way of displaying the recently and mostly downloaded files to the data consumers. So that the data consumer will know about the recent files which is shared by the data owner and can easily retrieve data from the list of data downloaded.

VI. CONCLUSION

In many cryptographic schemes, the number of keys shared for a single application is many, which is inconvenient for the users. In our proposed work, the private keys are compressed into a single key which is more secure and convenient to use at the receiver side. Also the number of ciphertext classes are unlimited which provides more convenient for the data owner to segregate the data sharing to the receivers.

VII. 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," IEEE trans. on parallel and distributed systems, vol. 25, no. 2, 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, Feb. 2013.

[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] Cheng-Chi Lee, Pei-Shan Chung, and Min-Shiang Hwang, "A Survey on Attribute-based Encryption Schemes of Access Control in Cloud Environments," Int. Jour. of Network Security, Vol.15, No.4, PP.231-240, July 2013.

[6] Joonsang Baek, Jan Newmarch, Reihaneh Safavi-Naini, and Willy Susilo, "A Survey of Identity-Based Cryptography," unpublished.

[7] F. Guo, Y. Mu, Z. Chen, and L. Xu, "Multi-Identity Single-Key Decryption without Random Oracles," Proc. Information Security and Cryptology (Inscrypt '07), vol. 4990, pp. 384-398, 2007.

[8] F. Guo, Y. Mu, and Z. Chen, "Identity-Based Encryption: How to Decrypt Multiple Ciphertexts Using a Single Decryption Key, "Proc. Pairing Based Cryptography Conf. (Pairing '07), vol. 4575, pp. 392-406, 2007.

[9] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," unpublished.

[10] Bryan Sullivan, "Preventing a Brute Force or Dictionary Attack: How to Keep the Brutes Away from Your Loot," unpublished.

[11] W.-G. Tzeng, "A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy," IEEE Trans. Knowledge and Data Eng., vol. 14, no. 1, pp. 182-188, Jan./Feb. 2002.