Scalable and Secure Data Sharing in Cloud Storage Using Aggregate Key Generation Algorithm

Sangamesh[1] Rajshekhar Ghogge[2] UdayaTheja.V[3]
[1,3]M.Tech Student [2]Associate Professor
[1,2,3]Department of ISE
[1,2,3]Dr. AIT, Bangalore

Abstract— In Cloud Computing data storage and data sharing is an important and efficient technique. In this paper we will show how efficiently, securely and Flexibly method to share data with other people in cloud storage system. This system produce constant-size cipher texts such that efficient delegation of decryption rights for any set of cipher texts are possible. The novelty 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. In other words, the secret key holder can release a constant-size aggregate key for flexible choices of ciphertext set in cloud storage, but the other encrypted files outside the set remain confidential. In other words, the secret key holder can release constant-size aggregate key for flexible choices of ciphertext set in cloud storage, but the other encrypted files outside the set remain confidential. This compact aggregate key can be conveniently sent to others or be stored in a smart card with very limited secure storage. We provide formal security analysis of our schemes in the standard model. We also describe other application of our schemes.

Key words: Cloud storage, Data sharing, Key aggregate encryption, Efficient delegation and Decryption

I. INTRODUCTION

Cloud Computing has become a new age technology that has got huge potentials in enterprises and markets. Cloud storage gainining popularity recently. Companies are able to rent resources from cloud for storage and other computational purposes so that their infrastructure cost can be reduced significantly. It is also used as core technology behind many online services for personal applications. Further they can make use of company-wide access to applications, based on pay-as-you-go model. Hence there is no need for getting licenses for individual products. Nowadays, it is easy to apply for free accounts for email, photo album, file storage and remote access.

Considering data privacy in cloud computing environment, A traditional way to ensure data privacy is to relay on the server to enforce the access control after authentication, which means any unexpected privilege increase will expose all data. In a shared-lease cloud computing environment things even become bad. Data from different user can be hosted on a separate virtual machine. Regarding availability and security of files, there are a more number of cryptographic schemes were proposed. This scheme allowing a third-party auditor to check the availability of files on behalf of the data owner without leaking any information regarding the information, or without compromising the data owner's secrecy(1).

Data from different clients can be hosted on separate virtual machines (VMs) but reside on a single physical machine. Data in a target VM could be stolen by instantiating another VM co-resident with the target one (2). Regarding availability of files, there are a series of cryptographic schemes which go as far as allowing a third-party auditor to check the availability of files on behalf of the data owner without leaking anything about the data (3), or without compromising the data owners anonymity (4). Likewise, cloud users probably will not hold the strong belief that the cloud server is doing a good job in terms of confidentiality.

Data sharing is an important functionality in cloud storage. For example, bloggers can let their friends view a subset of their private pictures; an enterprise may grant her employees access to a portion of sensitive 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 (5).
Below the Drop Box is an example for this system.

II. RELATED WORK

Considering data privacy, a traditional way to ensure it is to rely on the server to enforce the access control after authentication, which means any unexpected privilege escalation will expose all data. In a shared-tenancy cloud computing environment, things become even worse. Regarding availability of files, there are a series of cryptographic schemes which go as far as allowing a third-party auditor to check the availability of files on behalf of the data owner without leaking anything about the data, or without compromising the data owners anonymity. 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, with proven security relied on number-theoretic assumptions is more desirable, whenever the user is not perfectly happy with trusting the security of the VM or the honesty of the technical staff. In modern cryptography, a fundamental problem we often study is about leveraging the secrecy of a small piece of knowledge into the ability to perform crypto-graphic functions (e.g. encryption, authentication) multiple times. In this paper, we study how to make a decryption key more powerful in the sense that it allows decryption of multiple ciphertexts, without increasing its size(6).

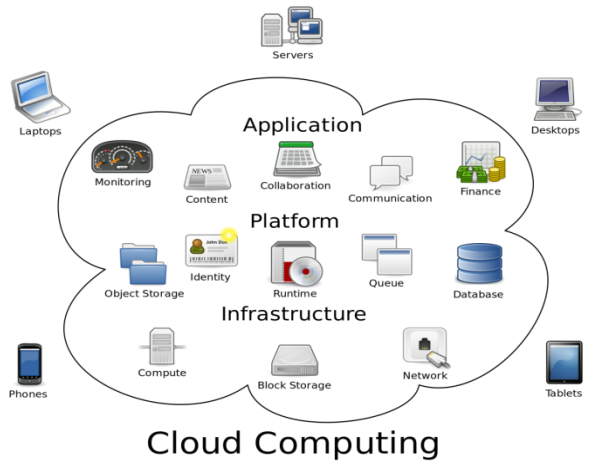

Fig. 2.1: Cloud Computing Storage

In figure 2.1 cloud computing storage it can accessed by all kind of platform. In cloud storage the main reason for cloud access files and it can be stored and it can be shared to many cloud users and while storing and sharing some unauthorized users want to access the important files and important information where the authenticated users think that what they uploaded files are safe. But in traditional system security features for data storing and data sharing they do not have much compared to our proposed system.

A. Cryptographic keys for a predefined hierarchy

Cryptographic key assignment schemes aim to minimize the expense in storing and managing secret keys for general cryptographic use. Utilizing a tree structure, a key for a given branch can be used to derive the keys of its descendant nodes. More advanced cryptographic key assignment schemes support access policy that can be modeled by an acyclic graph or a cyclic graph. An encryption scheme which is originally proposed for concisely transmitting large number of keys in broadcast scenario.

B. Compact Key in Identity-Based Encryption (IBE)

IBE is a type of public-key encryption in which the public-key of a user can be set as an identity string of the user. There is a trusted party called private key generator in IBE which holds a master-secret key and issues a secret key to each user with respect to the user identity. The encryptor can take the public parameter and a user identity to encrypt a message. The recipient can decrypt this ciphertext by his secret key. In this system has a number of applications: it gives very efficient forward secure public key and identity based cryptosystems (with short ciphertexts)(7).

C. Attribute-based encryption (ABE)

ABE allows each ciphertext to be associated with an attribute, and the master-secret key holder can extract a secret key for a policy of these attributes so that a ciphertext can be decrypted by this key if its associated attribute conforms to the policy. However, the major concern in 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.

D. Primitive is proxy re-encryption (PRE)

To delegate the decryption power of some ciphertexts without sending the secret key to the delegate, a useful primitive is proxy re-encryption (PRE). A PRE scheme allows sender to delegate to the server (proxy) the ability to convert the ciphertexts encrypted under her public-key into ones for receiver. The public key and private key in primitive re-encryption scheme does not allow to hack any unauthorized access.

III. PROPOSED METHOD

Our proposed method is "To design 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)."

We solve this problem by introducing a special type of public-key encryption which we call key-aggregate cryptosystem (KAC). In 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(8).

The sizes of ciphertext, public-key, master-secret key and aggregate key in our 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. Figure 3.1 shows the proposed method system Architecture.

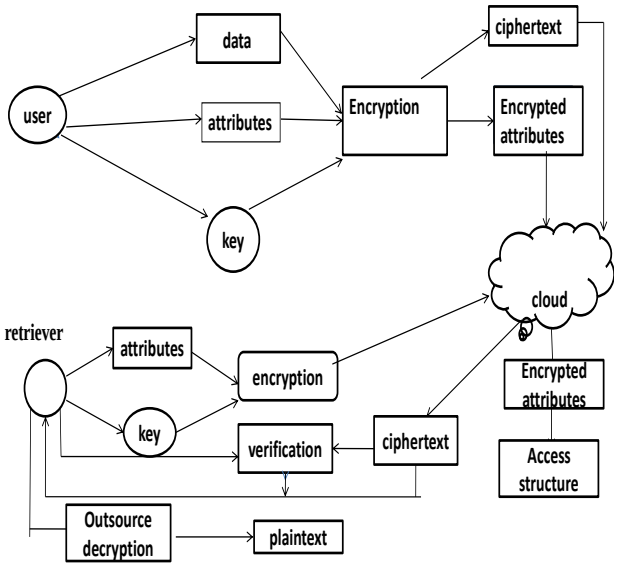

Fig. 3.1: System Architecture

The proposed system design an efficient public-key encryption scheme which supports flexible allocation. In this scheme any subset of the cipher texts (produced by the encryption scheme) is decrypt by a constant-size decryption key (generated by the proprietor of the master-secret key). Constant-size decryption key require pre-defined hierarchical relationship. The fixed hierarchy is used. In that there is only one way in which we can partition the record. If we want to give out access rights based on something else (e.g. based on document type or sensitivity of data) we will have to look at all the low-level categories involved, and give a separate decryption key for each. More number of decryption key was used. Figure 3.2 Key aggregate cryptosystem Which involves generation of secret key.

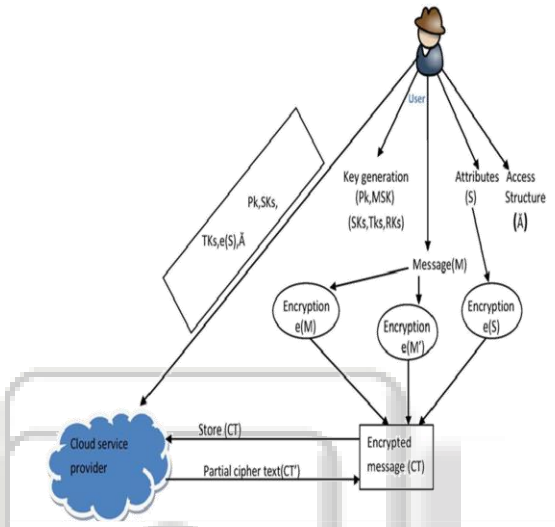

Fig. 3.2: Key Aggregate Cryptosystem

To study how to make decryption key more powerful in the sense that it allows decryption of multiple cipher texts, without increasing its size. To design an efficient public-key encryption scheme which support flexible delegation in the sense that any subset of the ciphertexts is decryptable by a constant-size decryption key .Introduce a special type of public-key encryption which called, key-aggregate cryptosystem (KAC). In 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. Aggregate key can be sent to receiver via a secure e-mail.

IV. METHODOLOGY

A key-aggregate encryption scheme consists of five polynomial-time algorithms as follows. The data owner establishes the public system parameter via Setup and generates a public/master-secret key pair via KeyGen. Messages can be encrypted via Encrypt by anyone who also decides what ciphertext class is associated with the plaintext message to be encrypted. The data owner can use the master-secret to generate an aggregate decryption key for a set of ciphertext classes via Extract. The generated keys can be passed to delegates securely (via secure e-mails or secure devices) Finally, any user with an aggregate key can decrypt any ciphertext provided that the ciphertexts class is contained in the aggregate key via Decrypt. Class is contained in aggregate key via Decrypt.

A. Setup(1;n):

Executed by data owner to setup an account on an untrusted server. On input a security level parameter
And number of cipher classes it outputs the public system parameter param, which is omitted from the input of the other algorithms for brevity. The setup algorithm takes no input other than the implicit security parameter. It outputs the public parameters PK and a master key MK.

B. Key Gen:

Executed by the data owner to randomly generate a public/master-secret key pair (pk; msk). Encrypt(pk; i; m): executed by anyone who wants to encrypt data. On input a public-key pk, an index i denoting the ciphertext class, and a message m, it outputs a ciphertext C. Key Generation(MK,S). The key generation algorithm takes as input the master key MK and a set of attributes S that describe the key. It outputs a private key SK

C. Extract(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 K_S.

D. Encrypt(PK,M, A).

The encryption algorithm takes as input the public parameters PK, a message M, and an access structure A over the universe of attributes. The algorithm will encrypt M and produce a ciphertext CT such that only a user that possesses a set of attributes that satisfies the access structure will be able to decrypt the message. We will assume that the ciphertext implicitly contains A.

E. Decrypt(K_S; S; i; C):

executed by a delegate who received an aggregate key K_S generated by Extract.
A canonical application of KAC is data sharing. 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, by distributing to each authorized user a single and small aggregate key. Here we describe the main idea of data sharing in cloud storage using KAC, illustrated in Figure 2. Suppose Alice wants to share her data m_1; m_2; : : : ; m on the server. She first performs Setup(1 ; n) to get param and execute KeyGen to get the public/master-secret key pair (pk; msk). The system parameter param and public-key pk can be made public and master-secret key msk should be kept secret by Alice. Anyone (including Alice herself) can then encrypt each m_i by C_i = Encrypt(pk; i; m_i). The encrypted data are uploaded to the server.

We take the tree structure as an example. Alice can first classify the ciphertext classes according to their subjects like Figure 3. Each node in the tree represents a secret key, while the leaf nodes represents the keys for individual ciphertext classes. Filled circles represent the

keys for the classes to be delegated and circles circumvented by dotted lines represent the keys to be granted. Note that every key of the non-leaf node can derive the keys of its descendant nodes(9).

Most of these schemes produce keys for symmetric-key cryptosystems, even though the key derivations may require modular arithmetic as used in public-key cryptosystems, which are generally more expensive than "symmetric-key operations" such as pseudorandom function. We take the tree structure as an example. Alice can first classify the ciphertext classes according to their subjects like Fig. 3. Each node in the tree represents a secret key, while the leaf nodes represents the keys for individual ciphertext classes. Filled circles represent the keys for the classes to be delegated and circles circumvented by dotted lines represent the keys to be granted. Note that every key of the non leaf node can derive the keys of its descendant nodes, if Alice wants to share all the files in the "personal" category, she only needs to grant the key for the node "personal," which automatically grants the delegate the keys of all the descendant nodes ("photo," "music").

In general, hierarchical approaches can solve the problem partially if one intends to share all files under a certain branch in the hierarchy. On average, the number of keys increases with the number of branches. It is unlikely to come up with a hierarchy that can save the number of total keys to be granted for all individuals (which can access a different set of leaf-nodes) simultaneously.

V. Advantages and Disadvantages

A. *Advantages*

- A decryption key is more powerful in the sense that it allows decryption of multiple cipher texts, without raising its size.
- The size of master-secret key, cipher text, public-key, and aggregate key in our KAC schemes are all are kept constant size.
- KAC scheme is flexible in the sense that there is, no special relation is required between the classes.
- A canonical application of KAC is efficient data sharing scheme.
- The key aggregation property is especially useful when the delegation key 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 cipher text expansion, by distributing to each authorized user a single, compact, small aggregate key.
- The delegation of decryption can be efficiently implemented with the aggregate key.
- Number of cipher text classes is large.
- It is easy to key management.
- Particular Member can view their messages.

B. *Disadvantages*

- The costs and complexities involved generally increase with the number of the decryption keys to be shared.
- The encryption key and decryption key are different in public key encryption.
- Constant-size decryption key require pre-defined hierarchical relationship.
- The fixed hierarchy is used. In that there is only one way in which we can partition the record.

VI. Results and Discussion

To make the best out of our extended scheme (i.e., to make the key size as small as possible), we suggest that the ciphertext classes for different purposes should be corresponded to different public-keys. This is reason-able in practice and does not contradict our criticism on hierarchical methods that an efficient assignment of hierarchy requires a priori knowledge on what to be shared. Using our example, pk_1 and pk_2 correspond to "personal" and "work"(10).

It is likely to have many sub-categories under either of them but it may not be equally likely to share both of them (if the user does not gossip about office drama with friends and do not expose party photos to colleagues). Another example, say a user's categorization include "music" and "game". One day she becomes a graduate student and needs to publish, and therefore find the new need to add a category "paper", which is probably independent of "music" and "game".

This key extension approach can also be seen as a key update process. In case a secret value is compromised, we can replace the compromised pk_1 with a new key pk_2. The small aggregate key size minimizes the communication overhead for transferring the new key.

VII. Conclusion

How to protect users data privacy is a central question of cloud storage. With more mathematical tools, cryptographic schemes are getting more versatile and often involve multiple keys for a single application. In this article, we consider how to "compress" secret keys in public-key cryptosystems which support delegation of secret keys for different ciphertext classes in cloud storage. No matter which one among the power set of classes, the delegate can always get an aggregate key of constant size. Our approach is more flexible than hierarchical key assignment which can only save spaces if all key-holders share a similar set of privileges.

This compressed key support delegation of secret keys for different cipher text classes in cloud storage. Our approach is more flexible than hierarchical key assignment. The compressed key can only save spaces if all key-holders share a similar set of privileges. A limitation in our work is the predefined bound of the number of maximum ciphertext classes. In cloud storage, the number of ciphertexts usually grows rapidly. So we have to reserve enough ciphertext classes for the future extension. Otherwise, we need to expand the public-key

Reference

[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," in Applied Cryptography and Network Security - ACNS 2012, ser. LNCS, vol. 7341. Springer, 2012, pp. 526–543.

[2] L. Hardesty, "Secure computers aren't so secure," MIT press, 2009, http://www.physorg.com/news176107396.html.
[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, 2013.
[4] 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.
[5] S. S. M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R. H. Deng, "Dynamic Secure Cloud Storage with Provenance," in Cryptography and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday, ser. LNCS, vol. 6805. Springer, 2012, pp. 442–464.
[6] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," in Proceedings of Advances in Cryptology - EUROCRYPT '03, ser. LNCS, vol. 2656. Springer, 2003, pp. 416–432.
[7] 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.
[8] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records," in Proceedings of ACM Workshop on Cloud Computing Security (CCSW '09). ACM, 2009, pp. 103–114.
[9] F. Guo, Y. Mu, Z. Chen, and L. Xu, "Multi-Identity Single-Key Decryption without Random Oracles," in Proceedings of Information Security and Cryptology (Inscrypt '07), ser. LNCS, vol. 4990. Springer, 2007, pp. 384–398.
[10] www.google.com

