Multi User Data Sharing with Aggregated Keys in Clouds

M. Sasidharan[1] N. S Nithya[2]
[1]P.G. Student [2]Assistant Professor
[1,2]Department of Computer Science & Engineering
[1,2]K.S.R.College of Engineering, Namakkal, Tamilnadu, India

Abstract— Data sharing is an important functionality in cloud storage. Data owner maintains the shared data under cloud data centers. Semi-trusted third party servers are used to manage medical records. Personal Health Record (PHR) is used to create, manage and control their personal health data in one place through the web. Patient data can be shared with healthcare providers, family members and friends. Sensitive details are managed in medical records. Data owner decides the access privileges for the medical records. The Existing Public-key cryptosystems produce constant-size ciphertexts with efficient delegation of decryption rights for any set of ciphertexts. One can aggregate any set of secret keys and make them as compact as a single key. The secret key holder can release a constant-size aggregate key for flexible choices of ciphertext set in cloud storage. This compact aggregate key can be conveniently sent to others or be stored in a smart card with very limited secure storage. The key aggregation system is divided into five major steps. They are Setup, KeyGen, Encrypt, Extract and Decrypt. The setup process is designed to creating an account on an untrusted server by the data owner. The keygen process is executed by the data owner to randomly generate a public/master-secret key pair. Encrypt process is executed by anyone using public key and index value. Extract is carried out by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegate. Decrypt is executed by a delegate who received an aggregate key KS generated by Extract. Patient controlled encryption scheme is designed using Key Aggregate cryptosystem (KAC).The proposed key aggregate cryptosystem is enhanced with boundary less ciphertext classes. The system is improved with device independent key distribution mechanism. The key distribution process is enhanced with security features to protect key leakage. The key parameter transmission process is integrated with the ciphertext download process.

Key words: Cloud storage, data sharing, key-aggregate encryption, patient-controlled encryption

I. INTRODUCTION

Research in cloud computing is receiving a lot of attention from both academic and industrial worlds. In cloud computing, users can outsource their computation and storage to servers using Internet. This frees users from the hassles of maintaining resources on-site. Clouds can provide several types of services like applications, infrastructures and platforms to help developers write applications.

Much of the data stored in clouds is highly sensitive, for example, medical records and social networks. Security and privacy are, thus, very important issues in cloud computing. In one hand, the user should authenticate itself before initiating any transaction and on the other hand, it must be ensured that the cloud does not tamper with the data that is outsourced. User privacy is also required so that the cloud or other users do not know the identity of the user.

The cloud can hold the user accountable for the data it outsources and likewise, the cloud is itself accountable for the services it provides. The validity of the user who stores the data is also verified. Apart from the technical solutions to ensure security and privacy, there is also a need for law enforcement.

Efficient search on encrypted data is also an important concern in clouds. The clouds should not know the query but should be able to return the records that satisfy the query. This is achieved by means of searchable encryption. The keywords are before sent to the cloud encrypted and the cloud returns the result without knowing the actual keyword for the search. The problem here is that the data records should have keywords associated with them to enable the search. The correct records are returned only when searched with the exact keywords.

II. RELATED WORK

This section we compare our basic KAC scheme with other possible solutions on sharing in secure cloud storage. Cryptographic Keys for a Predefined Hierarchy, Compact Key in Symmetric-Key Encryption, Compact Key in Identity-Based Encryption (IBE) and Other Encryption Schemes are analyzed in the literature.

A. Predefined Hierarchy based Cryptographic Keys:

We start by discussing the most relevant study in the literature of cryptography/security. Cryptographic key assignment schemes [7] 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. Just granting the parent key implicitly grants all the keys of its descendant nodes. Sandhu proposed a method to generate a tree hierarchy of symmetric-keys by using repeated evaluations of pseudorandom function/blockcipher on a fixed secret. The concept can be generalized from a tree to a graph. More advanced cryptographic key assignment schemes support access policy that can be modeled by an acyclic graph or a cyclic graph. 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.

B. Symmetric-Key Encryption using Compact Keys:

Motivated by the same problem of supporting flexible hierarchy in decryption power delegation, Benaloh et al. presented an encryption scheme which is originally proposed for concisely transmitting large number of keys in broadcast scenario [8]. The construction is simple and we briefly review its key derivation process here for a concrete description of what are the desirable properties we want to achieve. The derivation of the key for a set of classes is as

follows: A composite modulus N = p. q is chosen where p and q are two large random primes. A master-secret key Y is chosen at random from ZZ_ N. Each class is associated with a distinct prime e_i. All these prime numbers can be put in the public system parameter. A constant-size key for set S' can be generated (with the knowledge of ϕ (N) as

$$ks = Y^{1/\prod_{i\in S'}(e_i)} \bmod N.$$

For those who have been delegated the access rights for S where S' $\subset$ S, ks' can be computed by

$$k_S^{\prod_{i\in S/S'}(e_j)}$$

Finally, we note that there are schemes which try to reduce the key size for achieving authentication in symmetric-key encryption. Sharing of decryption power is not a concern in these schemes.

C. Identity-based Encryption (IBE) using Compact Keys:

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. Guo et al. tried to build IBE with key aggregation. One of their schemes assumes random oracles but another does not. In their schemes, key aggregation is constrained in the sense that all keys to be aggregated must come from different "identity divisions." While there are an exponential number of identities and thus secret keys, only a polynomial number of them can be aggregated. Most importantly, their key-aggregation comes at the expense of O(n) sizes for both ciphertexts and the public parameter, where n is the number of secret keys which can be aggregated into a constant size one. This greatly increases the costs of storing and transmitting ciphertexts, which is impractical in many situations such as shared cloud storage. As we mentioned, our schemes feature constant ciphertext size, and their security holds in the standard model.

D. Other Encryption Schemes:

Attribute-based encryption (ABE) [9] 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. 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 (e.g., [6]). To delegate the decryption power of some ciphertexts without sending the secret key to the delegatee, a useful primitive is proxy re-encryption (PRE). A PRE scheme allows Alice to delegate to the server the ability to convert the ciphertexts encrypted under her public-key into ones for Bob. PRE is well known to have numerous applications including cryptographic file system.

III. DATA SHARING AND SECURITY IN CLOUDS

Cloud storage is gaining popularity recently. In enterprise settings, we see the 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 apply for 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 a mobile phone in any corner of the world.

Data privacy is a traditional way to ensure it is to rely on the server to enforce the access control after authentication (e.g., [1]), which means any unexpected privilege escalation will expose all data. In a shared-tenancy cloud computing environment, things become even worse. 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 coresident 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.

A cryptographic solution, for example, [5], 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. 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, 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. Finding an efficient and secure way to share partial data in cloud storage is not trivial. Below we will take Dropbox as an example for illustration.

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 cryptographic functions 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. Specifically, our problem statement is "To design an efficient public-key encryption scheme which supports flexible delegation in the sense that any subset of the ciphertexts is decryptable by a constant-size decryption 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.

IV. PROBLEM STATEMENT

Public-key cryptosystems produce constant-size ciphertexts with efficient delegation of decryption rights for any set of ciphertexts. One can aggregate any set of secret keys and make them as compact as a single key. The secret key holder can release a constant-size aggregate key for flexible choices of ciphertext set in cloud storage. This compact aggregate key can be conveniently sent to others or be stored in a smart card with very limited secure storage. The key aggregation system is divided into five major steps. They are Setup, KeyGen, Encrypt, Extract and Decrypt. The setup process is designed to creating an account on an untrusted server by the data owner. The keygen process is executed by the data owner to randomly generate a public/master-secret key pair. Encrypt process is executed by anyone using public key and index value. Extract is carried out by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegate.

Decrypt is executed by a delegate who received an aggregate key KS generated by Extract. Patient controlled encryption scheme is designed using Key Aggregate cryptosystem (KAC). The following problems are identified from the security scheme for cloud data services. Predefined boundary for the maximum number of secret keys, Limited size for ciphertext classes, Trusted hardware is required for key distribution and Key leakage possibility is high.

V. KEY-AGGREGATE CRYPTO SYSTEM

We first give the framework and definition for key aggregate encryption. Then we describe how to use KAC in a scenario of its application in cloud storage. 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-secret3 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 finally; any user with an aggregate key can decrypt any ciphertext provided that the ciphertexts class is contained in the aggregate key via Decrypt.

- Setup(1,n): executed by the data owner to setup an account on an untrusted server. On input a security level parameter 1 and the number of ciphertext classes n, it outputs the public system parameter param, which is omitted from the input of the other algorithms for brevity.
- KeyGen: executed by the data owner to randomly generate a public/master-secret key pair (p_k ,ms_k).
- Encrypt(p_k I,m): executed by anyone who wants to encrypt data. On input a public-key p_k , an index I denoting the ciphertext class, and a message m, it outputs a ciphertext C.
- Extract(msk, S): executed by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegatee. 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 KS.
- Decrypt(KS, S, i) C: executed by a delegatee who received an aggregate key KS generated by Extract. On input KS, the set S , an index i denoting the ciphertext class the ciphertext C belongs to, and C, it outputs the decrypted result m if i ∈ S.

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.

We describe the main idea of data sharing in cloud storage using KAC, illustrated. Suppose Alice wants to share her data m_1,m_2 , . . .,m on the server. She first performs Setup(1^{λ} ,n) to get param and execute KeyGen to get the public/master-secret key pair (p_k , ms_k). The system parameter param and public-key p_k can be made public and master-secret key ms_k should be kept secret by Alice. Anyone can then encrypt each mi by C_i = Encrypt. The encrypted data are uploaded to the server.

With param and p_k , people who cooperate with Alice can update Alice's data on the server. Once Alice is willing to share a set S of her data with a friend Bob, she can compute the aggregate key KS for Bob by performing Extract(ms(msk,S). Since KS is just a constant-size key, it is easy to be sent to Bob via a secure e-mail. After obtaining the aggregate key, Bob can download the data he is authorized to access. That is, for each i$\in S$, Bob downloads C_i from the server. With the aggregate key KS, Bob can decrypt each C_i by Decrypt (Ks, S, i, C_i) for each $i \in S$.

VI. SECURED MULTI USER DATA SHARING IN CLOUDS

The system is designed to share data with access control based security model described in Fig. 6.1. Individual and role based access policy model is used in the system. Data integrity verification process is integrated with the system. The system is divided into six major modules. They are Cloud Data Center, Data Owner, Key Management, Policy Controller, Data Security Process and Client.

The cloud data center manages the data owner and user accounts. Data owner maintains the shared data and user access privileges. Key generation and distribution operations are handled in key management module. User access permissions are managed under the policy controller module. Data security module is used to protect the shared data uploading process. The client module is used to access the shared data values provided by the data owner.

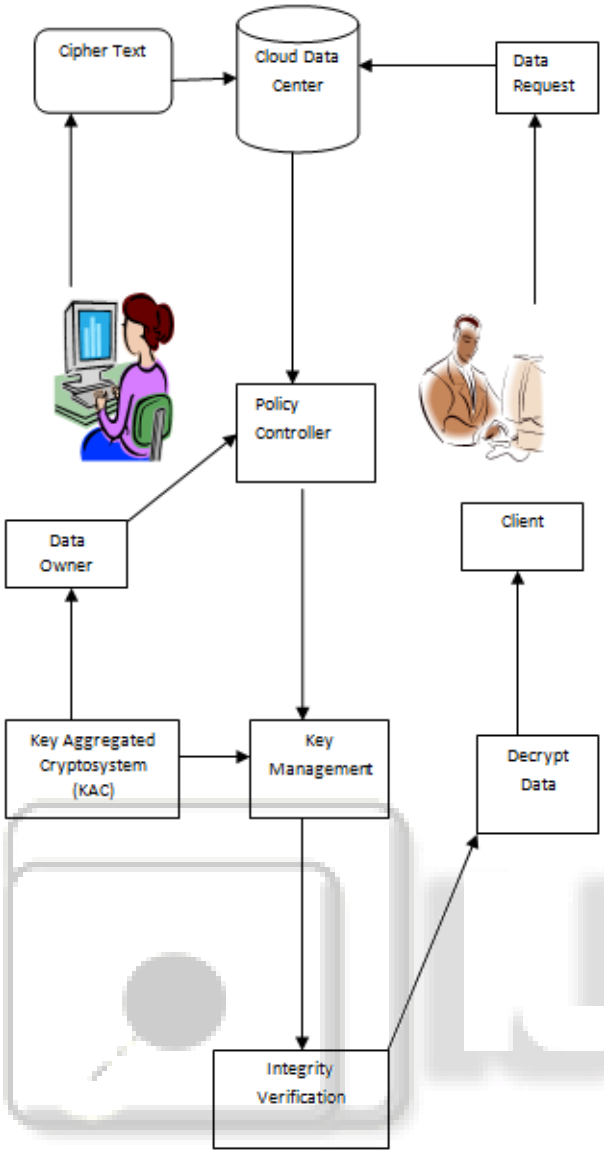

Fig. 6.1: Secured Multi User Data Sharing in Clouds

A. Cloud Data Center:

Cloud data center maintains the data owner and client details. Cloud data center allocates storage space for the data owners. Data center manages the shared data uploaded by the data owner. Client requests are processed by the data center.

B. Data Owner:

Data owner uploads their data in to the cloud data center. Data owner creates different user account to access their data values. User accounts are managed with access level based groups. Policy and key management operations are carried out by the data owner.

C. Key Management:

Key generation and distribution tasks are handled in the key management process. Master key and multiple secret keys are generated by the data owner. Master key is used for the encryption process and the secret keys are used for the decryption process. Compact key is constructed with the aggregated secret key values.

D. Policy Controller:

User access permissions are maintained in the policy controller. User level and group level access rights are issued by the data owner. Shared data values are provided with reference to the user access rights. Secret keys are assigned for the users with reference to their access levels.

E. Data Security Process:

The data security process protects the uploaded data. Data owner encrypts and upload the data to the data center. Multiple cipher text values are maintained for each shared data item. Data integrity verification is performed in the data upload and downloads process.

F. Client:

Client application is designed to access the shared data provided by the data owner. Encrypted data is downloaded from the data center. Decryption process is carried out using the secret key associated with the client access levels. The client data access details are reported to the data owner.

VII. Conclusion

The cloud data storages are used to share data from different data owners. Key Aggregated Cryptosystem (KAC) is used to share data with multiple users with security. The KAC scheme is enhanced with multiple ciphertexts classes. The system is improved with secured key distribution scheme to protect key leakages. Patient-controlled encryption scheme is applied for flexible hierarchy. Policy based security model is used to provide data security with access control mechanism. Distributed storage is supported with data encryption and access control mechanism. Device independent key distribution model is used in the system. Secured key distribution process is adopted to protect the key values.

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, 2012.

[2] L. Hardesty, Secure Computers Aren't so Secure. MIT press, http://www.physorg.com/news176107396.html, 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, 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 Int'l Conf. Distributed Computing Systems, 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] T. Okamoto and K. Takashima, "Achieving Short Ciphertexts or Short Secret-Keys for Adaptively Secure General Inner-Product Encryption," Proc.

10th Int'l Conf. Cryptology and Network Security, pp. 138-159, 2011.
[7] G. Ateniese, A.D. Santis, A.L. Ferrara and B. Masucci, "Provably-Secure Time-Bound Hierarchical Key Assignment Schemes," J. Cryptology, vol. 25, no. 2, pp. 243-270, 2012.
[8] J. Benaloh, "Key Compression and Its Application to Digital Fingerprinting," technical report, Microsoft Research, 2009.
[9] M. Chase and S.S.M. Chow, "Improving Privacy and Security in Multi-Authority Attribute-Based Encryption," Proc. ACM Conf. Computer and Comm. Security, 2009

