Secured Group Data Sharing Over Cloud by Using Key Aggregate and Searchable Techniques

Patil Snehal

Department Of Computer Science, RMD Sinhgad College of Engineering, University of Pune, Maharashtra, India

Abstract: ***Cloud computing has given the users the accessibility to deploy number of files to the centralized cloud and share those with number of users. The flexibility of cloud computing always comes with the hurdles of security concerns. The data owner always needs to encrypt the files before uploading and it must decrypt before end users. This system needs secure storage of keys, but as files gets increased in number keys management becomes complex. We have proposed the system called as (KASE). The system proposes aggregate key for file sharing in groups and searchable encryption. We have observed that to create trapdoors manually for specific files it becomes very tedious and hence we have applied the TF-IDF technique to avoid manual job.***

Keywords: Searchable encryption, Key aggregation, cloud computing, group data sharing

1. Introduction

Cloud systems can be used to enable data sharing capabilities and this can provide an abundant of benefits to the user. There is currently a push for IT organizations to increase their data sharing efforts. In enterprise settings, demand for data outsourcing is increased today. Data outsourcing should be assists in the strategic management of corporate data. This scheme is also used as a core technology behind many online services. These online services used for online application. Currently this scheme was easy to apply for free accounts for mail, photograph album, sharing of file with storage size more than 25GB. Together by using the current wireless technology, cloud users can access almost all of their files, directories and emails by a mobile phone in any corner of the world.

Some of major requirements of secure data sharing in the Cloud are as follows. Firstly the data owner should be able to specify a group of users that are allowed to view his or her data. Any member within the group should be able to gain access to the data anytime, anywhere without the data owner's intervention. No-one, other than the data owner and the members of the group, should gain access to the data, including the Cloud Service Provider. The data owner should be able to add new users to the group. The data owner should also be able to revoke access rights against any member of the group over his or her shared data. No member of the group should be allowed to revoke rights or join new users to the group. One trivial solution to achieving secure data sharing in the Cloud is for the data owner to encrypt his data before storing into the Cloud, and hence the data remain information-theoretically secure against the Cloud provider and other malicious users. When the data owner wants to share his data to a group, he sends the key used for data encryption to each member of the group. Any member of the group can then get the encrypted data from the Cloud and decrypt the data using the key and hence does not require the intervention of the data owner. However, the problem with this technique is that it is computationally inefficient and places too much burden on the data owner when considering factors such as user revocation. When the data owner revokes access rights to a member of the group, that member should not be able to gain access to the corresponding data. Since the member still has the data access key, the data owner has to re-encrypt the data with a new key, rendering the revoked member's key useless. When the data is re-encrypted, he must distribute the new key to the remaining users in the group and this is computationally inefficient and places too much burden on the data owner when considering large group sizes that could be in excess of millions of users. Hence this solution is impractical to be deployed in the real-world for very critical data such as business, government and medical related data. While considering data privacy, we cannot rely on traditional technique of authentication, because unexpected privilege escalation will expose all data. Solution is to encrypt data before uploading to the server with users own key. Data sharing is again important functionality of cloud storage, because user can share data from anywhere and anytime to anyone. For example, organization may grant permission to access part of sensitive data to their employees. But challenging task is that how to share encrypted data. Traditional way is user can download the encrypted data from storage, decrypt that data and send it to share with others, but it loses the importance of cloud storage.

2. Literature Survey

2.1 Introduction

This section provides the purpose of the feasibility study, the background of the proposed project, the methodology used for performing the study, and any reference materials used in conducting the feasibility study for the project. To check the feasibility of system, two methodologies are used: Surveying and Brain Storming. Literature survey by studying various IEEE papers and other related reference material is conducted. The feasibility study has conducted to determine projects viability. The results of this study will be used to make a decision whether or not to proceed with the project.

2.2 Existing Methodologies

In recent years, a growing number of researchers have been engaging in studying search-able encryption and various efficient search schemes over encrypted cloud data have been proposed. Many technical schemes related to cloud computing service are proposed by researchers. Cryptosystem for ne-grained sharing of encrypted data was introduced in[2]. This scheme was called Key-Policy Attribute-Based

Encryption (KPABE). In our cryptosystem, cipher texts are labeled with sets of attributes and private keys are associated with access structures that control which cipher texts a user is able to decrypt [2]. Multi-Identity Single-Key Decryption without Random Oracles, produce Multi-Identity Single-Key Decryption (MISKD). It is an Identity-Based Encryption (IBE) system where a private decryption key can map multiple public keys (identities). More exactly, in MISKD, a single private key can be used to decrypt multiple cipher texts encrypted with different public keys associated to the private key [3]. Dynamic and Efficient Key Management for Access Hierarchies, This Method has the following properties:

1) Only hash functions are used for a node to derive a descendant's key from its own key;
2) The space complexity of the public information is the same as that of storing the hierarchy;
3) The private information at a class consists of a single key associated with that class;

Updates (revocations, additions, etc.) handled locally in the hierarchy; the scheme is provably secure against collusion; and Key derivation by a node of its descendant's key is bounded by the number of bit operations linear in the length of the path between the nodes [4]. The dynamic scheme achieve a worst and average-case number of bit operations for key derivation that exponentially better than the depth of a balanced hierarchy.

3. Problem Definition

Cloud computing has given the users the accessibility to deploy number of files to the centralized cloud and share those with number of users. The flexibility of cloud computing always comes with the hurdles of security concerns. The data owner always needs to encrypt the files before uploading and it must decrypt before end users. This system needs secure storage of keys, but as files gets increased in number keys management becomes complex. We have proposed the system called as (KASE) . The system proposes aggregate key for file sharing in groups and searchable encryption. We have observed that to create trapdoors manually for specific files it becomes very tedious and hence we have applied the TF-IDF / Cosine similarity technique to avoid manual job.

4. Methodology / Approach

In this section first describe the general problem, and then define a generic framework for key aggregate searchable encryption (KASE) and provide requirements for designing a valid KASE scheme. We aim to propose the novel approach of key-aggregate searchable encryption (KASE) that satisfies several functional and security requirements. So that, build a key aggregate system which can be securely share with the groups of users. After then apply attribute based broadcast encryption system for encryption of files before uploading to the cloud. Also apply auto keywords extraction technique mainly TF-IDF to create the trapdoors for file searching. Encrypted cipher text and trapdoors will be uploaded to the cloud Access control technique will be applied to give access to only authorized users. We can perform data sharing with clouds using advanced framework of KASE algorithm. It has been is composed of seven algorithm for security parameter setup, key generation, encryption, key extraction, trapdoor generation, trapdoor adjustment, and trapdoor testing. Further describe this system in details; we describe its main work flows. System setup: When an organization submits a request, the cloud will create a database containing above four tables, assign a groupID for this organization and insert a record into table company. Moreover, it assigns an administrator account for the manager. Then, the group data sharing system will work under the control of manager. To generate the system parameters params, manager runs the algorithm KASE. Setup and updates the eld parameters in Table Company. User registration: When adding a new member, the manager assigns memberID, membeName, password and a key pair generated by any public key encryption (PKE) scheme for him, then stores the necessary information into the table member. A user's private key should be distributed through a secure channel.

User login: Like most popular data sharing products (e.g., Dropbox and citrix), our system relies on password verification for authenticating users. To further improve the security, multi-factor authentication or digital signatures may be used when available.

Data uploading: To upload a document, the owner runs KAE. Encrypt to encrypt the data and KASE. Encrypt to encrypt the keyword cipher texts, then uploads them to the cloud. The cloud assigns a docID for this document and stores the encrypted data in the path file Path, then inserts a record into the table docs. In addition, the owner can encrypt the keys using his/her private key and store them into the table
docs.

Data sharing: To share a group of documents with a target member, the owner runs KAE. Extract and KASE. Extract to generate the aggregate keys, and distributes them to this member, then inserts/updates a record in table sharedDocs. If the shared documents for this member are changed, the owner must re-extract the keys and update the eld docIDSet in table sharedDocs.

Keyword Search: To retrieve the documents containing an expected keyword, a member runs KASE. Trapdoor to generate the keyword trapdoor for documents shared by each owner, then submits each trapdoor and the related owners identity OwnerID to the cloud. After receiving the request, for each trapdoor, the cloud will run KASE. Adjust the trapdoor for each document in the docIDSet and run KASE. Test to perform keyword search. Then, the cloud will return the encrypted documents which contains the expected keyword to the member
.

Data Retrieving: After receiving the encrypted document, the member will run KAE. Decrypt to decrypt the document using the aggregate key distributed by the documents owner.

4.1 Major Constraint

A KASE scheme should satisfy three functional requirements as Compactness Searchability and Delegation. Also there is problem with federated clouds that can attracted a lot of attention nowadays.

4.2 Approaches for solving the problem and Efficiency issues

Our system contains four different phases. First define a general framework of key aggregate searchable encryption (KASE) composed of seven polynomial algorithms for security parameter setup, key generation, encryption, key extraction, trapdoor generation, trap-door adjustment, and trapdoor testing. We then describe both functional and security requirements for designing a valid KASE scheme. Then instantiate the KASE framework by designing a concrete KASE scheme. After providing detailed constructions for the seven algorithms, analyze the efficiency of the scheme, and establish its security through detailed analysis. Discuss of various practical issues in building an actual group data sharing system based on the proposed KASE scheme, and evaluate its performance. The evaluation confirms our system can meet the performance requirements of practical applications. Both analysis and evaluation results confirm that our work can provide an effective solution to building practical data sharing system based on public cloud storage.

Broadcast Encryption

In a broadcast encryption (BE) scheme, a broadcaster encrypts a message for some subset S of users who are listening on a broadcast channel [1]. Any user in S can use his/her private key to decrypt the broadcast. A BE scheme can be described as a tuple of three polynomial-time algorithms BE = (Setup,Encrypt,Decrypt) as follows:

Setup (1; n): This algorithm is run by the system to set up the scheme. It takes as input a security parameter 1 and the number of receivers n, outputs n private keys [d1;..; dn] and a public key pk.

Encrypt (pk; S): This algorithm is run by the broadcaster to encrypt a message for a subset of users. It takes as input a public key pk and a subset of users S1; ..; n, outputs a pair (Hdr,K), where Hdr is called the header and K is a message encryption key which is encapsulated in Hdr. Often refer to Hdr as the broadcast cipher-text. For a concrete message, it will be encrypted by K and broadcasted to the users in S.

Decrypt (pk, S, I, di, Hdr): This algorithm is run by the user to decrypt the received messages. It takes as input a public key pk, a subset of users S1; ..; n, a user id i21;; n, the private key di for user i and a header Hdr, outputs the message encryption key K or the failure symbol ?. The K will be used to decrypt the received messages.

To ensure the system to be correct, it is required that, for all S1; ..; n and all i2S, if (pk; (d1; ..; dn)RSetup(1; n)and(Hdr;K)REncrypt(pk; S)), then Decrypt(pk; S; i; di;Hdr) = K.

Searchable Encryption: Searchable encryption schemes fall into two categories, i.e., searchable symmetric encryption (SSE) and public key encryption with keyword search (PEKS). Both SSE and PEKS can be described as the tuple SE = (Setup, Encrypt, Trapdoor, and Test):

Setup (1): This algorithm is run by the owner to set up the scheme. It takes as input a security parameter 1, and outputs the necessary keys.

Encrypt (k;m): This algorithm is run by the owner to encrypt the data and generate its keyword cipher texts. It takes as input the data m, owners necessary keys including searchable encryption key k and data encryption key, outputs data cipher text and keyword cipher texts Cm.

Trpdr(k;w): this algorithm is run by a user to generate a trapdoor Tr for a key-word w using key k.

Test (Tr; Cm): This algorithm is run by the cloud server to perform a keyword search over encrypted data. It takes as input trapdoor Tr and the keyword cipher texts Cm, outputs whether Cm contains the specified keyword.

TF-IDF: Auto keywords extraction technique to create the trapdoors for file searching. For correctness, it is required that for a message m containing keyword w and a searchable encryption key k, if (Cm_Encrypt(k;m)) and (Tr_Trpdr(k;w)), then Test(Tr;Cm) = true.

5. Result and Discussion

The purpose of this document is to provide documentation for the design and implementation of the New Algorithm for sharing data over clouds using aggregate key. Both high-level requirements and implementation details are documented here to ensure successful completion of the project and continuity for future project development. This document is intended to be a detailed design supplement to the Terms of Reference for the development of applications. This document will provide detailed specifications for designing, implementing, and configuring application for KASE search results using TF-IDF for fitting trapdoor algorithm, but will not include end-user documentation. The intended audience of this specification includes project managers or developers or research oriented professional may use or extend this project in the future. Details in this document may be helpful for technically-oriented end-users of the cloud application.

6. Conclusions

Considering the practical problem of privacy preserving data sharing system based on public cloud storage which requires a data owner to distribute a large number of keys to users to enable them to access his/her documents, we for the rst time propose the concept of key-aggregate searchable encryption (KASE) and construct a concrete KASE scheme. Both analysis and evaluation results confirm that our work can provide an effective solution to building practical data sharing system based on public cloud storage.

7. Future Scope

In future, at extending the approach to reduce the number of trapdoors under multiowners setting and also to provide the solution for KASE in the case of federated clouds.

References

[1] Baojiang Cui, Zheli Liu and Lingyu Wan "Key-Aggregate Searchable Encryption (KASE) for Group Data Sharing via Cloud Storage", IEEE TRANSACTIONS ON COMPUTERS VOL: PP NO: 99 YEAR 2015

[2] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine Grained Access Control of Encrypted data," in Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS 06) ACM, 2006, pp. 89-98

[3] 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

[4] 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

[5] C. Chu, S. Chow, W. Tzeng, et al., "Key Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage", IEEE Transactions on Parallel and Distributed Systems, 2014, 25(2):468-477