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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 2, February 2015

Cryptographic Algorithms for Efficient and Secure Data Sharing in Cloud Storage

Shweta. P. Tenginkai[1], Vani K. S[2]

Student, Department of CSE, Acharya Institute of Technology, Bangalore, India[1]

Assistant Professor, Department of CSE, Acharya Institute of Technology, Bangalore, India[2]

Abstract: Cloud storage is a model of data storage, whose demand is greatly increasing. The data sharing is one of the concerning functionality in today's world of information as it involves security, efficiency and flexibility as their important aspects. Various schemes and methodologies are implemented to make data sharing more effective. With the introduction of encryption and decryption schemes, the storing, sharing and securing of data became rampant. The storing of these ciphertexts and the decryption keys is one of the major issues. There is a need for a mechanism which can minimize the cost of storing these ciphertexts and keys in a secured way. In this paper, we explore the various encryption schemes which were proposed to solve this problem. In this era of information, where there is presence of rich data, the true value lies in sharing, securing and storing them. Protecting users's data privacy is one of the critical goal of cloud storage. The present research efforts concentrates more on aggregation of these keys into a single aggregate key which will in turn reduce the burden on the network overhead.

Keywords: Cloud storage, data sharing, ciphertexts, encryption schemes

I. INTRODUCTION

Cloud computing has become a widely accepted paradigm for providing services over the internet. With the increasing popularity of cloud storage, the risks for security, data integration, confidentiality of data are implicity increasing. Therefore, the cloud provider must consider the security and confidentiality as the challenging factors for data sharing functionality.The care has to be taken to protect data, as cloud storage is storing of the data remotely which is regulated by third party. The third party takes the responsibility for keeping data accessible and available to users all the time. In today's world, it has become easy to go for free accounts to upload or store the data, photos, files or folders with storage capacity more than 25GB. Along with the fast growing internet, users can access and utilize all their files and mails from any place in the world. Instead of storing the data into the hard drive, user can save the data on the cloud which makes him avail all the data accessible for him from any corner of the world using internet. But considering the privacy of data, the traditional techniques for for authentication are not reliable, because the unavoidable privilege escalation will disclose the confidentiality of data. For protecting the confidentiality of the data stored in cloud storage, the care has to be taken for encrypting those data before uploading them on to the cloud by using some or the other cryptographic algorithms. The users are encouraged to encrypt their data before uploading them on the cloud by their own keys whenever the user is not satisfied with trusting the security of the Virtual Machines or the technical team.

In modern cryptography, encryption keys obtained are of two categories, symmetric and asymmetric (public) key. The public key encryption tends to be much more secured as it involves combination of two different keys, public and private key respectively. This gives more flexibility for various applications.

A. Need for Key Aggregation

Key aggregation plays an important role in handling the overhead on networks. With the increase in usage of different devices and systems, the traffic on networks is increasing. Considering a scenario where a particular user Alice wants to send an access key to her friend Bob, who wants to access some of the files. Alice has encrypted those files before uploading them onto the cloud. Then Alice can send an aggregate key of these corresponding secret keys of the various files using which Bob can decrypt them. Here, the burden on network is reduced as the problem of sending all the corresponding keys is replaced by sending just a single aggregate key. The expenses of having a tamper proof storage is usually high. The cost of secured storage for storing these secret keys is also reduced by storing the aggregate key due to its compact size.

In this paper, we will discuss the various cryptographic algorithms.

II. LITERATURE SURVEY

A. Hierarchical Model

One of the ways of key management scheme is the access hierarchy from [4], which consists of a set of partially ordered classes (represented as directed graph).This scheme solves the problem of obtaining the key or access by limiting it only for hash functions to obtain the access to descendant class. It takes care of the security and privacy, the space complexity is same as that of storing hierarchy. The problem of key management in access hierarchy is resolved to some extent. The ciphertext size obtained is constant; decryption key size depends on the hierarchy which is almost non-constant. The problem with this model is, the access to all the descendant classes can be obtained if any user who is granted the access (i.e. a

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 2, February 2015

key) for a certain class. On average, the number of keys involved increase with the count of branches, so it does not solve the problem completely.

B. Multi Identity Single-Key Decryption without Random Oracles

The Multi-Identity Single-Key Decryption (MISKD) is an Identity-Based Encryption (IBE) system from [6], where multiple public keys i.e. identities can be mapped using single private decryption key. A single private key is used to decrypt multiple ciphertexts which are encrypted with different public keys linked with private key. The aggregation is limited to some extent. It is a convenient system for managing and handling many private keys to the users in standard IBE. The encryption used is public-key encryption type and is highly secured in selective-ID model. The decryption key size is constant but the ciphertext size is not constant. The decryption in this scheme is highly efficient.

C. Multi Identity Single-Key Decryption using Random Oracles

It is similar to paper [6], but this paper [7] assumes random oracles and paper [6] does not. Key aggregation is limited to some extent as it involves aggregation key only if they belong to different identity divisions. The Identity-Based-Encryption (IBE) mechanism allows a data sender to encrypt the data to an identity without accessing his public key certificate, which solves the problem of certificate transmission overhead. This feature of IBE to carry out public key encryption without certificates makes it suitable for many practical applications. One of the common features of all these basic schemes is that they consider identities as a string of characters. There are an exponential number of identities and therefore secret keys, and aggregation of only a polynomial number of them is possible. Their Key-aggregation is possible at the expense of O(n) sizes for both the public parameter and ciphertexts, where n is the number of secret keys aggregated into a constant size key. This in turn drastically increases the cost of storage and transmission of ciphertexts which is not practically possible in many scenarios such as shared cloud storage.

D. Fuzzy Identity-Based Encryption Scheme

In this Fuzzy IBE scheme [9], there is a single compact secret key which decrypts ciphertexts encrypted by different identities. This scheme allows for error tolerance between the identity of private key and public key which is used to encrypt a ciphertext. The two practical applications of Fuzzy- IBE encryption using biometrics and attribute-based encryption are described. It uses set overlap as the distance metric between identities. Interesting feature about this scheme is that it hides the public key that is used to encrypt the ciphertext. The number of group elements in public parameters grows linearly with the maximum number of attributes, which can describe an encryption identity. The number of group elements which consists of user's private key grows linearly with the number of attributes associated with its identity. These numbers of group elements in ciphertext grows in linear fashion with the size of identity that are being encrypted to. These are limited to certain metric space, and not for an arbitrary set of identities. The problem of IBE with compact key is non constant ciphertext size. The encryption type used here is public key system.

E. Attribute Based Encryption Schemes

Attributes play a very critical role in Attribute-based Encryption (ABE) scheme. Attributes have been utilized to generate public key for the encryption of data and been used as an access policy to control user's access. In this [2] paper, five different ABE schemes are surveyed, described and compared- Attribute- based Encryption (ABE), Key-Policy Attribute-based Encryption (KP-ABE), Ciphertext-policy (CP-ABE), ABE with non-monotonic access structure, and Hierarchical Attribute-based Encryption (HABE). These schemes are grouped according to their access policy. KP-ABE is the access structure in user's private key and CP-ABE is the access structure in encrypted data. These schemes do not satisfy user accountability. If any new user wants to access data and his attributes are not present in the access structure, then these encrypted data will be re-generated. The access structure is pre-defined in these schemes. As these schemes are encrypted with attributes, a data owner is supposed to predefine these attributes that would be used, regardless of the number of users in the system. The collusion attacks are avoided as every attribute has its public key, secret key and the random polynomial, hence different users cannot combine their attributes to obtain the data. These schemes have the authority which is preferably suitable for the private cloud environments. These schemes almost cannot satisfy the criteria of scalability and user accountability except HABE. They tend to reduce the communication overhead, and provide a fine-grained access control.

F. Attribute Based Encryption for Fine-Grained Access Control

In this cryptosystem [8], ciphertexts are labeled with sets of attributes and private keys are linked with access structures that manage which ciphertexts a user is able to decrypt. Here, the data is stored on the server in an encrypted format, whereas the different users are allowed to decrypt the different chunk of data according to the access rights given to them as per security policy. This solves the problem of depending on the storage server for avoiding unauthorized data access. The Secret-Sharing Schemes (SSS) are used for dividing the secret among the number of parties. The useful data or information sent to a party is called the share (of the secret) for that party. Some access structure which describes the sets of the parties who must reconstruct the secret by using their shares is realized by every SSS. It supports delegation of private keys which includes Hierarchical Identity-Based Encryption (HIBE). It mainly focuses on collusion-resistance. The drawback it

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 2, February 2015

faces is, the size of the key often increases linearly with the number of attributes it hold. The ciphertext-size is also non constant.

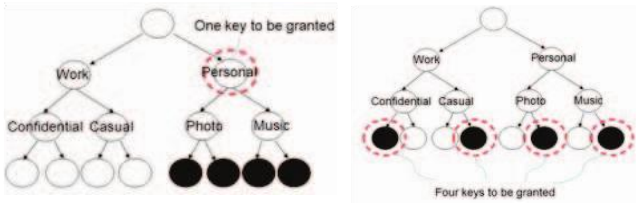
G. Multi-Authority Attribute Based Encryption Scheme

In this scheme [5], every ciphertext is associated with an attribute and secret key can be extracted by the Master-secret key holder to decrypt the ciphertext if, its associated attributes abide by the policy. In each earlier ABE schemes, the user has to go to trusted party for proving his identity before getting a secret key which allows him to decrypt messages. Thus an efficient multi-authority ABE scheme was introduced in which the user's secret key is no longer authorized by a single center authority. It is authorized separately by cooperative and independent authorities. But the problem with this scheme is, there is no focus on the compactness of secret keys. There is linear increase in the number of keys with the number of attributes it contain. In ABE scheme, attribute plays a very major role. Attributes have been exploited to obtain a public key for encryption data and used to control users' access.

H. Privacy-Preserving Public Auditing

In this paper [3], an effective TPA (Third Party Auditor) is introduced so that it would not bring any probability of attack on user data privacy, and will also not put an additional burden to user. In cloud storage, users can store their data and utilize the various resources, services or applications without causing burden of local data storage and its maintenance. However, it becomes a problem for users with limited computing resources. These users must not worry about the need to verify data integrity and use the cloud storage. This made public auditability for cloud storage very important so that users can employ a Third Party Auditor (TPA) for checking the integrity of outsourced data. High security and performance analysis show that this scheme is secure and highly efficient. The storage correctness and privacy-preserving features were given higher importance. The TPA is made secure and efficient in auditing capability to handle multiple auditing delegations. To perform auditing with minimum computation overhead and communication, care was taken to make it light weight. The use of TPA also has some drawbacks, as it is supposed to be a central, independent and reliable component; it may become bottleneck to the whole system. Any uncertain activities in TPA may cause entire cloud system to go down or reduction in its performance. Some time extra hardware or cryptographic co processor is needed when using TPA. As the data sent by the cloud data owner is in an encrypted form and the required credentials to decrypt them are kept hidden from the cloud service provider, during regulatory conformance, laws which make the data owner responsible for the protection of his data can be followed. Whenever the user is not completely happy with trusting the security or honesty of technical staff, they are motivated to encrypt their data with their own keys before dumping them to the server.

In hierarchical approaches, let us consider the Fig. 1. where the ciphertext classes are classified based on their subjects.



(a) (b)

Fig. 1. Compact key is not always possible for a fixed hierarchy

Here each node in the tree is a secret key, while leaf nodes are the keys for individual ciphertext classes. Circles with circumvented by dotted lines are keys to be granted and filled circles represent the keys for classes to be delegated. Here, the keys of descendant nodes can be derived by every key of the non leaf node.

In Fig. 1. a, if a user wants to share all his files present in the "Personal" category, he just needs to grant the key for "Personal" node , this automatically grants the keys of all the descendant nodes to the person whom he wants to share his files with. There is no problem in this case unless the classes to be shared are from the same branch.

In Fig 1. b, if user wants to share the files from different branches, he has to grant as many number of keys as the number of different branches containing these classes else the files from descendant nodes can also be accessed.

TABLE I
COMPARISONS BETWEEN THE BASIC KAC SCHEME AND OTHER RELATED SCHEMES

	Decryption key size	**Ciphertext size**	**Encryption type**
Key assignment schemes for a predefined hierarchy	Most likely non-constant (depends on the hieraarchy)	constant	symmetric or public-key
Symmetric-key encryption with Compact Key	constant	constant	symmetric-key
IBE with Compact Key	constant	non-constant	public-key
Attribute-Based Encryption	non-constant	constant	public-key
KAC	Constant	constant	public-key

On referring the Table 1 from [3] of comparison we can infer that Key Aggregate Cryptosystem (KAC) is the most convenient method in terms of the decryption key size and ciphertext size, as they are constant.

Although Symmetric-key encryption with Compact Key also has constant ciphertext size and decryption key size, its encryption type is symmetric-key type. Thus, KAC scheme has greater advantages over the other solutions.

DOI 10.17148/IJARCCE.2015.4260

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 2, February 2015

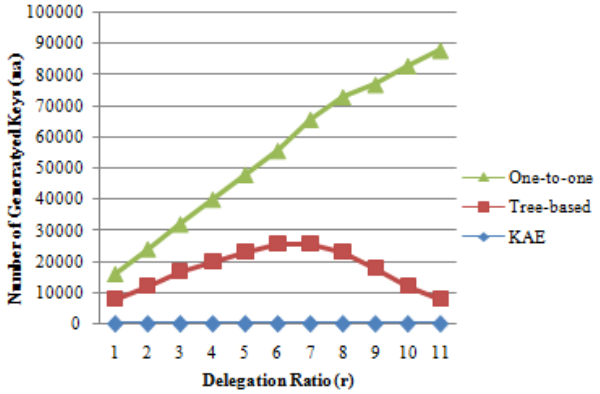


Fig. 2. Number of granted keys (n_a) required for different approaches

Looking at the performance analysis, a comparison of the number of keys granted between three methods is shown in the Fig. 2. referred from [3].

Here we can see, in one by one key granting, the number of granted keys will be same as the number of ciphertext delegate classes. With the tree based structure, the number of keys granted can be saved depending on the delegation ratio. Whereas in KAC scheme, it is efficiently implemented with the fixed size aggregate key. The constant-size aggregate key and constant-size ciphertext is the greatest advantage of this scheme. The Key Aggregate Cryptosystem (KAC) is the most efficient scheme when compared to the tree based structure and one by one granting of the keys.

Proposed Solution

Issues such as aggregation of key, constant size ciphertext, secure storage of them have remained the most important challnges. For improving the constraints of the above techniques, we propose a new scheme Key-Aggregation Cryptosystem(KAC). The KAC is an efficient and secured public-key cryptosystem for data sharing in the cloud storage. It produces constant-size ciphertexts and any number of secret keys can be aggregated.

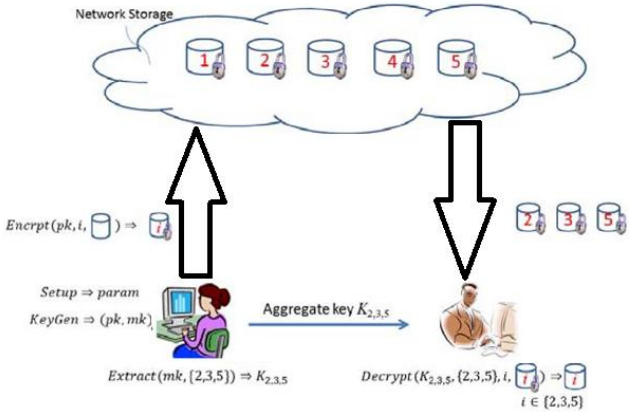


Fig. 3. KAC for data sharing in cloud storage.

In KAC, referring to Fig. 3. from [3], users encrypt the data using a public-key under an identifier of ciphertext known as class. These ciphertexts are actually categorized into separate classes. The owner of the key holds a master-secret called master-secret key, which is used to obtain secret keys for different classes. The authorized user can decrypt only those ciphertexts which he has the right to access.

The basic scheme has five polynomial-time algorithms and it is as follows:

Setup(1^l, n): This is executed by the data owner to create an account on any untrusted server. The security level parameter and the number of ciphertext classes n is taken as input. The public system parameter param is given as output.

KeyGen: The data owner executes this alogorithm for randomly generating a public/master-secret key pair (pk, msk).

Encrypt(pk, i , m): It is executed by the one who wants to encrypt the data. Public-key pk, an index i, corresponding to ciphertext class and a message m is taken as input. The ciphertext C is given as output.

Extract(msk, S): This is executed by the data owner for giving the decrypting power for certain set of ciphertext classes to the user. The master-secret key msk, and a set S of indices belonging to different classes is given as input. The aggregate key for the set S is given as output i.e. K_S.

Decrypt(K_S, S, i, C): It is executed by the delegatee who got the an aggregate key K_S, the set S, an index i associating the ciphertext class to which ciphertext C belongs to. The output obtained wil be the message m if i belongs to S.

Most importantly, the extracted key can be an aggregate key which will be as compressed as a secret key for a single class, but encompasses the decryption power for any subset of ciphertext classes. The ciphertext key size and the decryption key size both are constant.

A novel technique of aggregating the secret keys is proposed. In this schema one can aggregate as many number of secret keys and make them as compact as a single key, which has the power of all the keys aggregated in it. As data sharing is one of the prime functionality in cloud storage, the secured, efficient and flexible sharing of data is proposed. When compared to its compressing factor, it has the ability to compress the secret keys into an aggregate key which has same size as that of a single key. As it is public-key cryptosystem, it is the efficient technique which can be utilized.

CONCLUSION

In this paper, we discussed different cryptographic algorithms such as Hierarchical models, MISKD, Fuzzy Identity Based Encryption, Attribute Based Encryption, KAC. The KAC is an efficient and secured public-key cryptosystem for data sharing in the cloud storage. It produces constant-size ciphertexts in such a way that a decryption right for any set of ciphertexts is possible. Any number of secret keys can be aggregated and made as compact as a single key, containing the power of all the keys which are aggregated. The confidentiality of the encrypted files is preserved outside the set. There is no burden on the network overload, as there is utilization of compact aggregate keys. It also saves the expensive secure

DOI 10.17148/IJARCCE.2015.4260

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 2, February 2015

storage required to store these secret keys. There is no fuss of dealing with a hierarchy of delegation classes, more flexible than hierarchical approach. Regardless of the type among power set of classes, an aggregate key of constant size can be obtained. This will in turn reduce the secure storage and the overhead on the network.

REFERENCES

[1] C. K. Chu, Sherman S. M. Chow, W. G. Tzeng, J. Zhou, and R. H. Deng, "Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage", IEEE Transactions on Parallel and Distributed systems, vol. 25, no. 2, Feb 2014.

[2] C.C. Lee, P.S. Chung and M.S. Hwang, "A Survey on Attribute-based Encryption Schemes of Access Control in Cloud Environments", Int'l Journal of Network Security, Vol. 15, No. 4, PP.231-240, July 2013.

[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, Feb. 2013.

[4] M.J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, "Dynamic and Efficient Key Management for Access Hierarchies," ACM Trans. Informatio and System Security, vol. 12, no. 3, pp. 18:1-18:43, 2009.

[5] M. Chase and S.S.M. Chow, "Improving Privacy and Security in Multi-Authority Attribute-Based Encryption," Proc. ACM Conf. Computer and Comm. Security, pp. 121-130, 2009.

[6] 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.

[7] F. Guo, Y. Mu, 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.

[8] V. Goyal, O. Pandey, A. Sahai and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," Proc. 13th ACM Conf. Computer and Comm. Security (CCS '06), pp.89-98, 2006.

[9] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," Proc. 22nd Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '05), vol. 3494, pp. 457-473,2005.

[10] L. Hardesty, Secure Computers Aren't so Secure, MIT press, http://www.physorg.com/news176107396.html, 2009.

[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.

[12] 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.

[13] 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.

[14] Y. Sun and K. J. R. Liu, "Scalable Hierarchical Access Control in Secure Group Communications", proc.IEEE INFOCOM '04, 2004.

[15] B. Alomair and R. Poovendran, "Information Theoretically Secure Encryption with Almost Free Authentication", J. Universal Computer Science, vol. 15, no. 15, pp. 2937-2956, 2009.

[16] J. Benaloh, "Key Compression and Its Application to Digital Fingerprinting", technical report, Microsoft Research, 2009.