International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org
Volume 6, Issue 2, March - April 2017 ISSN 2278-6856

Key-Aggregate Cryptosystem for Data Sharing In Cloud Storage

Vasundhara Karale[1], Kiran Shelar[2], Monika Ganjewar[3], Akhtar Shaikh[4] and Prof.Ms.S.S.Pophale[5]

[1]Department of Information Technology,
Dr. V.V. Patil College of Engineering, Vilad Ghat, Ahmednagar, Maharashtra, India

[2]Department of Information Technology,
Dr. V.V. Patil College of Engineering, Vilad Ghat, Ahmednagar, Maharashtra, India

[3]Department of Information Technology,
Dr. V.V. Patil College of Engineering, Vilad Ghat, Ahmednagar, Maharashtra, India

[4]Department of Information Technology,
Dr. V.V. Patil College of Engineering, Vilad Ghat, Ahmednagar, Maharashtra, India

[5]Asstt. Prof., Department of Information Technology,
Dr. V.V. Patil College of Engineering, Vilad Ghat, Ahmednagar, Maharashtra, India

Abstract: ***Data sharing is a critical usefulness in cloud storage. In this article, we demonstrate to safely, productively, and adaptably share data with others in cloud storage. We depict new open key cryptosystems which deliver constant-measure ciphertexts with the end goal that proficient designation of decoding rights for any arrangement of ciphertexts are conceivable. The curiosity is that one can aggregate any arrangement of mystery keys and make them as minimal as a solitary key, however including the energy of all the keys being aggregated. As it were, the mystery key holder can discharge a constant-estimate aggregate key for adaptable decisions of ciphertext set in cloud storage, however the other scrambled documents outside the set remain classified. This minimal aggregate key can be advantageously sent to others or be put away in a savvy card with exceptionally restricted secure storage. We give formal security investigation of our plans in the standard model. We likewise portray other utilization of our plans. Specifically, our plans give the main open key patient-controlled encryption for adaptable pecking order, which was yet to be known.***

Keywords: cloud storage, data sharing, key aggregate encryption

1. INTRODUCTION

Data sharing is an imperative usefulness in cloud storage. For instance, bloggers can give their friends a chance to see a subset of their private pictures; an endeavor may concede her representatives access to a bit of sensitive data. The testing issue is the means by which to successfully share encrypted data. Obviously clients can download the encrypted data from the storage, decrypt them, then send them to others for sharing, however it loses the estimation of cloud storage. Clients ought to have the capacity to assign the access rights of the sharing data to others with the goal that they can access these data from the server straightforwardly. Be that as it may, finding an productive and secure approach to share fractional data in cloud storage is not minor. We will take Dropbox1 as an Example. Expect that Alice puts all her private photographs on Dropbox, and she wouldn't like to open her photographs to everybody. Because of different data spillage plausibility Alice can't feel diminished by simply depending on the security insurance instruments gave by Dropbox, so she encodes all the photographs utilizing her own particular keys before transferring. One day, Alice's companion, Bob, requests that her share the photographs assumed control over every one of these years which Bob showed up in. Alice can then utilize the share capacity of Dropbox, yet the issue now is the manner by which to assign the decryption rights for these photographs to Bob. A conceivable alternative Alice can pick is to safely send Bob the secret keys included. Normally, there are two extraordinary routes for her under the conventional encryption worldview:

Alice scrambles all documents with a solitary encryption key what's more, gives Bob the comparing secret key specifically. Alice scrambles documents with particular keys and sends Bob the comparing secret keys. Encryption keys also come with two flavors — symmetric key or asymmetric (public) key. Using symmetric encryption, when Alice wants the data to be originated from a third party, she has to give the encryptor her secret key; obviously, this is not always desirable. By contrast, the encryption key and decryption key are different in public-key encryption. The use of public-key encryption gives more flexibility for our applications. For example, in enterprise settings, every employee can upload encrypted data on the cloud storage server without the knowledge of the company's master-secret key. 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 (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 cipher-text 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 any such keys, i.e., the decryption power for any subset of ciphertext classes. With our solution, Alice can imply send Bob a single aggregate key via a secure e-mail. Bob can download the encrypted photos from Alice's Dropbox space and then use this aggregate key to decrypt these encrypted photos. The sizes of ciphertext, public-key, master-secret key and aggregate key in our KAC schemes are all of constant size.

We propose several concrete KAC schemes with different security levels and extensions in this article. All constructions can be proven secure in the standard model. To the best of our knowledge, our aggregation mechanism2 in KAC has not been investigated.

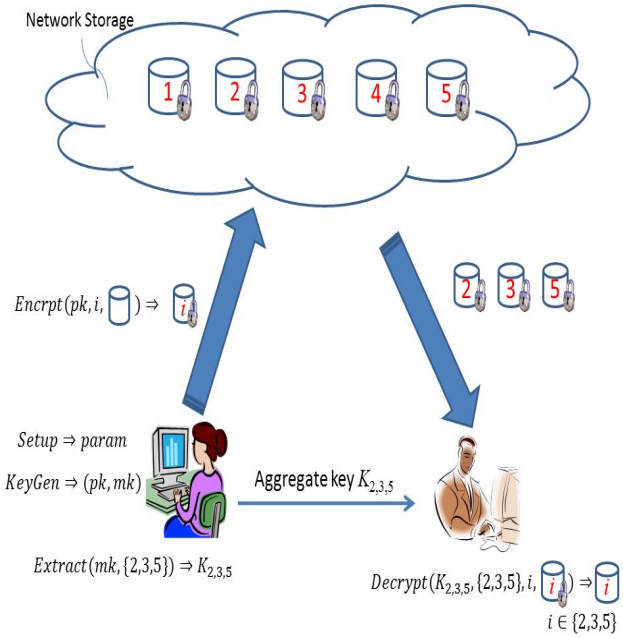

Fig. 1. Alice shares files with identifiers 2, 3, 6 and 8 with Bob by sending him a single aggregate key.

2. KEY-AGGREGATE ENCRYPTION

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. Framework

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 keypair 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 cipher-text provided that the cipher-text's class is contained in the aggregate key via Decrypt.

Setup(L, n): Executed by the data owner to setup an account on an untrusted server. On input a security level parameter L and the number of cipher-text classes n (i.e., class index should be an integer bounded by L and n), it outputs the public system parameter param, which is omitted from the input of the other algorithms for brevity.

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 cipher-text class, and a message m, it outputs a cipher-text C.

Extract (msk, S): Executed by the data owner for delegating the decrypting power for a certain set of cipher-text 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 KS.

Decrypt (KS, S, I, C): Executed by a delegate who received an aggregate key KS generated by Extract. On input KS, the set S, an index i denoting the cipher-text class the cipher-text C belongs to, and C, it outputs the decrypted result m if I belongs to S.

B. Sharing Encrypted Data

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 cipher-rtext 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 m1 , m2,……,m on the server. She first performs Setup(L , n) to get param and execute Key-Gen 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 mi by Ci = Encrypt(pk , I ,m). The encrypted data are uploaded to the server.

With param and pk, 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(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 2 S, Bob downloads Ci (and some needed values in param) from the server. With the aggregate key KS, Bob can decrypt each Ci by Decrypt(KS ,S , I , C) for each i belongs toS.

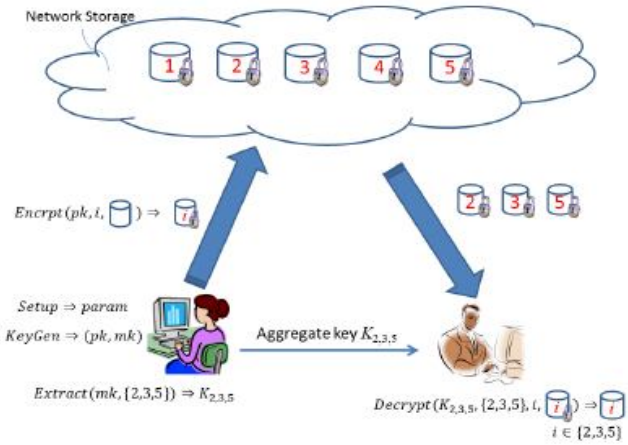

Fig. 2. Using KAC for data sharing in cloud storage

3 System Implementation

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 node 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.

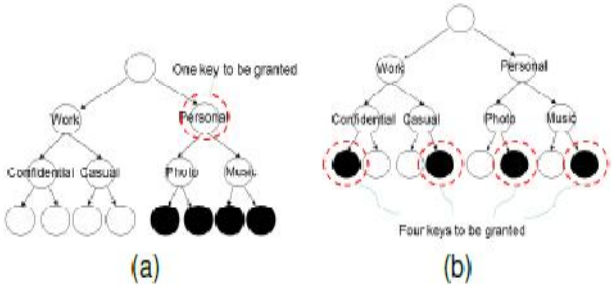

Fig. 3. Compact key is not always possible for a fixed hierarchy

In Figure 3(a), 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"). This is the ideal case, where most classes to be shared belong to the same branch and thus a parent key of them is sufficient.

However, it is still difficult for general cases. As shown in Figure 3(b), if Alice shares her demo music at work ("work"!"casual"!"demo" and "work"!"confidential" !"demo") with a colleague who also has the rights to see some of her personal data, what she can do is to give more keys, which leads to an increase in the total key size. One can see that this approach is not flexible when the classifications are more complex and she wants to share different sets of files to different people. For this delegate in our example, the number of granted secret keys becomes the same as the number of classes.

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.

4 Results

Following screen shots shows the implementation results of the proposed method.

Fig. 4. Login Form

Fig. 5. User Registration

Fig. 6. Token Insert

Fig. 7. Uploaded Files

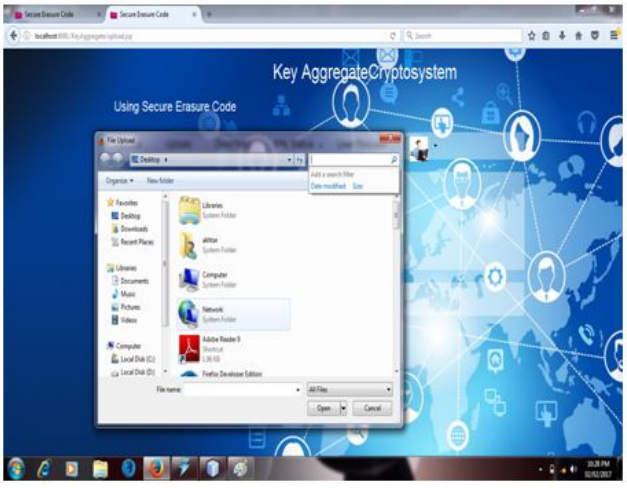

Fig.8. Browse Files

Fig.10. Downloaded Files

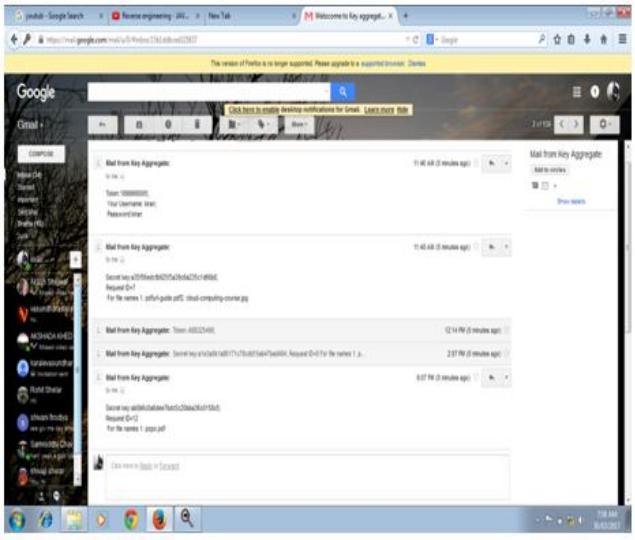

Fig.11. Mail Account

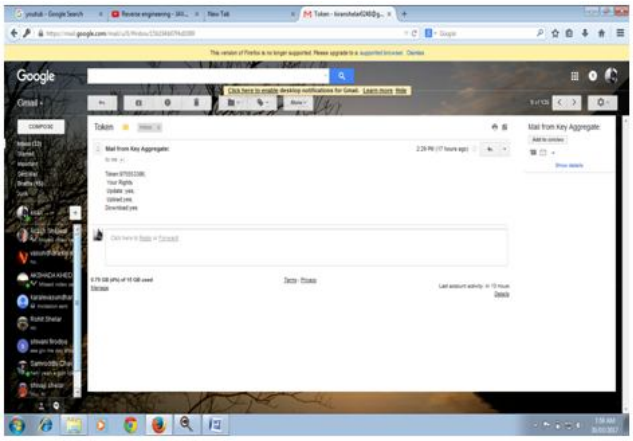

Fig.12. Aggregate Key Mail

5. CONCLUSION

Step by step instructions to secure clients' information protection is a focal question of distributed storage. With more scientific apparatuses, cryptographic plans are getting more flexible and frequently include different keys for a solitary application. In this article, we consider how to "pack" secret keys in public-key cryptosystems which bolster assignment of secret keys for various ciphertext classes in cloud capacity. Our approach is more adaptable than progressive key task which can just spare spaces in the event that every single key-holder shares a comparable arrangement of benefits. A constraint in our work is the predefined bound of the quantity of most extreme ciphertext classes. In cloud capacity, the quantity of cipher texts normally develops quickly. So we need to save enough ciphertext classes for the future augmentation. Else, we have to extend the public-key, although the parameter can be downloaded with cipher texts, it would be better if its size is free of the most extreme number of ciphertext classes. On the other hand, when one bears the appointed keys in a cell phone without utilizing unique put stock in equipment, the key is provoke to spillage, planning a spillage strong cryptosystem yet permits effective and adaptable key assignment is additionally a fascinating course.

REFERENCES

[1] Vidhate, Deepak A and Kulkarni, Parag "New Approach for Advanced Cooperative Learning Algorithms using RL Methods (ACLA)" Proceedings of the Third International Symposium on Computer Vision and the Internet, ACM, pp 12-20, 2016

[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

[3] Vidhate, Deepak A and Kulkarni, Parag "Innovative Approach Towards Cooperation Models for Multi-agent Reinforcement Learning (CMMARL)", Springer Nature series of Communications in Computer and Information Science, Vol. 628,pp. 468-478,2016

[4] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical Identity Based Encryption with Constant Size Ciphertext," in Proceedings of Advances in Cryptology - EUROCRYPT '05, ser. LNCS, vol. 3494. Springer, 2005, pp. 440–456.

[5] D. Boneh, R. Canetti, S. Halevi, and J. Katz, "Chosen-Ciphertext Security from Identity-Based Encryption," SIAM Journal on Computing, vol. 36, no. 5, pp. 1301–1328, 2007.

[6] 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,vol.12,no. 3, 2009.

[7] Vidhate, Deepak A and Kulkarni, Parag "Single Agent Learning Algorithms for Decision making in Diagnostic Applications", SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE), Vol.3, No.5,pp.46-52,2016

[8] D. Boneh and M. K. Franklin, "Identity-Based Encryption from the Weil Pairing," in Proceedings of Advances in Cryptology – CRYPTO '01, ser. LNCS, vol. 2139. Springer, 2001, pp. 213–229.

[9] M. Chase and S. S. M. Chow, "Improving Privacy and Security in Multi-Authority Attribute-Based Encryption," in ACM Conference on Computer and Communications Security, 2009, pp. 121–130.

[10] Vidhate, Deepak A and Kulkarni, Parag "Implementation of Multiagent Learning Algorithms for Improved Decision Making", International Journal of Computer Trends and Technology (IJCTT) Vol 35, No 2,pp 60-66, 2016

[11] S. M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R. H. Deng, "Dynamic Secure Cloud Storage with Provenance," in Cryptography and Security, LNCS, vol. 6805. Springer, 2012, pp. 442–464.

[12] Vidhate, Deepak, A; Kulkarni, Parag(2014): ""To improve association rule mining using new technique: Multilevel relationship algorithm towards cooperative learning", International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA), pp 241—246, 2014 IEEE