International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 02 Issue: 05 | Aug-2015 www.irjet.net p-ISSN: 2395-0072

Novel Dynamic Key Aggregate Cryptosystem for Cloud Data Sharing

Sagar Patil[1], Kirti Korabu[2]

[1] *Student, Information Technology, Sinhgad College of Engineering, Pune, Maharashtra, India*
[2] *Associate Professor, Information Technology, Sinhgad College of Engineering, Pune, Maharashtra, India*

Abstract*— *Cloud Storage Service provides facility to store and modify data easily by using network from any corner of world. Advantages of this service are good accessibility and reliability, strong protection, disaster recovery, and lowest cost. Multiple members can share that data through different virtual machines which may present on single physical machine. Cloud storage should have important functionality sharing data securely, efficiently, flexibly with others users. But unfortunately, user doesn't have physical control over the outsourced data in cloud storage environment. User's identity should not be revealed to everyone. At the same time data should not be revealed to unauthorized party including cloud storage service provider. Another challenge is that data owner should be able to share data as much they want to i.e. only selected content can be shared. Different data files are encrypted with different encryption keys and associated different decryption keys. Key aggregated cryptosystem uses the encryption keys of files which are to be shared. A single aggregate key is generated which has the same power to decrypt all those files whose keys are taken as input to generate aggregate key. This efficient and compact aggregate key is then send tended receiver securely.

Keywords— cloud data storage, datasharing, aggregate key, encryption, decryption, etc

1. Introduction

Cloud data storage allocates data storage to the user on cloud server and allows access to it using network from any corner of the world. Cloud technology is used to increase the capacity of the system dynamically without investing in new infrastructure, training new personnel, or licensing new software at local site of the user. User need not worry about all these things. In the recent years, cloud storage facility became a very popular and promising facility in IT industry. With this growth in use of cloud storage and more and more information of individuals and companies being placed in the cloud, concerns are beginning to grow about data safety and environment safety. Unauthorized access to data on cloud is not tolerable at all. Despite all the security mechanisms of the cloud service, customers are still reluctant to deploy their business in the cloud. There are some security issues in cloud computing which plays major role in slowing down its acceptance. The security of data ranked first as the greatest challenge issue of cloud computing. Key aggregate cryptosystem achieves confidentiality of data and allows multiple entities to communicate over the network such that an opponent can't understand the data being shared. There is an access control scheme that allows only valid users are allowed to access the data. This system also provides user revocation and prevents replay attacks to the system. Authentication and access control scheme is robust and more secure, unlike other access control schemes designed for clouds. Access control of data stored in cloud is distributed so that only authorized users with valid attributes can access that data. Users in existing systems rely on the server for the access control after the authentication. So if there is any unexpected privilege escalation, it will expose all data on cloud storage to unauthorized party. These things become even worse in a shared-tenancy cloud computing environment. Though the files from different users are placed on separate virtual machines (VMs), they may reside on a same physical machine. So, data in a VM could be stolen easily by instantiating another VM co-resident with the target VM on same physical machine [10]. Cloud users generally do not hold the strong belief that the cloud storage service provider is doing a good job in terms of handling data honestly. It is always desirable to use a cryptographic solution that is based on number-theoretic assumptions when the user is not perfectly trust the security of the VM or the honesty of the cloud storage service provider. These users are required to encrypt their files with their own private keys and then upload them to the cloud storage. Data sharing is the most important functionality of cloud storage service. For example, bloggers can decide which followers can view a subset of their private pictures; similarly an enterprise may grant its employees access to limited data and not to all. The challenging problem [1] is how to share only selected files. If data owner download

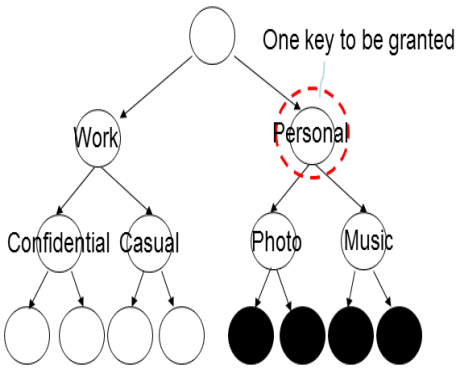
the encrypted files from the storage, decrypt these files and then send these files to other users for sharing, it loses the value of cloud storage service. Users should be able to decide and give the access rights of the shared files to other users so that they can access these files directly from the cloud storage. This system should generate a single aggregate key which will have power to decrypt all the requested files and will not be able to decrypt the remaining files on the cloud storage.

2. Security of cloud data storage

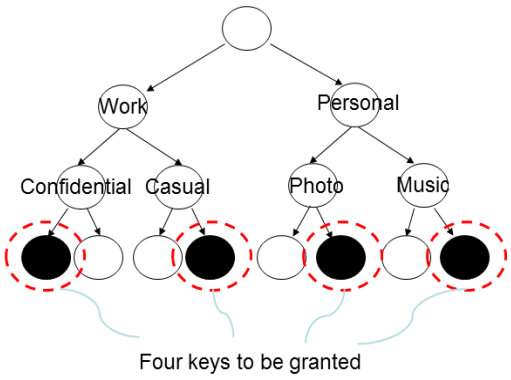
Many cloud service providers provide storage as a form of service. They take the data from the users and store them on large data centers, hence providing users a means of storage. Although these cloud service providers say that the data stored in the cloud is utmost safe but there have been cases when the data stored in these clouds have been modified or lost may be due to some security breach or some human error. Various cloud service providers adopt different technologies to safeguard the data stored in their cloud. But the question is: Whether the data stored in these clouds is secure enough against any sort of security breach? The virtualized nature of cloud storage makes the traditional mechanisms unsuitable for handling the security issues. These service providers use different encryption techniques like public key encryption and private key encryption to secure the data resting in the cloud. Another major issue that is mostly neglected is of Data-Remanence. It refers to the data left out in case of data removal. It causes minimal security threats in private cloud computing offerings, however severe security issues may emerge out in case of public cloud offerings as a result of data-remanence. Various cases of cloud security breach came into light in the last few years. Cloud based email marketing services company, Epsilon suffered the data breach, due to which a large section of its customers including JP Morgan Chase, Citibank, Barclays Bank, hotel chains such as Marriott and Hilton, and big retailers such as Best Buy and Walgreens were affected heavily and huge chunk of customer data was exposed to the hackers which includes customer email ids and bank account details. Another similar incident happened with Amazon causing the disruption of its EC2 service. The damage caused had proved to be quite costly for both the users and the system administrators. The above mentioned events depict the vulnerability of the cloud services. Another important aspect is that the known and popular domains have been used to launch malicious software or hack into the companies' secured database. It is proved that Amazon is prone to side-channel attacks, and a malicious virtual machine, occupying the same server as the target, can easily gain access to confidential data [10]. The question is: whether any such security policy should be in place for these trusted users as well? An incident relating to the data loss occurred last year with the online storage service provider "Media max" also known as "The Linkup" when due to system administration error, active customer data was deleted, leading to the data loss. SLA's with the Cloud Service providers should contain all the points that may cause data loss either due to some human or system generated error. Virtualization in general increases the security of a cloud environment. With virtualization, a single machine can be divided into many virtual machines, thus providing better data isolation and safety against denial of service attacks [10]. The VMs provide a security test-bed for execution of untested code from un-trusted users.

3. Related Work

Cryptographic key assignment schemes, et al. [9] try to reduce the expense in storing and managing large number of secret keys for general cryptographic use. With the help of tree structure, a key for a given branch can be used to derive the keys of its descendant nodes (but not the other way round). Just granting the parent key implicitly grants all the keys of its descendant nodes. In a system of tree hierarchy of symmetric keys, there are repeated evaluations of pseudorandom function/block-cipher 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, J. Benaloh et al. [7]. Most of these schemes produce keys for symmetric-key cryptosystems, even though the key derivations may require modular arithmetic as used in publickey cryptosystems, which are generally more expensive than "symmetric-key operations" such as pseudorandom function. Tree structure is taken as an example. Owner can first classify the cipher text classes according to their subjects like Figure 1.



(a)



(b)

Fig. 1. Keys for a fixed hierarchy

Each node in the tree represents a secret key, while the leaf nodes represent the keys for individual cipher text 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. In Figure 1(a), if owner 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 receiver 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 1(b), if 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 receiver in the 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. Motivated by the same problem of supporting flexible hierarchy in decryption power delegation (but in symmetric-key setting), Benaloh et al. [7] presented an encryption scheme which is originally proposed for concisely transmitting large number of keys in broadcast scenario. Its key derivation process is described here briefly for a concrete description of what are the desirable properties they want to achieve. This approach achieves similar properties and performances as in given schemes. However, it is designed for the symmetric-key setting instead. The encryptor needs to get the corresponding secret keys to encrypt data, which is not suitable for many applications. Since their method is used to generate a secret value rather than a pair of public/secret keys, it is unclear how to apply this idea for public-key encryption scheme. Finally, note that there are schemes which try to reduce the key size for achieving authentication in symmetric key encryption. However, sharing of decryption power is not a concern in these schemes. Chow et al. [8] explains a type of public-key encryption in which the public-key of a user can be set as an identity-string of the user (e.g., an email address). There is a trusted party called private key generator (PKG) in IBE which holds a master-secret key and issues a secret key to each user with respect to the user identity. The user who encrypts the files can make use of a public parameter and a user identity to encrypt a message. This cipher text can be decrypted by recipient using his secret key. In case of Identity Based Encryption, there is a constraint on key aggregation 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 cipher texts and the public parameter. Here n is the number of secret keys which can be aggregated into a constant size one. This increases the costs of storing and transmitting cipher texts greatly. This is impractical in many situations such as cloud storage service, Y Tong et. al [5].

4. System Architecture

A key-aggregate encryption scheme consists of five polynomial-time algorithms [1] as shown in Figure 2. 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 cipher text 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 cipher text classes via Extract. The generated keys can be passed to receivers 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 Decrypt4.

A. 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 cipher text classes n (i.e., class index should be an integer bounded by 1 and n), it outputs the public system parameter param, which is omitted from the input of the other algorithms for brevity.

B. KeyGen():

Executed by the data owner and randomly generates a master-secret key (msk).

C. Encrypt(pk; i;m):

Executed by data owner to encrypt data. On input msk, an index i denoting the cipher text class, and a message m, it outputs a cipher text C..

D. Extract(msk; S):

Executed by the data owner for delegating the decrypting power for a certain set of cipher text classes to the receiver. 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.

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

Executed by a receiver 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 in S.

5. Sharing Incrypted Data

A canonical application of KAC is data sharing. The key aggregation property is especially useful when one expects the delegation to be efficient and flexible. These 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 and small aggregate key. The main idea of data sharing in cloud storage using KAC is illustrated in Figure 2.

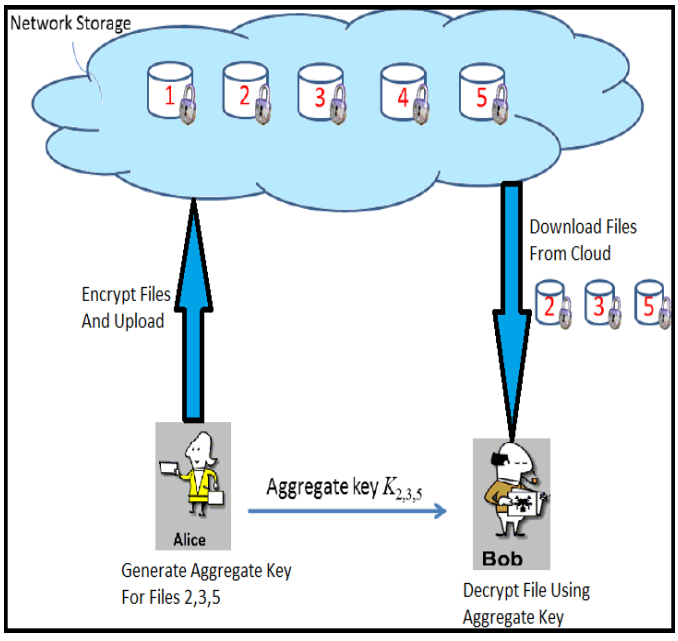


Fig. 2. File Sharing with Aggregate Key

Suppose owner wants to share her data m1;m2; ... ;mn on the server. She first performs Setup() to get param and execute KeyGen to get the master-secret key (msk). Master-secret key msk should be kept secret by owner. Anyone (including owner herself) can then encrypt each mi by Ci = Encrypt(msk; i;mi). The encrypted data are uploaded to the server. Once owner is willing to share a set S of her data with a friend receiver, she can compute the aggregate key KS for receiver by performing Extract (msk; S). Aggregate key is then sent to the receiver. After obtaining the aggregate key, receiver can download the data he is authorized to access. That is, for each i in S, receiver downloads Ci (and some needed values in param) from the server. With the aggregate key KS, Receiver can decrypt each Ci by Decrypt(KS; S; i; Ci) for each i in S.

6. Result And Analysis

A. Effect of Delegation Ratio on Number of Keys Generated

Many researchers have worked a lot in the area of cloud storage security especially in creation of aggregate key. All the methods and algorithms developed and used by them did not provide effective results. Most of the techniques are hectic for the user to handle the system. Results of conventional methods and proposed method are discussed here. A comparison of the number of granted keys according to respective delegation ratios between three methods is depicted in Figure 3.

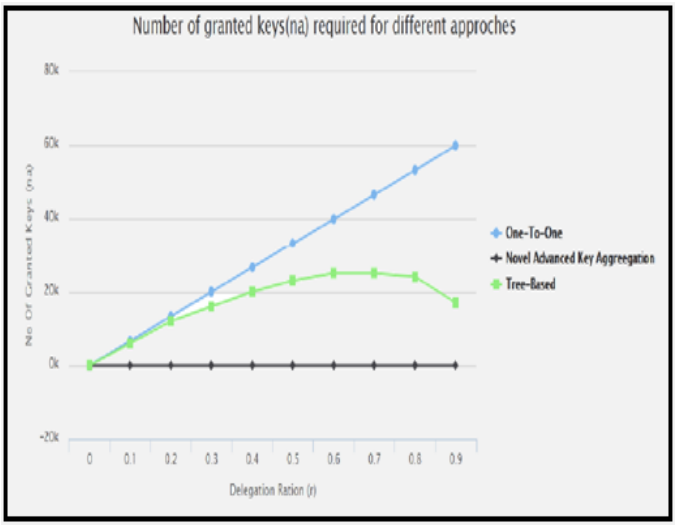


Fig. 3 Number of granted keys vs Delegation ratio

One-to-One Method is represented by blue colored line and Tree Based Method is represented by the green curve line. Proposed system is represented by black horizontal

line. Graph shows that if we grant the key one by one, the number of granted keys would be exactly equal to the number of the delegated cipher text classes. Tree-based structure saves a number of granted keys according to the delegation ratio as shown in the graph. In contradiction with this, our proposed approach, the number of aggregate keys remains constant. The delegation of decryption can be efficiently implemented with the aggregate key, which is only of fixed size. In this experiment, the delegation is randomly chosen. It covers the situation that the needs for delegating to different users may not be predictable as time goes by, even after a careful initial planning. Hierarchical key assignment does not save much in all cases. But the proposed system saves much overhead of key management.

B. Effect of File Size on Encryption Time

We can analyze the impact of file size on the encryption time for that file using graph shown in Figure 4. As the size of file increases there is increase in encryption time. So we can conclude the encryption time is proportional to the size of input file.

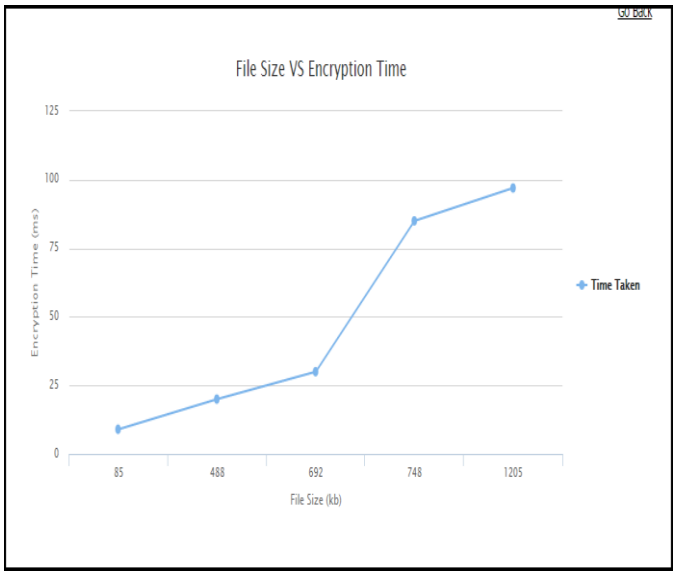


Fig. 4 Encryption Time vs File Size

C. Effect of File Size on Encryption Time for Files with Different Types of Content

We can study the Figure 6.3 to study if there is any impact of contents of files on the encryption time required for that file. We can conclude that encryption time for a file is depends not only on the size of file but also on contents of the file. Two different files with different contents may take different time for their encryption.

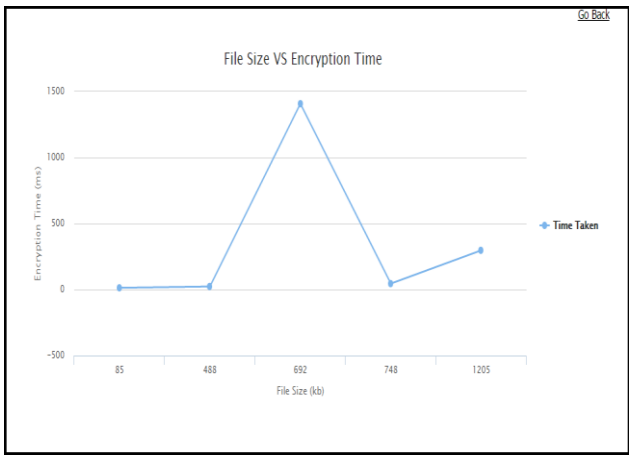


Figure 6.3 Encryption Time vs File Size for Files with Different Contents

Smaller file with complicated contents like images, takes more time for encryption than little larger file with simple contents, like text.

7. Conclusion

In this paper, we propose a solution to “reduce” secret keys in which system supports delegation of secret keys for various encrypted files stored on cloud storage as a single aggregate key. This approach is more flexible than hierarchical key assignment. No matter which one among the power set of classes, the delegatee can always get a single aggregate key having power to decrypt respective files only. This system enhances user privacy and confidentiality of knowledge of data in cloud storage.

References

[1] Cheng-Kang Chu, Sherman S. M. Chow, Wen-GueyTzeng, Jianying Zhou, and Robert H. Deng, "Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage" in Parallel and Distributed Systems, Vol. 25 , Issue: 2, IEEE Transactions, pp.468 – 477, 2014.

[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.Transactions on Information and System Security (TISSEC), vol. 12, no. 3, 2009.

[5] 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.

[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 (TISSEC), vol. 12, no. 3, 2009.

[7] 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.

[8] S. S. M. Chow, Y. Dodis, Y. Rouselakis, and B. Waters, "Practical Leakage-Resilient Identity-Based Encryption from Simple Assumptions," in ACM Conference on Computer and Communications Security, 2010, pp. 152–161.

[9] Hélder Pereira Borges, José Neuman de Souza, Bruno Schulze and Antonio Roberto Mury, "Automatic Generation of Platforms in Cloud Computing" in Network Operations and Management Symposium (NOMS), IEEE, ppi. 1311 - 1318, 2012