# Reliable Approach towards Key-Aggregate Cryptosystem for Data Sharing in Cloud Storage

AshokkumarKalal[#1], RajshriNikam[*2], Prof.SayaliKokane[#3,] Prof. Ramesh Lavhe[#4]
Affiliation
ashok.kalal@abmspcoerpune.org

***Article Info***
***Page Number: 518-523***
***Publication Issue:***
***Vol 69 No. 1 (2020)***

**Abstract**
Data sharing is an important functionality in cloud storage. In this we show how to securely, efficiently, and flexibly share data with others in cloud storage. We describe new public-key cryptosystems which produce constant-size cipher-texts such that efficient delegation of decryption rights for any set of cipher-texts are possible. The novelty is that one can aggregate any set of secret keys and make them as compact as a single key, but encompassing the power of all the keys being aggregated. In other words, the secret key holder can release a constant-size aggregate key for flexible choices of cipher-text set in cloud storage, but the other encrypted files outside the set remain confidential. This compact aggregate key can be conveniently sent to others or be stored in a smart card with very limited secure storage. In particular, our schemes give the first public-key patient-controlled encryption for flexible hierarchy, which was yet to be known.

***Article Received:*** *20 January 2020*
***Revised:*** *28 March 2020*
***Accepted:*** *10 June 2020*
***Publication:*** *07 August 2020*

**Keywords**: Key Aggregate Cryptosystem, Cloud Storage, Cipher-text, Smart Card.

## 1.0 INTRODUCTION

Data in a cloud storage environment is usually stored in the space offered by third-party companies. Instead of being provided by a single host, the storage space is integrated and distributed through centralized management. Generally speaking, the commonly seen storage protocols are NAS and SAN. Nevertheless, due to the great number of users and devices in the cloud network, the managers often cannot effectively manage the efficiency of various storage nodes. As a result, the complexity of controlling the hardware and the network traffic is increased and the performance of the cloud network is decreased.

Cloud computing services can be classified as either computing or storage. As far as data storage is concerned, although numerous schemes have been presented to improve file chunking and data compression, the waste of resources caused by revisions or changes is often overlooked. For instance, a file that is re-uploaded to the server may seriously affect the network bandwidth as well as the server workload, and also degrade efficiency. In addition, the cloud network covers a great scope and domain and the data written on storage devices by different users might be similar or identical, because of user habits and available resources. Consequently, the system manager can no longer guarantee the optimal status of each storage node in the cloud system. With the enlargement of the network, data integration bottleneck and waste of resources may occur as system processes duplicate & redundant data, despite the flexibility and rapidity of the cloud storage system.

This study uses the index name server (INS) to process cloud storage functions, including file compression, chunk matching, data de-duplication, real-time feedback control, IP information, and busy level index monitoring. Therefore, our proposed INS can manage and optimize the storage nodes according to the client-side transmission conditions so that every storage node can maintain its optimal status and provide suitable resources to clients. Moreover, to balance the load in the system, we use INS to dynamically monitor IP information and busy level index to avoid network congestion or long waiting times during transmissions. The simulation results prove that the performance enhancement of the data transmission can reach up to 20%–50%, which not only improves the performance of the cloud storage system, but also distributes the files to reduce the load on the storage nodes.

## 2.0 LITERATURE REVIEW

Encryption keys also come with two flavors, symmetric key or asymmetric (public) key. Using symmetric encryption, when client wants the data to be originated from a third party, he/she has to give the encryptor him/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. Since the decryption key should be sent via a secure channel and kept secret, small key size is always desirable. For example, we can't expect large storage for decryption keys in the resource-constraint devices like smart phones, smart cards or wireless sensor nodes. Especially, these secret keys are usually stored in the tamper-proof memory, which is relatively expensive. The present research efforts mainly focus on minimizing the communication requirements like aggregate signature [1].

### a. Cryptographic Keys for a Predefined Hierarchy –

We start by discussing the most relevant study in the literature of cryptography/security. Cryptographic key assignment schemes (e.g. [2], [3], [4], [5]) 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 (but not the other way round). Just granting the parent key implicitly grants all the keys of its descendant nodes. Sandhu [6] proposed a method to generate a tree hierarchy of symmetric-keys by using 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 modelled by an acyclic graph or a cyclic graph [8], [9], [7]. 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.

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 simultaneously.

### b. Compact Key in Symmetric-Key Encryption –

Motivated by the same problem of supporting flexible hierarchy in decryption power delegation (but in symmetric-key setting), Benaloh[10] presented an encryption scheme which is originally

proposed for concisely transmitting large number of keys in broadcast scenario [11]. 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. This approach achieves similar properties and performances as our 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.

In fuzzy IBE [12], one single compact secret key can decrypt cipher-texts encrypted under many identities which are close in a certain metric space, but not for an arbitrary set of identities and therefore it does not match with our idea of key aggregation.

**c. Other Encryption Schemes –**

Attribute-based encryption (ABE) [13], [14] allows each cipher-text 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 cipher-text 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 cipher-text-size is not constant (e.g., [15])

To delegate the decryption power of some cipher-texts without sending the secret key to the delegate, a useful primitive is proxy re-encryption (PRE) (e.g., [16], [17], [18], [19]). PRE is well known to have numerous applications including cryptographic file system [20]. The transformation key of proxy should be well-protected. Using PRE just moves the secure key storage requirement from the delegate to the proxy. It is thus undesirable to let the proxy reside in the storage server. That will also be inconvenient since every decryption requires separate interaction with the proxy.

**Summarization –**

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-text expansion, by distributing to each authorized user a single and small aggregate key.

## 3.0 PROBLEM DEFINITION AND SCOPE

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 (e.g. encryption, authentication) multiple times. In this paper, we study how to make a decryption key more powerful in the sense that it allows decryption of multiple cipher-texts, 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 cipher-texts (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 cipher-texts 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 cipher-text classes.

## 4.0 PROPOSED SYSTEM

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 (e.g. encryption, authentication) multiple times. 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 cipher-texts 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 cipher-text classes.

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 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.

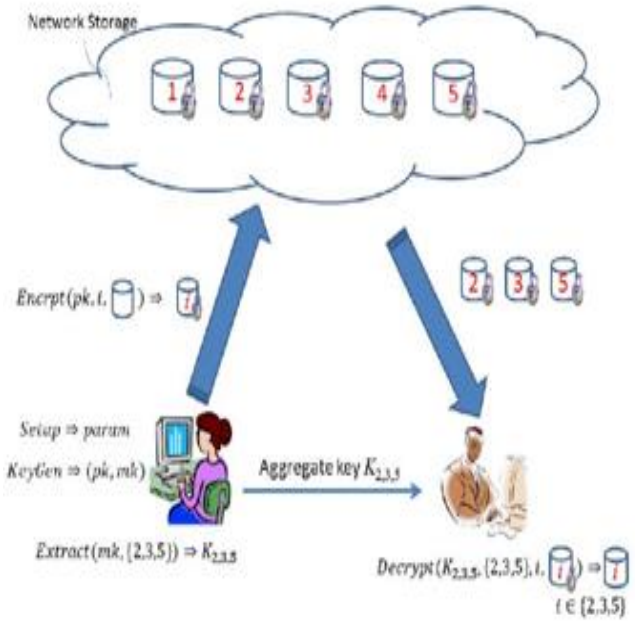


**Fig. 1: System Architecture**

**Table 1- Comparisons between our basic KAC scheme and other related schemes**

| Schemes | Decryption key size | Cipher-text size | Encryption type |
|---|---|---|---|
| Key assignment schemes for a predefined hierarchy (e.g., [7]) | Most likely non-constant (Depends on hierarchy) | Constant | Symmetric-key or public-key |
| Symmetric-key encryption with Compact Key (e.g., [8]) | Constant | Constant | Symmetric-key |
| IBE with Compact Key (e.g., [9]) | Constant | Non-constant | Public-key |
| Attribute-Based Encryption (e.g., [10]) | Non-constant | Constant | Public-key |
| KAC | Constant | Constant | Public-key |

## 5.0 CONCLUSION

In this way we consider how to "compress" secret keys in public-key cryptosystems which support delegation of secret keys for different cipher-text classes in cloud storage. No matter which one among the power set of classes, the delegate can always get an aggregate key of constant size. Our approach is more flexible than hierarchical key assignment which can only save spaces if all key-holders share a similar set of privileges. Although the parameter can be downloaded with cipher-texts, it would be better if its size is independent of the maximum number of cipher-text classes. On the other hand, when one carries the delegated keys around in a mobile device without using special trusted hardware, the key is prompt to leakage, designing a leakage-resilient cryptosystem, yet allows efficient and flexible key delegation is also an interesting direction.

## ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Er. AtulMarathe (Adjunct Professor), Er. Ashok Ranade (Innovation Club Member) for their invaluable guidance and support throughout the research process. We also wish to thank Dr. Sunil B. Thakare (Principal, APCOER, Pune) for their support. Finally, we are grateful to all of the research participants who generously gave their time and effort to this project.

## REFERENCES

[1] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," in Proceedings of Advances in Cryptology - EUROCRYPT '03, ser. LNCS, vol. 2656. Springer, 2003, pp. 416–432.

[2] S. G. Akl and P. D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," ACM Transactions on Computer Systems (TOCS), vol. 1, no. 3, pp. 239–248, 1983.

[3] Dhablia, D., & Timande, S. (n.d.). Ensuring Data Integrity and Security in Cloud Storage.

[4] Verma, M. K., & Dhabliya, M. D. (2015). Design of Hand Motion Assist Robot for Rehabilitation Physiotherapy. International Journal of New Practices in Management and Engineering, 4(04), 07–11.

[5] Mr. Dharmesh Dhabliya, M. A. P. (2019). Threats, Solution and Benefits of Secure Shell. International Journal of Control and Automation, 12(6s), 30–35.

[6] Raju K, Dannis G and and Robert T, "Recent G. C. Chick and S. E. Tavares, "Flexible Access Control with Master Keys," in Proceedings of Advances in Cryptology - CRYPTO '89, ser. LNCS, vol. 435. Springer, 1989, pp. 316–322.

[7] W.-G. Tzeng, "A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy," IEEE Transactions on Knowledge and Data Engineering (TKDE), vol. 14, no. 1, pp. 182–188, 2002.
[8] 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.
[9] R. S. Sandhu, "Cryptographic Implementation of a Tree Hierarchy for Access Control," Information Processing Letters, vol. 27, no. 2, pp. 95–98, 1988.
[10] 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.
[11] Y. Sun and K. J. R. Liu, "Scalable Hierarchical Access Control in Secure Group Communications," in Proceedings of the 23th IEEE International Conference on Computer Communications (INFOCOM '04). IEEE, 2004.
[12] 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.
[13] J. Benaloh, "Key Compression and Its Application to Digital Fingerprinting," Microsoft Research, Tech. Rep., 2009.
[14] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," in Proceedings of Advances in Cryptology - EUROCRYPT '05, ser. LNCS, vol. 3494. Springer, 2005, pp. 457–473.
[15] 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 Com-munications Security (CCS '06). ACM, 2006, pp. 89–98.
[16] 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.
[17] T. Okamoto and K. Takashima, "Achieving Short Ciphertexts or Short Secret-Keys for Adaptively Secure General Inner-Product Encryption," in Cryptology and Network Security (CANS '11), 2011.
[18] R. Canetti and S. Hohenberger, "Chosen-Ciphertext Secure Proxy Re-Encryption," in Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07). ACM, 2007.
[19] C.-K. Chu and W.-G. Tzeng, "Identity-Based Proxy Re-encryption Without Random Oracles," in Information Security Conference (ISC '07), ser. LNCS, vol. 4779. Springer, 2007, pp. 189–202.
[20] C.-K. Chu, J. Weng, S. S. M. Chow, J. Zhou, and R. H. Deng, "Conditional Proxy Broadcast Re-Encryption," in Australasian Conference on Information Security and Privacy (ACISP '09), ser. LNCS, vol. 5594. Springer, 2009, pp. 327–342.
[21] S. S. M. Chow, J. Weng, Y. Yang, and R. H. Deng, "Efficient Unidirectional Proxy Re-Encryption," in Progress in Cryptology - AFRICACRYPT 2010, ser. LNCS, vol. 6055. Springer, 2010, pp. 316–332.