

## Scalable Data Sharing In Cloud Storage Using Improved Aggregate Key Crypto System

**Ravilla Srinivasulu**

PG Scholar,

Department of CSE,

Sri Chundi Ranganayakulu Engineering College,  
Chilakaluripet, Guntur, AP, India.

**G.Mallikharjuna Rao**

Assistant Professor,

Department of CSE,

Sri Chundi Ranganayakulu Engineering College,  
Chilakaluripet, Guntur, AP, India.

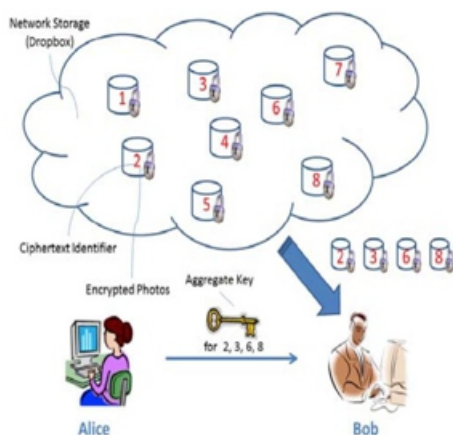
### ABSTRACT:

Data sharing is an important functionality in cloud storage. In this paper, we show how to securely, efficiently, and flexibly share data with others in cloud storage. We describe new public-key cryptosystems that produce constant-size ciphertexts such that efficient delegation of decryption rights for any set of ciphertexts 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 ciphertext 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. We provide formal security analysis of our schemes in the standard model. We also describe other application of our schemes. In particular, our schemes give the first public-key patient-controlled encryption for flexible hierarchy, which was yet to be known.

### INTRODUCTION:

CLOUD storage is gaining popularity recently. In enterprises settings, we see the rise in demand for data outsourcing, which assists in the strategic management of corporate data. It is also used as a core technology behind many online services for personal applications. Nowadays, it is easy to apply for free accounts for email, photo album, file sharing and/or remote access, with storage size more than 25 GB (or a few dollars for more than 1 TB). Together with the current wireless technology, users can access almost all of their files and emails by a mobile phone in any corner of the world. Considering data privacy, a traditional way to ensure it is to rely on the server to enforce the access control after authentication (e.g., [1]), which means any unexpected privilege escalation will expose all data.

In a shared-tenancy cloud computing environment, things become even worse. Data from different clients can be hosted on separate virtual machines (VMs) but reside on a single physical machine. Data in a target VM could be stolen by instantiating another VM coresident with the target one [2]. Regarding availability of files, there are a series of cryptographic schemes which go as far as allowing a third-party auditor to check the availability of files on behalf of the data owner without leaking anything about the data [3], or without compromising the data owners anonymity [4]. Likewise, cloud users probably will not hold the strong belief that the cloud server is doing a good job in terms of confidentiality. A cryptographic solution, for example, [5], with proven security relied on number-theoretic assumptions is more desirable, whenever the user is not perfectly happy with trusting the security of the VM or the honesty of the technical staff. These users are motivated to encrypt their data with their own keys before uploading them to the server. Data sharing is an important functionality in cloud storage. For example, bloggers can let their friends view a subset of their private pictures; an enterprise may grant her employees access to a portion of sensitive data. The challenging problem is how to effectively share encrypted data. Of course users can download the encrypted data from the storage, decrypt them, then send them to others for sharing, but it loses the value of cloud storage. Users should be able to delegate the access rights of the sharing data to others so that they can access these data from the server directly. However, finding an efficient and secure way to share partial data in cloud storage is not trivial. Below we will take Dropbox 1 as an example for illustration. Our Contributions 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 ciphertexts, without increasing its size. Specifically, our problem statement is



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

ciphertext 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 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 ciphertext classes. With our solution, Alice can simply 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 scenario is depicted in Fig. 1.

The sizes of ciphertext, public-key, master-secret key, and aggregate key in our KAC schemes are all of constant size. The public system parameter has size linear in the number of ciphertext classes, but only a small part of it is needed each time and it can be fetched on demand from large (but nonconfidential) cloud storage. Previous results may achieve a similar property featuring a constant-size decryption key, but the classes need to conform to some predefined hierarchical relationship.

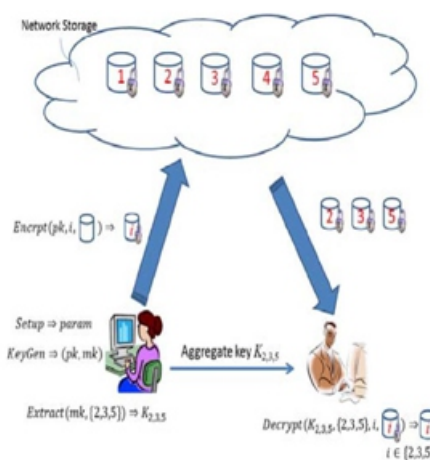
Our work is flexible in the sense that this constraint is eliminated, that is, no special relation is required between the classes. The detail and other related works can be found in Section 3. We propose several concrete KAC schemes with different security levels and extensions in this paper. All constructions can be proven secure in the standard model. To the best of our knowledge, our aggregation mechanism in KAC has not been investigated.

## 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.

### Framework:

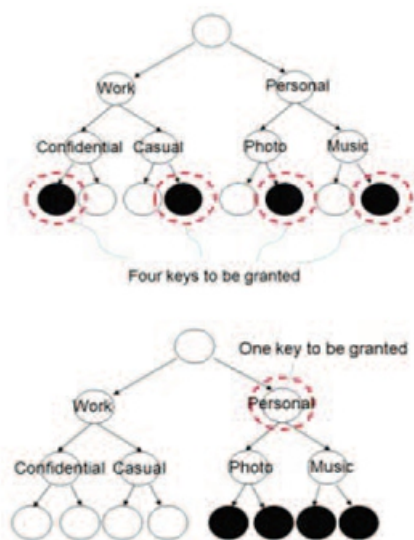
A key-aggregate encryption scheme consists of five polynomial-time algorithms as follows. The data owner establishes the public system parameter via Setup and generates a public/master-secret key pair via KeyGen. Messages can be encrypted via Encrypt by anyone who also decides what ciphertext class is 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 ciphertext provided that the ciphertext's class is contained in the aggregate key via Decrypt.



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

We take the tree structure as an example. Alice can first classify the ciphertext classes according to their subjects like Fig. 3. Each node in the tree represents a secret key, while the leaf nodes represent the keys for individual ciphertext classes. Filled circles represent the keys for the classes to be delegated and circles circumscribed 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 Fig. 3a, 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 Fig. 3b, 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 delegatee in our example, the



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

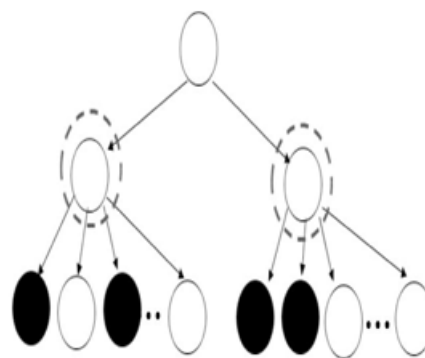
Compact Key in Identity-Based Encryption (IBE) IBE (e.g., [20], [21], [22]) is a type of public-key encryption in which the public-key of a user can be set as an identity string of the user (e.g., an email address). There is a trusted party called private key generator in IBE which holds a master-secret key and issues a secret key to each user with respect to the user identity. The encryptor can take the public parameter and a user identity to encrypt a message.

The recipient can decrypt this ciphertext by his secret key. Guo et al. [23], [9] tried to build IBE with key aggregation. One of their schemes [23] assumes random oracles but another [9] does not. In their schemes, key aggregation is constrained 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 [23], [9] comes at the expense of  $O(n)$  sizes for both ciphertexts and the public parameter, where  $n$  is the number of secret keys which can be aggregated into a constant size one. This greatly increases the costs of storing and transmitting ciphertexts, which is impractical in many situations such as shared cloud storage. As we mentioned, our schemes feature constant ciphertext size, and their security holds in the standard model. In fuzzy IBE [21], one single compact secret key can decrypt ciphertexts 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.

### 3.4 OTHER ENCRYPTION SCHEMES:

Attribute-based encryption (ABE) [10], [24] allows each ciphertext 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 ciphertext can be decrypted by this key if its associated attribute conforms to the policy. For example, with the secret key for the policy  $\delta_{2,3,6,8}$ , one can decrypt ciphertext tagged with class 2, 3, 6, or 8. However, 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 ciphertext-size is not constant (e.g., [25]).



**Fig. 4. Key assignment in our approach.**

### PERFORMANCE:

For encryption, the value  $e_{g,1}$  can be precomputed and put in the system parameter. On the other hand, we can see that decryption only takes two pairings while only one of them involves the aggregate key. That means we only need one pairing computation within the security chip storing the (secret) aggregate key.

It is fast to compute a pairing nowadays, even in resource-constrained devices. Efficient software implementations exist even for sensor nodes. Discussions on the “magic” of getting constant-size aggregate key and constant-size ciphertext simultaneously comes from the linear-size system parameter.

Our motivation is to reduce the secure storage and this is a tradeoff between two kinds of storage. The parameter can be placed in nonconfidential local storage or in a cache provided by the service company. They can also be fetched on demand, as not all of them are required in all occasions. The system parameter can also be generated by a trusted-party, shared between all users and even hard-coded to the user program (and can be updated via “patches”).

In this case, while the users need to trust the parameter-generator for securely erasing any ephemeral values used, the access control is still ensured by a cryptographic mean instead of relying on some server to restrict the accesses honestly.

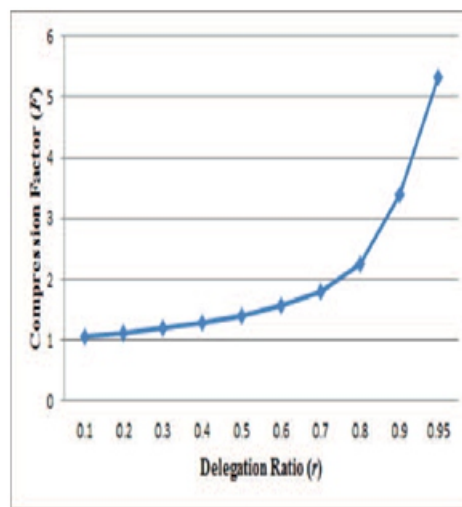
## PERFORMANCE ANALYSIS

### 5.1 Compression Factors:

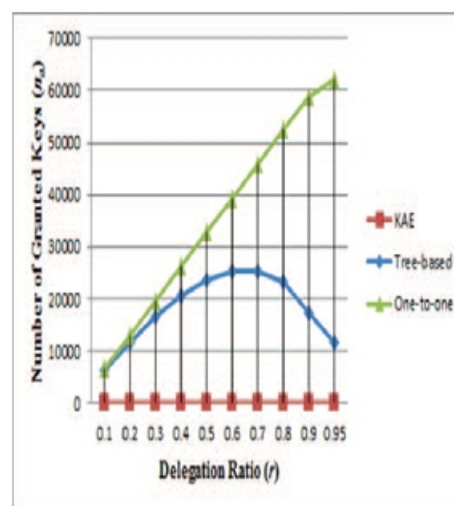
For a concrete comparison, we investigate the space requirements of the tree-based key assignment approach described in Section 3.1. This is used in the complete-subtree scheme, which is a representative solution to the broadcast encryption problem following the well-known-subset-cover framework [33].

It employs a static logical key hierarchy, which is materialized with a full binary key tree of height  $h$  (equals to 3 in Fig. 3), and thus can support up to  $2^h$  ciphertext classes, a selected part of which is intended for an authorized delegatee. In an ideal case as depicted in Fig. 3a, the delegatee can be granted the access to  $2^{h_s}$  classes with the possession of only one key, where  $h_s$  is the height of a certain subtree (e.g.,  $h_s = \frac{1}{4} \cdot 2$  in Fig. 3a).

On the other hand, to decrypt ciphertexts of a set of classes, sometimes the delegatee may have to hold a large number of keys, as depicted in Fig. 3b. Therefore, we are interested in  $n_a$ , the number of symmetric keys to be assigned in this hierarchical key approach, in an average sense.



(a)



(b)

Fig. 5. (a) Compression achieved by the tree-based approach for delegating different ratio of the classes. (b) Number of granted keys ( $n_a$ ) required for different approaches in the case of 65,536 classes of data

## CONCLUSION AND FUTURE WORK:

How to protect users’ data privacy is a central question of cloud storage. With more mathematical tools, cryptographic schemes are getting more versatile and often involve multiple keys for a single application. In this paper, we consider how to “compress” secret keys in public-key cryptosystems which support delegation of secret keys for different ciphertext classes in cloud storage. No matter which one among the power set of classes, the delegatee 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. A limitation in our work is the predefined bound of the number of maximum ciphertext classes. In cloud storage, the number of ciphertexts usually grows rapidly. So we have to reserve enough ciphertext classes for the future extension. Otherwise, we need to expand the public-key as we described in Section 4.2. Although the parameter can be downloaded with ciphertexts, it would be better if its size is independent of the maximum number of ciphertext 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 [22], [34] yet allows efficient and flexible key delegation is also an interesting direction.

## REFERENCES:

- [1] S.S.M. Chow, Y.J. He, L.C.K. Hui, and S.-M. Yiu, "SPICE – Simple Privacy-Preserving Identity-Management for Cloud Environment," Proc. 10th Int'l Conf. Applied Cryptography and Network Security (ACNS), vol. 7341, pp. 526-543, 2012.
- [2] L. Hardesty, Secure Computers Aren't so Secure. MIT press, <http://www.physorg.com/news176107396.html>, 2009.
- [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] B. Wang, S.S.M. Chow, M. Li, and H. Li, "Storing Shared Data on the Cloud via Security-Mediator," Proc. IEEE 33rd Int'l Conf. Distributed Computing Systems (ICDCS), 2013.
- [5] 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.
- [6] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," Proc. 22nd Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '03), pp. 416-432, 2003.
- [7] M.J. Atallah, M. Blanton, N. Fazio, and K.B. Frikken, "Dynamic and Efficient Key Management for Access Hierarchies," ACM Trans. Information and System Security, vol. 12, no. 3, pp. 18:1-18:43, 2009.
- [8] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records," Proc. ACM Workshop Cloud Computing Security (CCSW '09), pp. 103-114, 2009.
- [9] 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.
- [10] 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.
- [11] S.G. Akl and P.D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," ACM Trans. Computer Systems, vol. 1, no. 3, pp. 239-248, 1983.
- [12] G.C. Chick and S.E. Tavares, "Flexible Access Control with Master Keys," Proc. Advances in Cryptology (CRYPTO '89), vol. 435, pp. 316-322, 1989.