Key-Aggregate Cryptosystem based on Elliptic Curve Cryptography for Data Sharing in Cloud Storage with Result and Analysis

Kulkarni Mayuri A.
PG Student,
M.B.E.S. College of Engg
Ambajogai

V. R. Chirchi
Professor
PG Dept.,
M.B.E.S. College of Engg
Ambajogai

ABSTRACT
It is important to share data securely, efficiently and flexibly in cloud storage. We describe public-key encryption technique based on elliptic-curve theory which is used to create faster, smaller and more efficient cryptographic keys. This public-key cryptosystem produces constant size cipher texts and user can aggregate any set of secret keys and make them as compact as single and can decrypt any set of cipher texts by using that compact aggregate key but, files outside the set remain confidential. In this cryptosystem it is possible to efficiently assign decryption rights for the set of cipher texts to any users. The secret key holder can release a constant-size aggregate key for set of cipher texts and this compact aggregate key conveniently shared with others with very limited secure storage. In this paper, we study how to create a decryption key more powerful so that it can allows decryption of multiple cipher texts, without increasing key size.

Keywords
Key-aggregate cryptosystem, Elliptic curve cryptography, public-key cryptosystem, Data sharing.

1. INTRODUCTION
Cloud storage is becoming very popular recently. Online data almost always present in shared environments so that, ensuring privacy is a very important task. Nowadays, many online services are available that are using personal data. Any user can easily apply for free accounts for email, photos, files etc. with specified storage space. Also with the help of wireless technology users are accessing their files, emails by their mobiles from anywhere. Traditionally, data privacy was provided by depending on server ensuring access control mechanism and authentication but, in this any unexpected privilege escalation will expose all data. Solution for this is to encrypt data before uploading to cloud storage.

In this paper, to encrypt data effectively we propose a basic key-aggregate scheme based on elliptic curves which is public-key cryptosystem that assign decryption rights to multiple cipher text classes using a single constant sized key. Any user can combine set of secret keys and make them as a single small key which hold the same ability of all the keys and this key is called as aggregate key. By using this aggregate key user can decrypt a specified set of cipher texts. Consider example of Dropbox. Suppose, Alice puts all photos on Dropbox and she does not want to share her data with everyone. For more privacy she encrypts all photos before uploading to the server. Bob asks her to share some photos in which he appears. Now there are two possibilities:

1. Alice can encrypt all files with a single encryption key and share that key directly with Bob.
2. Alice can encrypt files with distinct keys and share all corresponding keys with Bob.

First approach is improper since, unwanted data also gets exposed to Bob. In second approach, numbers of keys are as many as number of shared files which may be hundreds or thousands in numbers so transferring these keys requires secure channel and large storage space which is expensive. Therefore, best solution to above problem is Alice has to encrypt data with distinct keys but share a single decryption key of constant size to Bob.

To make Key-Aggregate cryptography (KAC) more efficient we are using Elliptic-Curve cryptography (ECC) with it. ECC is a public key encryption technique that can be used to create faster, smaller and more efficient cryptographic keys. ECC provides more security for a smaller key size therefore; it also reduces the processing overhead. That means smaller parameters can be used in ECC as compare to other systems such as RSA and DSA, but with equivalent level of security.

2. RELATED WORK

2.1 Cryptography Using Predefined Hierarchy of Secret Key
Most efficient techniques for access control of online data is use of pre-defined hierarchy of secret keys [2] i.e. in the form of tree structure. In tree structure key assigned to a particular node is used to derive the keys of its descendent nodes i.e. granting the access to key corresponding to any node implicitly grants access to all keys to its descendent nodes. This technique minimizes the expense in storing and managing secret keys.

Sandhu [3] proposed a method to generate a tree hierarchy to define access control using symmetric keys. With this scheme information is classified into classes and these classes are organized as a rooted tree i.e. hierarchy. In this tree most privileged security class is at the root. User stores a single key of fixed size associated to its security class and keys for the security classes in the subtree are generated from this key by using one-way functions.

Generally, hierarchical approach is efficient only if user wants to share all files under particular branch of hierarchy. But, as the number of branches increases the number of keys increases.

2.2 Symmetric-Key Encryption using Compact Key
This encryption scheme [4] is invented for sharing number of keys at a time in broadcast scenario. Encryptor needs to get the respective secret key for encryption of data trough secure

channel. This key sharing via secure channel is costly and not always suitable for many applications on cloud.

2.3 Identity-Based Encryption with Compact Key

Identity-Based Encryption(IBE) [5] is based on public-key cryptosystem in which the public key of a user is some unique information about the identity of the user (e.g. a user's email address). In public-key cryptosystem key pair i.e. public key and private key used for encryption-decryption process. This means that a sender who has access to the public parameters of the system can encrypt a message using e.g. the text-value of the receiver's name or email address as a key. The receiver obtains its decryption key from a central authority, which needs to be trusted as it generates secret keys for every user. In IBE relies on trusted party called as private key generator(PKG) which has a master-secret key and provides each user a secret key depends upon identity of user. This increases the costs of storing and transmitting ciphertexts which is not suitable in many situations as shared cloud storage.

Fuzzy IBE [6] allows to one single compact secret key to decrypt cipher texts those are encrypted under many identities which are near about close in specific manner. Fuzzy IBE allows private key as identity and to decrypt cipher text encrypted with identity only if those identities are close to each other. Fuzzy IBE can also enable encryption using biometric inputs as identities.

2.4 Attribute Based Encryption

In Attribute Based Encryption [7] and [8] each user is identified by a set of attributes. Each cipher text in ABE is associated with particular set of attributes and it can be only decrypted by user who has access to corresponding secret key.

The master secret key holder can extract a secret key and securely shares with user who satisfies the access control policies defined by data owner depends upon attributes of users. A drawback of this scheme is that the size of key increases with the number of attributes and the cipher text size is not constant.

2.5 Proxy Re-Encryption

In Proxy Re-Encryption (PRE) [9] proxy is allowed to convert a cipher text with one key into cipher text of same message but with different key i.e. this scheme grants to proxy server the ability to convert the cipher text encrypted by one user's public key into cipher text with another user's public key. In this scheme proxy doesn't have knowledge of data that has been sent. Any user has to trust the proxy that it only converts cipher texts according to given instructions.

Proxy Re-Encryption scheme transfers the rights of secure key storage from the delegate to the proxy. This is inconvenient because every user needs to interact with the proxy for decryption.

3. PROPOSED SYSTEM

We describe the more efficient Key-Aggregate cryptosystem (KAC) that combines different decryption keys for the files of same class into a single key for that subset of files. We create a more powerful decryption key so that it can decrypt number of cipher texts in the same class without increasing key size.

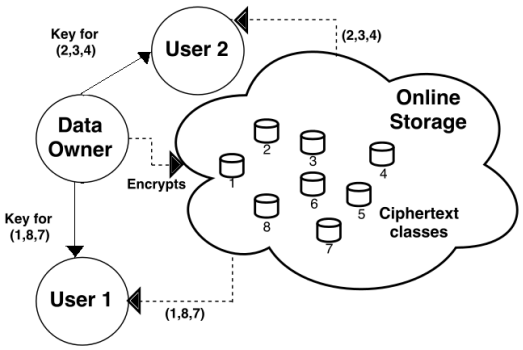


Fig 1: Key-Aggregate Cryptosystem for Online Storage

"We design Key-Aggregate Encryption scheme based on Public Key Encryption which is efficient and flexible in the sense that any subset of cipher texts is decryptable by decryption key of constant size which also called as aggregate key."

Data owner encrypts all files with the public key and also with the identifier for that cipher text. Data owner also contains master secret key from which different secret keys for cipher text classes can be extracted. This extracted key is called as aggregate key which contains power of keys to decrypt number of cipher texts under same class.

With this solution, data owner encrypts all files under the identifier for the cipher text and sends a single aggregate key to the receiver via secure channel. Receiver then downloads the specific files from cloud storage and uses this aggregate key to decrypt these files.

Fig 1 shows Key-Aggregate Cryptosystem, here data owner encrypt files by using its private key and stores encrypted files on cloud storage. Then data owner extract aggregate keys from master secret key and shares aggregate key for encrypted files with identifiers 1,8 & 7 with user1 and shares aggregate key for encrypted files with identifiers 2,3,& 4 with user2, so that user1 and user2 can decrypt those three files by using single aggregate key.

3.1 Key Aggregate Cryptosystem

A Key-Aggregate Cryptosystem consists of the following algorithms:

1. Setup(1^{λ},n): It takes input the number of cipher text classes 'n' and group order parameter 'λ' and gives public and private parameters as a output.
The data owner executes the setup phase.

2. KeyGen (): Outputs the public key 'PK' and master secret key 'msk' pair.

This phase also gets executed by data owner.

3. Encrypt(PK,i,m): Takes input public key 'PK', cipher text class 'i' and the message 'm' and gives output the cipher text 'C'.

This phase is executed by any user who wants to store the encrypted data on cloud storage.

4. Extract(msk,S): Takes input the master secret key 'msk' and a subset $S \subseteq \{1,2,\ldots n\}$ and computes the aggregate key 'K_s' for the given subset of cipher text classes.

This phase is executed by data owner for providing decryption rights for a particular set of ciphertexts classes to particular user.

5. Decrypt(K_s,S,i,C={c1,c2,c3,…}): Takes input as the aggregate key 'K_s' corresponding to a subset S⊆ {1,2,..n}, the cipher text class 'i' and the set of cipher texts 'C' and gives output as a decrypted message 'm' .

This phase is executed by user who got aggregate key and decryption authority.

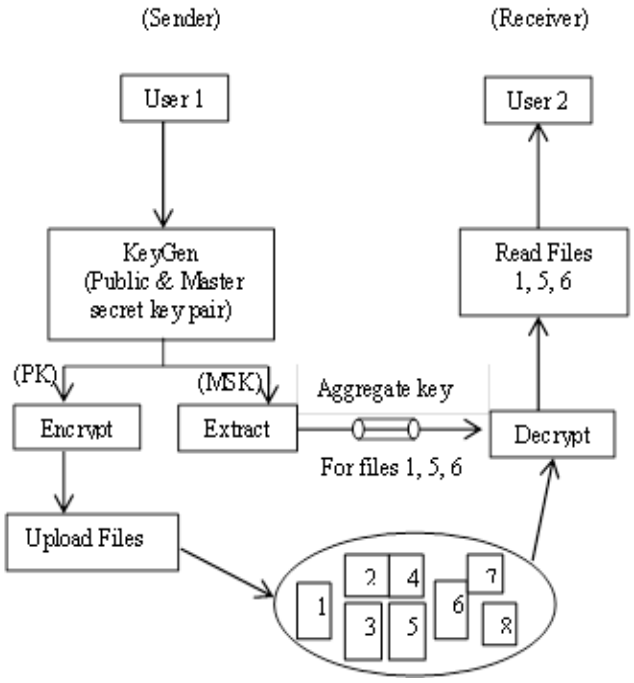


Fig 2: Key-Aggregate Cryptosystem Data Flow Diagram

3.2 Elliptic Curve Cryptography

In 1985, Neal Koblitz and Victor Miller independently proposed using elliptic curves to design public key cryptographic systems. In the late 1990`s, ECC was standardized by a number of organizations and it started receiving commercial acceptance. Nowadays, it is mainly used in the resource constrained environments, such as ad-hoc wireless networks and mobile networks.

Elliptic curves are used to construct the public key cryptography system. The private key *d* is randomly selected from [1, *n*-1], where *n* is integer. Then the public key *Q* is computed by *dP*, where *P,Q* are points on the elliptic curve. Like the conventional cryptosystems, once the key pair (*d*, *Q*) is generated, a variety of cryptosystems such as signature, encryption/decryption, key management system can be set up.

ECC requires significantly smaller key size with same level of security. Benefits of having smaller key sizes are faster computations need less storage space. ECC ideal for constrained environments such as Pagers, PDAs, Cellular Phones, Smart Cards.

Table 1: NIST recommended key sizes

Symmetric algorithm (bit)	RSA and DH (bit)	ECC (bit)
56	512	112
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

4. PERFORMANCE ANALYSIS

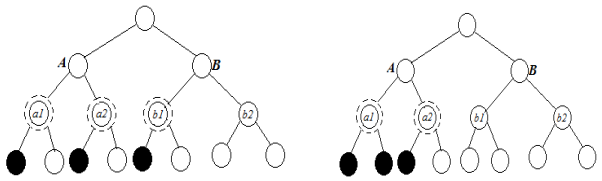
For this approach to evaluate performance we consider tree structure in fig:3 as an example. Here height of tree is 4 (h=4).

For this tree total number of classes i.e. N=31 ($N = 2^{h+1} - 1$). Consider now Alice has demanded some set of classes i.e. n = 3 so for this Delegation Ratio i.e. r = 3/31 = 0.1 (r = n/N).

To provide access on 3 classes how many keys need to be distributed to Alice? N_a i.e. Total number of keys distributed for given r. ($N_a = 2.5$)

Compression factor F for certain h is average number of delegated classes that each granted key can decrypt i.e. ratio of total number of classes to the total number of keys distributed for given r ($F = n/N_a$). F = 3/2.5 = 1.2

Here for this approach higher Compression Factor is preferable because it means each granted key can decrypt more ciphertexts.



(a) (b)

Fig 3 : Key assignment for 3 classes in tree hierarchy of height h=4

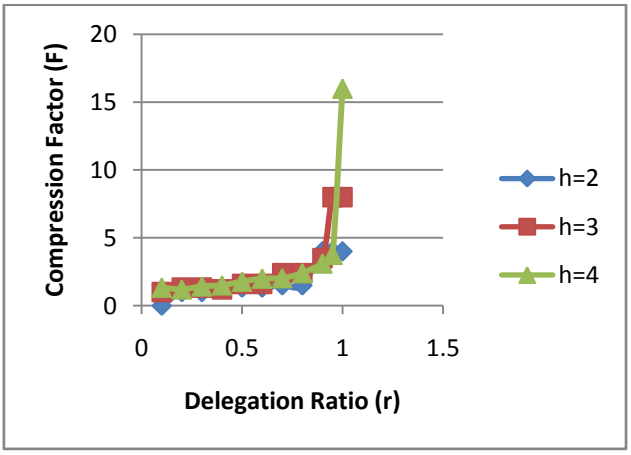


Fig 4 : Compression Factor (F) for Different Delegation Ratio

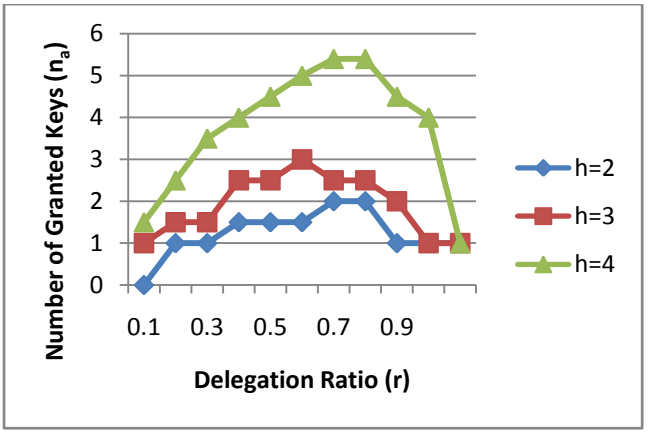


Fig 5 : Number of Granted Keys (n_a) required for Different Delegation Ratio

5. FUTURE WORK

To share data flexibly through cloud computing users prefers to upload encrypted data on cloud. We can provide additional security by verifying the encrypted data by admin of cloud before uploading it to cloud. It is also important to provide additional security while sharing decryption key with users.

6. CONCLUSION

Our Key-Aggregate Cryptosystem ensures that the cipher text and aggregate key are of constant size. Use of Elliptic Curve Cryptography (ECC) in addition provides advantage of shorter key length providing faster computations. ECC can provide a level of security with a 164-bit key where other systems require 1024-bit key for same level of security. As ECC provides more security with lower computing power and low resource usage , it is widely used for mobile applications.

7. REFERENCES

[1] Cheng-Kang Chu, Sherman S. M. Chow, Wen-Guey Tzeng, Jianying Zhou, and Robert H. Deng," Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage," IEEE Transactions on Parallel and Distributed Systems. Volume: 25, Issue: 2. Year :2014

[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] R. S. Sandhu, "Cryptographic Implementation of a Tree Hierarchy for Access Control," Information Processing Letters, vol. 27, no. 2, pp. 95–98, 1988.

[4] 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.

[5] 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.

[6] 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.

[7] 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 Communications Security (CCS '06). ACM, 2006, pp. 89–98.

[8] 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.

[9] 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, pp. 185–194.

[10] L. Hardesty, "Secure computers aren't so secure," MIT press, 2009, http://www.physorg.com/news176107396.html.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," ACM Transactions on Information and System Security (TISSEC), vol. 9, no. 1, pp. 1–30, 2006.

[20] D. Boneh, C. Gentry, and B. Waters, "Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys," in Proceedings of Advances in Cryptology - CRYPTO '05, ser. LNCS, vol. 3621. Springer, 2005, pp. 258–275.