IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 12, December 2015

Owner Secured Data Storage on Cloud and Sharing with Key Aggregate Cryptosystem

Mr.Nilesh Deshmukh [1], Mr.Arivanantham Thangavelu [2], Mr.Vaibhav Muddebihalkar [3]

Assistant Professor, Computer Engineering Department, DYPIET, Pimpri, Pune[1,2,3]

Abstract: Data sharing is an important functionality in cloud storage. This work, show how to efficiently, securely, and flexibly share data with others in cloud storage. It describes new public-key cryptosystems that produce constant-size ciphertexts such that efficient delegation of decryption rights for any set of ciphertexts is possible. More importantly one can aggregate any set of secret keys and make them as compact as a single key, but having the power of all the keys being aggregated. This compact aggregate key can be conveniently sent to others or be stored in a smart card with very limited secure storage.

Keywords: Cloud storage, data sharing, encryption, key-aggregation.

I. INTRODUCTION

Cloud storage is nowadays very popular. Cloud storage is storing of data to the physical storage which is maintained by third party. Cloud storage is about saving of digital data in logical pool and physical storage spans multiple servers which are manage by third party. Third party is responsible for making data available and accessible and physical environment should be protected and running without failed [22]. Instead of storing data to the hard drive or any other local storage, we save data to remote storage which is accessible from anywhere and anytime. Resulting reduced efforts of carrying physical storage to everywhere. By using cloud storage we can access information from any computer through internet which omitted limitation of accessing information from same computer where it is stored. While considering data privacy, we cannot always rely on traditional process of authentication, as unexpected privilege escalation will expose all data. The solution is to encrypt data before uploading to the cloud server with user's own key. Data sharing is again important functionality of cloud storage, because user is able to data from anywhere and anytime to anyone. For example, organization may grant permission to access part of sensitive data to their employees. Although challenging task is to share encrypted data. Traditional way is user can download the encrypted data from storage and decrypt that data and send it to share with others, but it loses the importance of cloud storage. Cryptography technique can be applied in a two major ways- one is symmetric key encryption and other is asymmetric key encryption. In symmetric key encryption, same keys are used for encryption and decryption[2], [3]. By contrast, in asymmetric key encryption, public key is used for encryption and private key for decryption. Using asymmetric key encryption is more flexible [1]. Here it is illustrated by following example. Suppose Alice puts all data on drop box and she does not want to expose her data to everyone. Due to data leakage possibilities she does not trust on privacy mechanism provided by drop box, so she encrypts all data before uploading to the server. If Bob ask her to share some data then Alice use share function of Drop Box. But how to share encrypted data is problem now. There are two severe ways: 1. Alice encrypt data with single secret key and share that secret key directly with the Bob. 2. Alice can encrypt data with distinct keys and send Bob corresponding keys to Bob via secure channel. In first method, unwanted data also gets expose to the Bob, which is inadequate. In second approach, same no. of keys are required as many as shared files, which may be hundred or thousand, also transferring these keys require secure channel and storage space which can be expensive.

Therefore best solution to above problem is Alice encrypts data with distinct public keys, but send single decryption key of constant size to Bob. As the decryption key should be sent via secure channel and kept secret small size is always enviable. To design an efficient public-key encryption scheme, supporting 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)

II. RELATED WORK

This section we compare our basic KAC scheme with other possible solutions on sharing in secure cloud storage. We summarize our comparisons in Table 1.

A. SYMMETRIC-KEY ENCRYPTION WITH COMPACT KEY

Benaloh et al. [2] presented an encryption scheme which is originally proposed for concisely transmitting large number of keys in broadcast scenario [3]. 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. The derivation of the key for a set of classes (which is a subset of all possible cipher text classes) is accomplished in following scenario. Selected p and q are two large random primes and composite modulus is taken. A master secret key is chosen randomly. Each class is associated with a distinct prime. These prime numbers can be the public system parameter. A constant-size key for set can be generated. The access rights for S' can be generated for delegate. But it works for the symmetric-key setting only. Corresponding secret keys for encryption of data has to get by content provider and is not suitable for many applications. As method is used to generate a secret value and not a pair of public/secret keys, its use for public-key encryption scheme is unclear. Concluding that there are schemes in symmetric-key encryption, to reduce the key to achieve authentication [4]. However, providing decryption power is not a purpose in these schemes.

B. IBE WITH COMPACT KEY

IBE (Identity-based encryption) [5], [6], [7] is a 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, mobile number). There is a private key generator (PKG) in IBE which holds a master-secret key and issues a secret key to each user with respect to the identity of user. Using the public parameter and a identity of user the content provider can encrypt a message, which recipient can decrypt by his secret key. Guo et al. [8], [9] tried to build IBE with key aggregation. Where, key aggregation is constrained such that all keys to be aggregated must belong to different - identity divisions. But there are an exponential number of identities and hence secret keys, aggregation can only be achieved on a polynomial number of them [1]. This increases the costs of storing and transmitting ciphertexts significantly, leading many times to impractical situations such as shared cloud storage. An alternate way is to apply hash function repeatedly on string denoting the class, until it produces a prime as the output of the hash function. [1] We mentioned our schemes feature constant ciphertext size, holding their security in the standard model. In fuzzy IBE [10], one single compact secret key can decrypt ciphertexts encrypted under many identities which are close in a certain metric space. Thus it is not suitable for an arbitrary set of identities and therefore not satisfies our idea of key aggregation.

C. ATTRIBUTE-BASED ENCRYPTION

In Attribute-based encryption scheme (ABE) [11], [12] each ciphertext is associated with an attribute, and allows the master-secret key holder to extract a secret key for a policy of these attributes so that a ciphertext can be decrypted using this key if its associated attribute conforms to the policy. For example, the secret key for the policy (1 V 4 V 7 V 8), can have power to decrypt ciphertext tagged with class 1, 4, 7 or 8. However, the major concern in ABE is collusion-resistance but not the compactness of secret keys. However the size of the key often increases linearly with the number of attributes it encompasses, or the ciphertext-size is not constant (e.g., [13]).

III. KEY-AGGREGATE CRYPTOSYSTEM

In key-aggregate cryptosystem (KAC)[1], users encrypt a message under a public-key, along with an identifier called class of ciphertext. The ciphertexts are further categorized into different classes. Also the key owner holds a master-secret called master-secret key, and can be used to extract secret keys for different classes. The novelty is that, the extracted key can be an aggregate key which is as compact as a secret key for a single class, still aggregates the power of many such keys, i.e., having the decryption power for any subset of ciphertext classes.[1]

With our example, Alice can send Bob a single aggregate key through a secure e-mail. Whereas Bob can download the encrypted shared photos of from Alice's cloud space and then use this aggregate key to decrypt these encrypted data. The ciphertext, public-key, master-secret key and aggregate key in this schemes are all of constant size. The public system parameter is linear in size to the number of ciphertext classes, still only a small part of it is needed each time and it can be accepted on demand from large (but non-confidential) cloud storage.

Different Schemes	**Ciphertext size**	**Decryption key size**	**Encryption type**
Key assignment schemes	Constant	Non-constant	Symmetric or public-key
Symmetric-key encryption with compact key	Constant	Constant	Symmetric key
IBE with compact key	Non-constant	Constant	Public key
Attribute based encryption	Constant	Non-constant	Public key
KAC	Constant	Constant	Public key

Table 1: Comparison between KAC scheme and other related scheme

IV. PROPOSED FRAMEWORK

Cheng-Kang Chu et. al has suggested the basic idea for Key-Aggregate Cryptosystem [1]. We are considering this basic framework for our work. The data owner establishes the public system parameter through Setup and generates a public/master-secret key pair through KeyGen. Data can be encrypted using Encrypt by anyone who has knowledge, what ciphertext class is associated to the plaintext message to be encrypted. By using the master-secret key data owner can generate an aggregate decryption key for a selected set of ciphertext classes via Extract. The generated keys can be sent to delegates securely through secure e-mails or 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. Key aggregate encryption schemes consist of five polynomial time algorithms as follows:

1. Setup (1λ, n): The data owner establishes public system parameter via Setup. For input of a security level parameter 1λ and number of ciphertext classes n, produces the public system parameter *param* as output.
2. KeyGen: It is executed by data owner to randomly generate a public/ master-secret key pair (Pk, msk).
3. Encrypt (pk, i, m) : It is executed by data owner and for message m and index i ,it computes the ciphertext as C.
4. Extract (msk, S): It is executed by data owner for delegating the decrypting power for a certain set of ciphertext classes and it outputs the aggregate key for set S denoted by Ks.
5. Decrypt (Ks, S, I, C): It is executed by a recipient of an aggregate key Ks generated by Extract. On input Ks, set S, an index i denoting the ciphertext class ciphertext C belongs to and output is decrypted result m.

A canonical application of KAC is data sharing. The key aggregation property is useful when we expect delegation to be efficient and flexible.

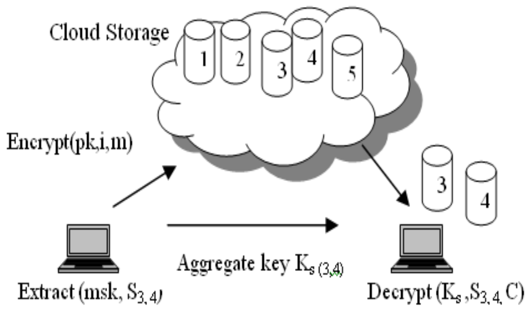

Figure 1: Using KAC for data sharing in cloud storage.

The KAC schemes enables a data owner to share her data in a confidential and selective way, with a fixed and short ciphertext expansion, by sending to each authorized user a single and small aggregate key. Data sharing in cloud storage using KAC, is illustrated in Figure 1. Suppose Alice wants to share her data m1, m2,...., mn on the server. She first performs Setup (1λ, n) to get param and execute KeyGen 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 can then encrypt each mi by Ci = Encrypt (pk, i, mi). The encrypted data are uploaded to the server. Using param and pk, people who cooperate with Alice can update Alice's data on the server. When Alice wants to share a set S with a friend Bob from her data, she can compute the aggregate key KS by performing Extract (msk, S). Since KS is just a constant size key, it is easy to be sent to Bob through a secure e-mail. By obtaining the aggregate key, Bob can download the data from cloud server, which he is authorized to access. That is, for each i € S, Bob downloads Ci from the server. Using the aggregate key KS, Bob can decrypt each Ci by Decrypt (KS, S, i, Ci) for each i € S.

V. PERFORMANCE ANALYSIS

In performance analysis we have compared the execution time of encryption of various size of file as the time taken will be significant and will affect the uploading time of that file. So we compared the process of upload and encryption in terms of time. Moreover the process of uploading starts simultaneously as soon as encryption of input blocks starts.

1) Encrypt/ Upload:

Input Size in KB	Encrypt Time in second	Upload Time in second	Difference
10	0.073	0.086	0.013
20	0.04	0.059	0.019
30	0.079	0.081	0.002
40	0.052	0.056	0.004
50	0.105	0.113	0.008
60	0.086	0.107	0.021
70	0.099	0.105	0.006
80	0.097	0.101	0.004
90	0.184	0.194	0.01
100	0.192	0.222	0.03

Table 2: Result table for Encrypt vs Upload

1. We observed that the data encryption takes less than 30 seconds, in order to encrypt 1MB data file and less than 10 seconds for a file up to 500 kb.
2. We have also noticed by analyzing various results of encrypting and uploading time, the uploading time of any size of data is directly dependent on the encrypting time of that data and difference varies between 100 to 200 ms or 5% more than encryption.

2) Upload/Update:

1. We have proposed the update i.e modification of files which are again uploaded on cloud in encrypted form.
2. We analyzed the corresponding update process with regular upload.

Upload Size in KB	Time in second	Update Size in KB	Time in second
100	1.52	150	2.7
150	2.6	200	3.5
200	3.6	250	4.6
250	4.1	300	5.3
300	5.5	350	6.8
350	7.3	400	8.8
400	9.2	450	9.2
450	9.8	500	10.9
500	10.8	550	11.3

Table 3: Upload Vs Update response in range 100KB -500 KB

We concluded that the update process has similar characteristics of regular uploading with linear growth or decrease in time depending on increasing or decreasing the size of file.

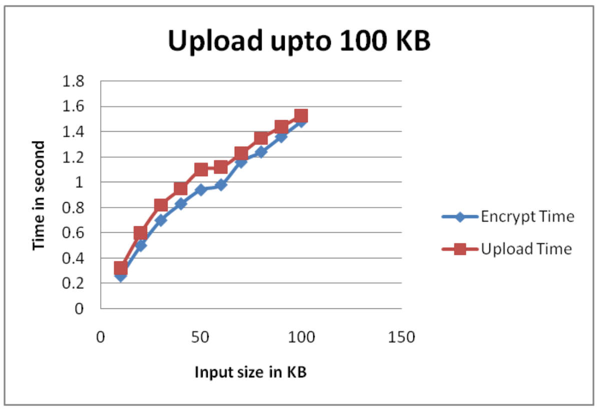

Figure 2: Encryption Vs Upload time

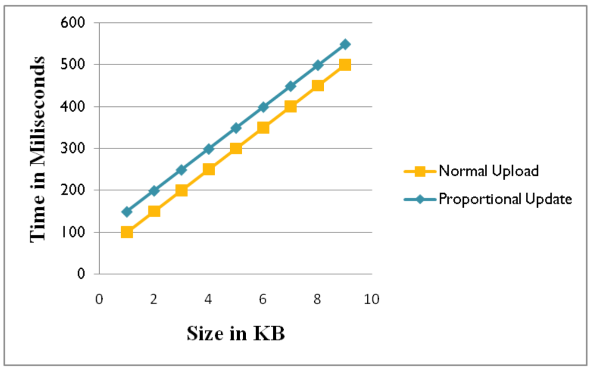

Figure 3: Upload/Update time response 100 to 500 kb

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 12, December 2015

VI. CONCLUSION

Users' data privacy is a central question of cloud storage. In public-key cryptosystems compress secret keys which support delegation of secret keys for different cipher text classes in cloud storage. It does not matter which one of the power set of classes, the delegate always gets a constant size aggregate key. In cloud storage, the data usually grows rapidly without any restrictions. So we need to expand the public-key. Although the parameter can be downloaded with cipher texts, it would be desirable if its size is independent of the maximum number of cipher text classes.

ACKNOWLEDGEMENT

I would like to extend my gratitude to many people who helped me to bring this paper fruition. First I would like to thank **Prof. Aditi Jahagirdar**. I am so deeply grateful for her help, professionalism, and valuable guidance throughout this paper. I would also like to thank to my friends and colleague. This accomplishment would not have been possible without them. Thank you.

REFERENCES

[1] Cheng-Kang Chu ,Chow, S.S.M, 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] 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.

[3] J. Benaloh, Key Compression and Its Application to Digital Fingerprinting," Microsoft Research, Tech. Rep., 2009.

[4] B. Alomair and R. Poovendran, "Information Theoretically Secure Encryption with Almost Free Authentication," J. UCS, vol. 15, no. 15, pp. 2937–2956, 2009.

[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] 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.

[8] F. Guo, Y. Mu, and Z. Chen, "Identity-Based Encryption: How to Decrypt Multiple Ciphertexts Using a Single Decryption Key," in Proceedings of Pairing-Based Cryptography (Pairing '07), ser. LNCS, vol. 4575. Springer, 2007, pp. 392–406.

[9] F. Guo, Y. Mu, Z. Chen, and L. Xu, "Multi-Identity Single-Key Decryption without Random Oracles," in Proceedings of Information Security and Cryptology (Inscrypt '07), ser. LNCS, vol. 4990. Springer, 2007, pp. 384–398.

[10] 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.

[11] 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.

[12] 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.

[13] 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, pp. 138–159.

[14] S. S. M. Chow, Y. J. He, L. C. K. Hui, and S.-M. Yiu, "SPICE - Simple Privacy-Preserving Identity-Management for Cloud Environment," in Applied Cryptography and Network Security - ACNS2012, ser. LNCS, vol. 7341. Springer, 2012, pp. 526543.

[15] L. Hardesty, "Secure computers arent so secure," MIT press, 2009, http://www.physorg.com/news1761073.

[16] 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. 362375, 2013.

[17] B. Wang, S. S. M. Chow, M. Li, and H. Li, "Storing Shared Dataon the Cloud via Security- Mediator," in International Conference on Distributed Computing Systems - ICDCS 2013. IEEE, 2013.

[18] S. S. M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R. H. Deng, "Dynamic Secure Cloud Storage with Provenance," in Cryptography and Security: From Theory to Applications Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday, ser. LNCS, vol. 6805. Springer, 2012, pp. 442464.

[19] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Variably Encrypted Signatures from Bilinear Maps," in Proceedings of Advances in Cryptology – EUROCRYPT 03, ser. LNCS, vol. 2656. Springer, 2003, pp. 416432.

[20] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, "Dynamic and E_cient Key Man- agement for Access Hierarchies," ACM Transactions on Information and System Security (TISSEC), vol. 12, no. 3, 2009

[21] D. Naor, M. Naor, and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers," Proc. Advances in Cryptology Conf.(CRYPTO'01),pp.41-62,2001.

[22] 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.

[23] 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.

[24] 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.