International Journal of Engineering Research and General Science Volume 2, Issue 6, October-November, 2014
ISSN 2091-2730

Data sharing in cloud storage with key-aggregate cryptosystem.

Mrs. Komal Kate[1], Prof. S. D. Potdukhe[2]

[1]PG Scholar, Department of Computer Engineering, ZES COER, pune, Maharashtra

[2]Assistant Professor, Department of Computer Engineering, ZES COER, pune, Maharashtra

Abstract— Cloud storage is a storage of data online in cloud which is accessible from multiple and connected resources. Cloud storage can provide good accessibility and reliability, strong protection, disaster recovery, and lowest cost. Cloud storage having important functionality i.e. securely, efficiently, flexibly sharing data with others. New public–key encryption which is called as Key-aggregate cryptosystem (KAC) is introduced. Key-aggregate cryptosystem produce constant size ciphertexts such that efficient delegation of decryption rights for any set of ciphertext are possible. Any set of secret keys can be aggregated and make them as single key, which encompasses power of all the keys being aggregated. This aggregate key can be sent to the others for decryption of ciphertext set and remaining encrypted files outside the set are remains confidential.

Keywords— Cloud storage, Key-aggregate cryptosystem (KAC), Ciphertext, Encryption, Decryption, secret key.

INTRODUCTION

Cloud storage is nowadays very popular storage system. Cloud storage is storing of data off-site to the physical storage which is maintained by third party. Cloud storage is saving of digital data in logical pool and physical storage spans multiple servers which are manage by third party. Third party is responsible for keeping data available and accessible and physical environment should be protected and running at all time. 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. It reduces 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 rely on traditional technique of authentication, because unexpected privilege escalation will expose all data. Solution is to encrypt data before uploading to the server with user's own key. Data sharing is again important functionality of cloud storage, because user can share data from anywhere and anytime to anyone. For example, organization may grant permission to access part of sensitive data to their employees. But challenging task is that how to share encrypted data. Traditional way is user can download the encrypted data from storage, 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. By contrast, in asymmetric key encryption different keys are used, public key for encryption and private key for decryption. Using asymmetric key encryption is more flexible for our approach. This can be illustrated by following example.

Suppose Alice put all data on Box.com 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 Box.com, so she encrypt all data before uploading to the server. If Bob ask her to share some data then Alice use share function of Box.com. But problem now is that how to share encrypted data. 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 approach, unwanted data also get expose to the Bob, which is inadequate. In second approach, no. of keys is as many as no. of shared files, which may be hundred or thousand as well as 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. Since 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 which supports 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).[1]

RELATED WORK

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 ciphertext classes) is as follows. A composite modulus is chosen where p and q are two large random primes. A master secret key is chosen at random. Each class is associated with a distinct prime. All these prime numbers can be put in the public system parameter. A constant-size key for set can be generated. For those who have been delegated the access rights for S′ can be generated. However, it is designed for the symmetric-key setting instead. The content provider needs to get the corresponding secret keys to encrypt data, which is not suitable for many applications. Because 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, we note that there are schemes which try to reduce the key size for achieving authentication in symmetric-key encryption, e.g., [4]. However, sharing of decryption power is not a concern in these schemes.

IBE WITH COMPACT KEY

Identity-based encryption (IBE) (e.g., [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 user identity. The content provider 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. [8], [9] tried to build IBE with key aggregation. 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.[1] This significantly increases the costs of storing and transmitting ciphertexts, which is impractical in many situations such as shared cloud storage. As Another way to do this is to apply hash function to the string denoting the class, and keep hashing repeatedly until a prime is obtained as the output of the hash function.[1] we mentioned, our schemes feature constant ciphertext size, and their security holds 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, but not for an arbitrary set of identities and therefore it does not match with our idea of key aggregation.

ATTRIBUTE-BASED ENCRYPTION

Attribute-based encryption (ABE) [11], [12] 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 $(1 \vee 3 \vee 6 \vee 8)$, one can decrypt ciphertext tagged with class 1, 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., [13]).

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

KEY-AGGREGATE CRYPTOSYSTEM

In key-aggregate cryptosystem (KAC), users encrypt a message not only under a public-key, but also under an identifier of 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 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 ciphertext classes.[1]

With our example, Alice can send Bob a single aggregate key through a secure e-mail. Bob can download the encrypted photos from Alice's Box.com space and then use this aggregate key to decrypt these encrypted data. The sizes of ciphertext, public-key, master-secret key and aggregate key in 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 non-confidential) cloud storage.

FRAMEWORK

The data owner establishes the public system parameter through Setup and generates a public/master-secret key pair through KeyGen. Data 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 key pair to generate an aggregate decryption key for a set of ciphertext classes through Extract. The generated keys can be passed to delegatees securely through 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. Key aggregate encryption schemes consist of five polynomial time algorithms as follows:

1. Setup (1^{λ} , n) : The data owner establish public system parameter via Setup. On input of a security level parameter 1^{λ} and number of ciphertext classes n , it outputs the public system parameter *param*

2. KeyGen: It is executed by data owner to randomly generate a public/ master-secret key pair (P_k, 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 delegate who received, 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.

SHARING ENCRYPTED DATA

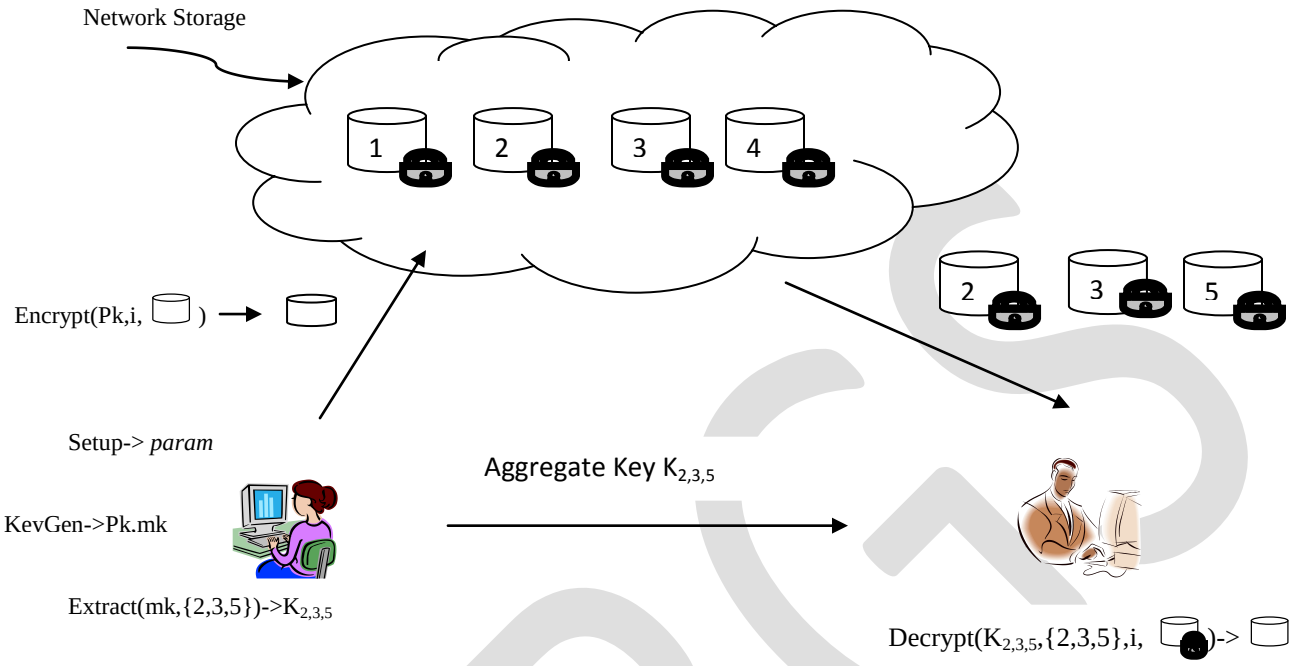

Figure. 1. Using KAC for data sharing in cloud storage.

A canonical application of KAC is data sharing. The key aggregation property is especially useful when we expect delegation to be efficient and flexible. The KAC schemes enable a content provider to share her data in a confidential and selective way, with a fixed and small ciphertext expansion, by distributing to each authorized user a single and small aggregate key.

Data sharing in cloud storage using KAC, illustrated in Figure 1. Suppose Alice wants to share her data m1,m2,....,m_n on the server. She first performs Setup (1^{λ}, 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. With param and pk, people who cooperate with Alice can update Alice's data on the server. Once Alice is willing to share a set S of her data with a friend Bob, she can compute the aggregate key KS for Bob 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. After obtaining the aggregate key, Bob can download the data he is authorized to access. That is, for each i ϵ S, Bob downloads Ci from the server. With the aggregate key KS, Bob can decrypt each Ci by Decrypt (K_S, S, i, Ci) for each i ϵ S.

CONCLUSION

Users data privacy is a central question of cloud storage. 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 delegatee can always get an aggregate key of constant size. In cloud storage, the number of cipher texts usually grows rapidly without any restrictions. So we have to reserve enough cipher text classes for the future extension. Otherwise, we need to expand the public-key. 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.

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