RESEARCH ARTICLE OPEN ACCESS

Data Sharing in Cloud Storage Using Aggregate Key Cryptosystem

Balaji M [1], Rajashekar S A [2]
Student, Assistant Professor
Department of Computer Science and Engineering
East West Institute of Technology
Bengaluru, Karnataka

ABSTRACT
Data sharing is an important functionality in cloud storage. In this article, 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 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.
Keywords:- Cloud storage, data sharing, key-aggregate encryption, Identity-based encryption, patient-controlled encryption

I. 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.

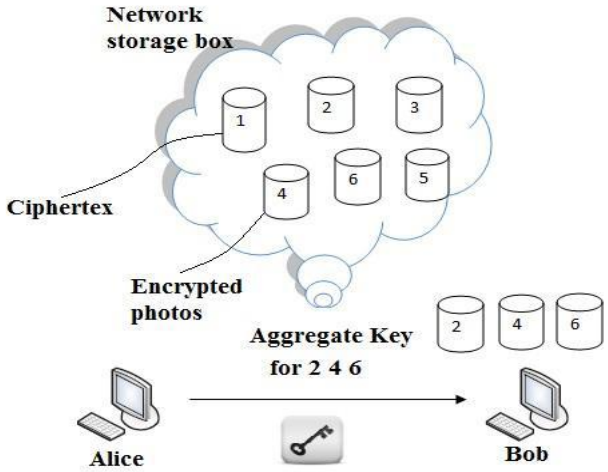

Fig 1 File sharing between Alice and Bob

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].

II. LITERATURY SURVEY

There exist several expressive ABE schemes where the decryption algorithm only requires a constant number of pairing computations. Recently, Green *et al.* proposed a remedy to this problem by introducing the notion of ABE with outsourced decryption, which largely eliminates the decryption overhead for users. Based on the existing ABE schemes, Green *et al.* also presented concrete ABE schemes with outsourced decryption.

In these existing schemes, a user provides an untrusted server, say a proxy operated by a cloud service provider, with a transformation key TK that allows the latter to translate any ABE ciphertext CT satisfied by that user's attributes or access policy into a simple ciphertext CT', and it only incurs a small overhead for the user to recover the plaintext from the transformed ciphertext CT'. The security property of the ABE scheme with outsourced decryption guarantees that an adversary (including the malicious cloud server) be not able to learn anything about the encrypted message; however, the scheme provides no guarantee on the correctness of the transformation done by the cloud server. In the cloud computing setting, cloud service providers may have strong financial incentives to return incorrect answers, if such answers require less work and are unlikely to be detected by users.

Considering data privacy, a traditional way to ensure it is to rely on the server to enforce the access control after authentication, which means any unexpected privilege escalation will expose all data. In a shared-tenancy cloud computing environment, things become even worse.

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, or without compromising the data owners anonymity. 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, 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.

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 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.

B. 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.

C. 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]).

In a multi attribute-authorities numbers of attributes are analyzed regarding the decryption key and the user must get a particular key related to the attribute while decrypting a message. The decryption keys are allocated independently to users those who have attribute identity without interaction between each other. Multi-authority attribute-based encryption allows real time deployment of attribute based privileges as different attributes are issued by different authorities.

III. PROPOSED SYSTEM

For sharing selected data on the server Alice first performs the Setup. Later the public/master key pair (pk, mk) is generated by executing the KeyGen. The msk master key is kept secret and the public key pk and param are made public. Anyone can encrypt the data m and this data is uploaded on server. With the decrypting authority the other users can access those data. If Alice is wants to share a set S. of her data with a friend Bob then she can perform the aggregate key KS for Bob by executing Extract (mk, S). As kS is a constant size key and the key can be shared through secure e-mail. When the aggregate key has got Bob can download the data and access it.

IV. IMPLEMENTATION

1 Setup Phase

The data owner executes the setup phase for an account on server which is not trusted. The setup algorithm only takes implicit security parameter.

2 KeyGen Phase

This phase is executed by data owner to generate the public or the master key pair (pk, msk).

3 Encrypt Phase

This phase is executed by anyone who wants to send the encrypted data. Encrypt (pk, m, i), the encryption algorithm takes input as public parameters pk, a message m, and i denoting ciphertext class. The algorithm encrypts message m and produces a ciphertext C such that only a user that has a set of attributes that satisfies the access structure is able to decrypt the message.

- Input= public key pk, an index i, and message m
- Output = ciphertext C.

4 Extract Phase

This is executed by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegate.

- Input = master-secret key mk and a set S of indices corresponding to different classes
- Outputs = aggregate key for set S denoted by kS.

5 Decrypt Phase

This is executed by the candidate who has the decryption authorities. Decrypt (kS, S, i, C), the decryption algorithm takes input as public parameters pk, a ciphertext C, i denoting ciphertext classes for a set S of attributes.

- Input = kS and the set S, where index i = ciphertext class
- Outputs = m if i element of S

A. DATA SHARING

KAC in meant for the data sharing. The data owner can share the data in desired amount with confidentiality. KAC is easy and secure way to transfer the delegation authority. The aim of KAC is illustrated in Figure 2.

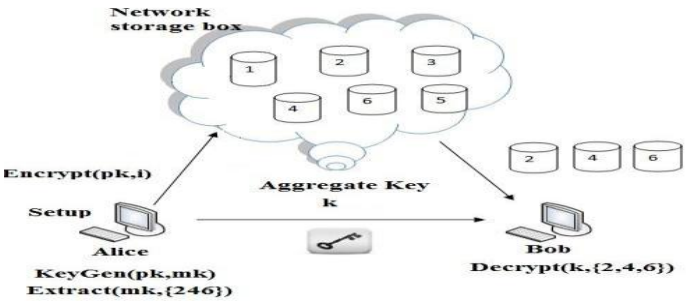

Fig 2 Use of KAC for data sharing

V. CONCLUSIONS

To protect 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. Here, 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.

ACKNOWLEDGMENT

I would like to thank all the people who have helped in completion of my dissertation work. To name a few my project guide Asst. prof. Mr. Rajashekar, who motivated me to present/ publish paper in international journal and conference, my HOD Dr. Arun Biradar and my Principal of EWIT Dr. K. Channakeshavalu for their constant support and guidance

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. 886 www.ijergs.org

[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.