Asymmetric Key Aggregate Encryption Technique for Data Sharing in Cloud Storage

Shilpashree P[1]
Department of Computer Science and Engineering
Vemana IT, Vishweshwarayya Technological University
Belagavi, Karnataka, India.

Dr. K. N. Narasimha Murthy[2]
Department of Computer Science and Engineering
Vemana IT, Vishweshwarayya Technological University
Belagavi, Karnataka, India.

***Abstract* - The data sharing is a kind of functionality in cloud storage and it is very essential too. In the cloud storage the data can be shared with others via secured, efficient and economic mode. In this paper it describes the technique that characterise unique or new public- key cryptosystems. It produces constant-size ciphertexts such as qualified authorisation of decryption rights for any set of ciphertexts. The uniqueness works as anyone can combine any set of secret keys and make them as compressed single key by encircling the power of all the keys being combined. In other words, the secret key holder can release a constant-size aggregate key for flexible choices of ciphertext set in cloud storage, but outside the set the other encrypted files remain confidential. With very limited secure storage the compact aggregate key can be conveniently sent to others or be stored in a smart card. It provides formal security analysis of these schemes in the standard model.**

Index Terms — asymmetric, cloud storage, data sharing, key aggregate encryption.

I. INTRODUCTION

Cloud Storage based data sharing is a convenient and important functionality in current trend [27]. It shows how to share the data in an secured, efficient, and economic way with other cloud storage providers. Data from different clients can reside on a single physical machine but they are on separate Virtual Machines (VMs). Data sharing is an important functionality in cloud storage because data in a target VM could be stolen by instantiating another VM co-resident with the target machine[1]. For example, the data owner can let their friends to view a subset of their own private pictures. Owner may grant their employees access to a portion of acute data. The challenging problem is how to effectively share the encrypted data. The users can download the encrypted data from stored source, decrypt it, and then send to others for sharing, but doing this can loses the value of cloud storage. The user can access the data from the server directly but the users should be able to delegate the access rights of the sharing data to others.

Recently cloud storage is gaining popularity. In enterprise settings, the cloud storage is the rise in demand for data outsourcing, and it assists in the strategic management of corporate data. Behind many online services for personal applications the cloud storage is also used as a core technology. As we know that nowadays, it is easy to apply for free accounts for email, file sharing, photo album and/or remote access, with more than 25GB storage size. Together users can access almost all their files and as well as emails by a mobile phone with the current wireless technology in any corner of the world. The consideration of data privacy, a traditional way to ensure it is to rely on the server to enforce the access control in data privacy after authentication.

An illustration in Figure 1 that Agent-1 puts all their private photos on Dropbox and some of the photos of them does not want to expose to everyone. Agent-1 cannot feel relieved by just relying on the privacy protection mechanisms provided by Dropbox due to various data leakage possibility, so their own keys can encrypts all the photos before uploading. One day, Agent-2, who is the friend of Agent-1, asks

Agent-1 to share the photos taken over all these years which Agent-2 appeared in. The share function of Dropbox can be used by Agent-1, but the problem that how to give the delegation rights for these photos to Agent-2. The possible way is to send secret key to Agent-2 securely in which secret keys are allowed.

Naturally, there are two extreme ways for Agent-1 under the traditional encryption paradigm:

1. First, Agent-1 encrypts all files with a single key encryption and gives Agent-2 the corresponding secret key directly.
2. The second, Agent-1 encrypts files with distinct keys and sends Agent-2 the corresponding secret keys.

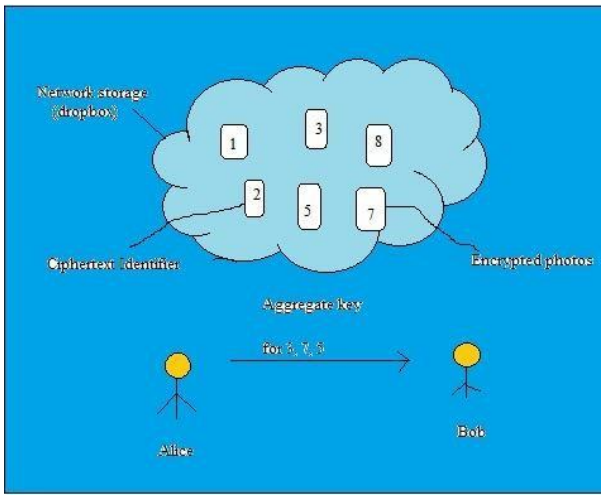


Figure 1: Agent-1 shares files with identifiers 3, 5 and 7 with Agent-2 by sending a single aggregate key.

In the two extremes obviously here in the first method is inadequate since all un-chosen data may also be leaked to Agent-2. And in the second method, there are practical concerns on efficiency. The costs and complexities involved generally increase with the number of the decryption keys to be shared. Transferring these secret keys inherently requires a channel which is secured, and storing these keys needs rather expensive secure storage. In short, it is very heavy as well as costly to do that.

Encryption keys also come with two flavours they are

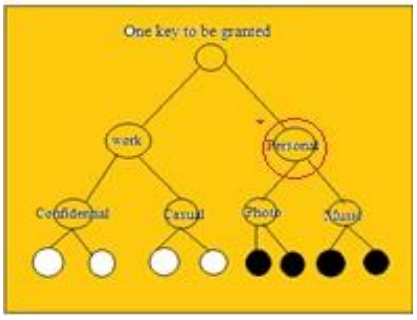
(1) Symmetric key or
(2) Asymmetric (public) key

Using symmetric encryption, Agent-1 has to give the secret key for the encryptor when they want their data to be originated and this is not desirable always. The decryption and the encryption key are different in public-key encryption. The use of this encryption gives more flexibility for the applications. For example, without the knowledge of the company's master-secret key in the enterprise settings, everybody can upload the data that is encrypted in the cloud storage. So, the better solution for this problem is here the Agent-1 encrypts all her files with distinct public-keys, and then only sends the Agent-2 a single decryption key. Since the decryption key must be sent through some channel which is secured and confidential. Here small key size is always good to consider. Because the large storage for decryption key cannot expect like the devices smart cards and smart phones. These secret keys are stored in tamper- proof memory due to expensive costs.

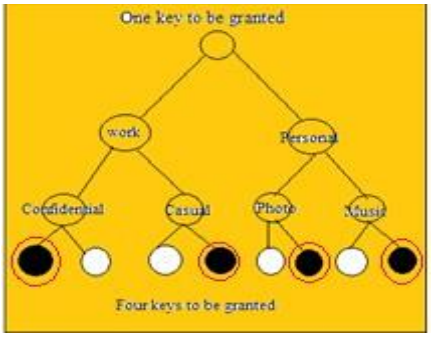
So the paper is organised as follows: In the section 2, literature survey is discussed. In the section 3, key aggregate encryption is discussed.

II. LITERATURE SURVEY

Cryptographic Keys for a Predefined Hierarchy using a tree structure, a key can be used to derive the descendent nodes of keys [11], [12], [13], [14]. It is like granting the parent key it implicitly grants all the descendant nodes of the keys. A method for tree hierarchy of symmetric keys is by using pseudorandom function/block-cipher on a fixed secret [16], [17], [7]. Most of these schemes produce symmetric-key cryptosystems for the keys. The example of this is tree structure.



(a)



(b)

Figure 2: Compact key is not always possible for a fixed hierarchy.

In the above Figure 2, Agent-1 can classify the ciphertext classes according to the function. A secret key is represented by each node and individual ciphertext classes are represented by the leaf nodes. The classes to be delegated are represented by the filled circles and keys to be granted are represented

by the circles circumvented by the dotted lines. The every key of the non-leaf node can derive the keys of its descendant nodes.

Compact Key in Identity Based Encryption (IBE) (e.g., [20], [21], [22]) is a type of public-key encryption and in which the public key of a user can be set as an identity-string of the user. Private key generator is the trusted party in IBE and it holds a master-secret key and it issues a secret key to the each user w.r.t user identity. Public parameter and the user identity can be used by the encryptor to encrypt a message. The decryptor can decrypt this ciphertext by his secret key. One of the schemes assumes that random oracles but some other not. Key aggregation should come from different identity divisions but there are an exponential identities in number and thus secret keys, only some of them aggregate which are polynomial numbers. The important is that the key aggregation comes [23], [9] at O(n) sizes of the expense for both public parameter and the ciphertexts, where n is the secret keys in number which can be aggregated in a size of constant one.

Compact key in Symmetric-key Encryption for supporting flexible hierarchy was motivated by the same problem in decryption power delegation but in symmetric setting. Benaloh et al. [8] presented an encryption scheme and it has originally proposed [18] for transmitting large number of keys in broadcast scenario. In the simple way the key has got constructed and they briefly review it's key derivation process for a concrete description of what are the desirable properties they want to achieve. The approach achieves similar performance and the properties of their schemes. Instead it has designed for the symmetric key settings. The corresponding secret keys the encryptor must needs to encrypt the data and which is not suitable for many applications. Considering their method is used to generate a value which is secret rather than a pair of secret/public keys, this is unclear to apply this idea for public-key encryption scheme.

Finally, they note that there schemes which try to reduce the key size in symmetric key encryption in achieving authentication, e.g., [19]. However, the sharing of decryption power is not concern in these schemes.

The other Encryption Schemes by A.Sahai et al. [5], conducted the experiments to securely share the data example like, Attribute-Based Encryption for Fine-Grained Access Control of Encrypted data. Encryption of the data is needed whenever the more sensitive data is shared and stored by third-party on the internet. The drawback of encrypting data is that it can be share to another party then the data owner has to give the private key, this level is called coarse grained level. They develop a unique cryptosystem for fine grained sharing of data that is encrypted and that they call it as Key-Policy Attribute-Based Encryption (KP-ABE). In their cryptosystem, the message or the ciphertexts are labelled with sets of attributes and the private key are with access structures that restrict which ciphertext a user is able to decrypt.

B. Waters et al, selected decryption key size as non-constant. They took the ciphertext size as constant and the encryption type as public key.

Attribute Based Encryption (ABE) [10], [24] allows each ciphertext to be combined with an attribute and the master secret key holder can extract a secret key for a policy of these attributes. So the ciphertext can be decrypted by this key if it's combined attribute conforms to the policy. The major concern in ABE is collusion resistance but not the compactness of secret key.

III. KEY AGGREGATE ENCRYPTION

A KAE called Key Aggregate Encryption scheme and it consists of five polynomial-time algorithms and are as follows. The data owner is the one who establishes the public system parameter through Setup and generates a master/public-secret key through KeyGen. Anyone who decides what ciphertext class can encrypt the messages and it is associated with the plaintext message to be encrypted. The master-secret key can be used by the data owner to generate an aggregate decryption key to a set of ciphertext classes through Extract. These keys which have got generated can be passed to the delegates in secure way via e-mails or the devices which are secure. Finally, anybody else can decrypt any ciphertext with the use of aggregate key. The ciphertext's class is contained in the aggregate key via Decrypt.

- Setup: This is executed by the data owner to setup an account on an untrusted server. On input a security level parameter and the number of ciphertext classes n (i.e., class index should be an integer bounded by 1 and n), it outputs the parameter called public system parameter param and this is omitted from the input of the other algorithms for brevity.
- KeyGen: This is executed by the data owner to generate randomly a public/master-secret key pair.
- Encrypt: This is executed by anybody who wants to encrypt the data. This is done by an input a public-key pk, an ciphertext class by denoting index i, and a message m, by doing this it outputs a ciphertext C.
- Extract: This is executed by the data owner for delegating the decrypting rights for a

certain set of ciphertext classes to a delegatee. On input the master-secret key called msk and a set S of indices corresponding to different classes, it outputs the aggregate key for set S denoted by KS.

- Decrypt: This is executed by a delegatee who received an aggregate key KS generated by Extract.

(A) Sharing Encrypted Data:

A data sharing is a kind of canonical application. This property is useful when it provide the efficient and flexible delegation. The scheme here is enable to share their data in a confidential way with a small and fixed ciphertext expansion by a single aggregate key. The cloud storage using KAC scheme is illustrated in Figure 3.

Suppose that Alice wants to share her personal data on the server, first she performs the Setup to get param and then execute the KeyGen to get the master/public secret key pair. The parameter and master-secret key both should be kept secret by Alice. Anyone or even Alice can encrypt the data and encrypted data are uploaded to the server. With the use of public and param key, the people who cooperate with Alice can update Alice's data on the server. Alice can compute the aggregate key when she willing to share a set S of her data with Alice's friend Bob. Alice can compute the aggregate key for Bob by performing Extract. Since the key is constant size so it can be easily sent to Bob via a secure e- mail. After getting the e-mail the Bob can download the data and he is the right person to access. Bob can download the file then decrypt each by using some needed values and parameters.

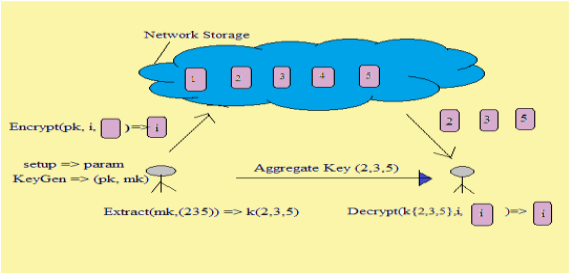


Figure 3: Using KAC for data sharing in cloud storage

IV. CONSTRUCTION OF KAC

A Basic Construction – The design is inspired from collusion-resistant encryption scheme proposed by Boneh et al.,[26]. These schemes supports constant size secret keys but every key only has the power for decrypting ciphertexts associated to a index in particular. It has come up with a new Extract algorithm and the corresponding Decrypt algorithm. It includes the following steps:

(A) Initial Setup Algorithm:

1. Based on number (n) of classes
2. Owner required to create a n pairs of Encryption key (E_k).
3. Decryption key (D_k).

(B) KAC – Key Aggregate Cryptosystem generation Algorithm:

- Owner has to select the number of classes allowed (I) for a particular users.
- Based on classes allowed, Extract the corresponding keys from a set-up keys (k_1, $k_2 \ldots k_i$).
- Compress all the keys and encrypt the file using symmetric cryptosystem and create master secret key (MSK).
- The KeyGen is executed by the data owner to generate randomly a MSK.
- Distribute MSK to data consumer.

Extract and Decrypt Algorithm:

- Data owner has to provide MSK and file requested by the data consumer.
- Verify data consumer has an access to file which they have requested.
- If pass then find the class of the file. Extract
- the key (K_c) belongs to ciphertext.
- The data consumer then decrypts the MSK.
- The data consumer decompresses the content.
- Decrypt the file using K_c.

Performance:

The encryption can be pre-computed and the decryption takes two pairings means it needs only one pairing computation within the security chip storing the secret aggregate key. Nowadays it is fast to compute a pairing even in the devices like resource constrained. The motivation of this paper is to reduce the secure storage. In public key extension, if a user needs to classify his/her ciphertexts into more than n classes, he/she can register for additional key pairs (pk, msk). Since essentially the new public key can be treated as a new user, their is a concern that the key aggregation is not possible across two independent users. Their still one can achieve shorter key size and gain flexibility illustrated in Figure 4.

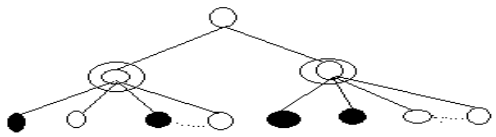


Figure 4: Key assignment approach.

Figure 4 shows the flexibility of the scheme. Here it achieves local aggregation, it means the secret keys can always be aggregated under the same branch. The extended public scheme is very similar to the basic scheme so it is omitted here. But it makes the key size as small as possible. This key extension approach can also be given as key update process. The aggregation key size is small then it minimises the communication overhead.

CONCLUSION

The protection of the data privacy is the central question of cloud storage. Cryptographic schemes are getting more versatile with more mathematics tools. Here the main consideration is to compress the secret keys in public-key cryptosystems and it supports the delegation of secret keys for ciphertext classes. Among the power set of classes, the delegate can get the aggregate key of constant size. The number of the ciphertexts in cloud storage usually grows rapidly. So enough ciphertext classes have to reserve for the future extension or else need to expand the public- key. On the other way, around in a mobile device when anyone carries the delegated keys without using any trusted hardware, the key is prompt for the leakage, the designing of leakage-resilient cryptosystem yet allows flexible and efficient delegation is also an interesting direction.

V PERFORMANCE ANALYSIS

For a comparison, first to investigate the space requirements for the file upload. Here we take a parameter like file size in kb, start time, end time and total time. To measure this we use a tool called wire shark portable as a network analyser tool. Using this tool one can measure the upload time of a file. As the size of the file increases the time is also get increases. The key aggregation never affects the file upload time and download time. The file is encrypted and then we upload and the generated key produced by the KAC is encrypted and then sent to the consumers e-mail. So the key sent to the e-mail is in unreadable format and using this key the file can be decrypted by the data consumer. The Figure 5 shows the upload time of file versus file size.

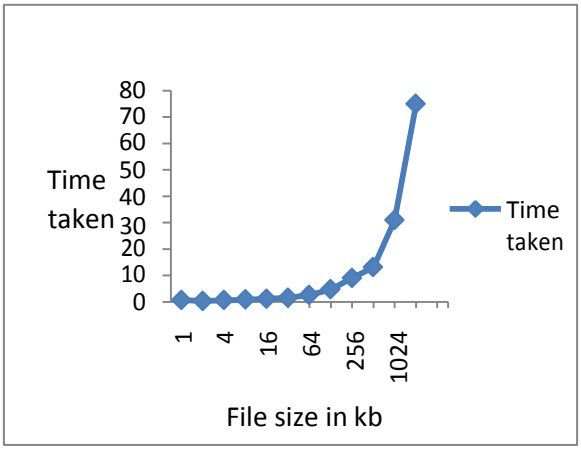


Figure 5: Upload time of file versus filesize.

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," in Applied Cryptography and Network Security – ACNS 2012, ser. LNCS, vol. 7341. Springer, 2012, pp. 526–543.

[2] L. Hardesty, "Secure computers aren't so secure," MITpress,2009,http://www.physorg.com/news176107396.ht ml.

[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, 2013.

[4] 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.

[5] 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. 442–464.

[6] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," in Proceedings of Advances in Cryptology - EUROCRYPT '03, ser. LNCS, vol. 2656. Springer, 2003, pp.416–432.

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

[8] 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.

[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] 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.

[11] 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.

:

[12] 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.
[13] 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.
[14] 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.
[15] R. S. Sandhu, "Cryptographic Implementation of a Tree Hierarchy for Access Control," Information Processing Letters, vol. 27, no. 2, pp. 95–98, 1988.
[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] Q. Zhang and Y. Wang, "A Centralized Key Management Scheme for Hierarchical Access Control," in Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '04). IEEE, 2004, pp. 2067–2071.
[18] J. Benaloh, "Key Compression and Its Application to Digital Fingerprinting," Microsoft Research, Tech. Rep., 2009.
[19] B. Alomair and R. Poovendran, "Information Theoretically Secure Encryption with Almost Free Authentication," J. UCS, vol. 15, no. 15, pp. 2937–2956, 2009.
[20] 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.
[21] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," in Proceedings of Advances in Cryptology - EUROCRYPT '05, ser. LNCS, vol. 3494. Springer, 2005.
[22] 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.
[23] 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.
[24] 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.