Secure Channels Reduction by KAC with Transmitting Aggregate Keys for Cloud Sharing Data

Mr. Mohemmed Yousuf ,Mr.Godhandaraman ,Mr.Karthick Myilvahanan

Abstract: Sharing data is an important functionality in a public cloud. The data/files are stored in a cloud in an encrypted form and data owners want to give delegate rights to the Authorized cloud users. There is a particular secure channel exist between the data/files and the users which are costlier. There is a solution for this problem is key aggregate cryptosystem which combines all the secret keys of encrypted data and form a one constant size aggregate key which is used to decrypt multiple files from the cloud. In this paper we are creating CPA and CCA secure KAC, elliptic curves are efficiently using to implement the aggregate key. These constant size aggregate keys reduce the use of secret channels.

Key words: Cloud computing, data sharing, data security, key aggregate cryptosystem, broadcast encryption

INTRODUCTION

The Cloud computing has millions of users around the world. And because of this increasing number of users the current cloud computing has pushed much restriction of data sharing capabilities for numerous applications. Government ,corporations, healthcare, social networks are used to store the data in a cloud and it has the ability to allow multiple users around the globe , cloud is also used by the social networking sites to create a more connected world used to share text as well as multimedia. The cloud computing also allows remote monitoring and diagnosis of patients with smart phones.

In spite of all its advantages, the cloud is vulnerable to privacy and security attacks that are a major interruption to its good acceptance as the

Primary means of data sharing in today's world. The IDC Enterprise panel carried out the survey in August 2008 according to that survey 75percent of security challenges found and the cloud users worried about their critical business and IT systems being at risk of attack. While security issues from external agents are common and malicious service providers must also be taken into consideration. As online data almost always resides in shared environment it is important to provide security to the data of the cloud

The some of the primary requirements of users in a cloud based data sharing service:

Data confidential: Illegal users and the cloud service providers should not be able to access the data at any given time. Data should remain confidential while sharing,

User revocation: any user's access rights must be revoked by the owner without affecting other authorized users

Scalability and efficiency: maintaining scalability and efficiency is the biggest challenge faced by the data management on the cloud in the face of very large user bases and dynamically changing data usage patterns.

The important functionality of cloud storage is data sharing. For example, users can let their friends view a subset of their private pictures or a data; an enterprise may grant her employees access to a part of private data. The challenging problem is how to effectively share encrypted data. of course the data users can able to download the encrypted data from the cloud, decrypt them, then share them with others , but it loses the value of cloud storage. Users should be able to delegate the access rights of the sharing data to others so that they can access these data from the server directly. However, finding an efficient and secure way to share partial data in cloud storage is not trivial.

A conventional way of ensuring data privacy is to depend on the server to implement access control mechanisms. This approach is prone to privilege escalation attacks in shared data environments such as the cloud, where data corresponding to multiple users could exist in on the same server. The secure online data sharing technology comes in two major flavors-trusting a third party assessor, or using the user's own key to encrypt her data while preserving secure.

In both case, a user would want a consistent and efficient cryptographic method in place, with prescribed guarantees of security, high scalability and ease of use. The main challenge in designing such a cryptosystem lies in effective sharing of encrypted data. In a cloud the data sharing design on the cloud is successful only if data owners can delegate the access rights to their data efficiently to

multiple users, who can then access the data directly from the cloud servers.

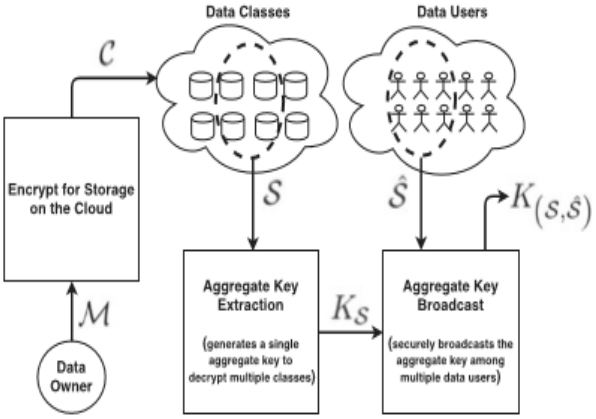

Fig1: A desirable online data sharing by aggregate key.

Fig. 1 describes a realistic online data sharing setup on the cloud. Assume that a data owner Alice is using an online data sharing service such as Microsoft One Drive to store certain classes of data (here class may refer to any data structure such as a file folder or any collection of these). She wants to add an extra layer of security for her data by storing them in an encrypted fashion. Now, she wants to share a specific subsets of these documents with a set of ˆ S of data users. For this, she needs to give each of these users with decryption rights to specific classes of the data that they are authorized to access. A naive solution is for each message class having different decryption key and shares them consequently with the designated users via secured channels. This method is not practically deployable for two major reasons. First, the number of secret keys would increase with the number of data classes. Second, any user decryption rights would require Alice to entirely re-encrypt the corresponding subset of data, and distribute the new set of keys to the other existing valid users. This makes the scheme inefficient and difficult to scale. In public key cryptosystems the decryption key is usually sent via a secure channel, smaller key sizes are desirable. Moreover, resource constrained devices such as wireless sensor nodes and smart phones cannot afford large expensive storage for the decryption keys either.

An ideal scenario, as described in Fig. 1, is where Alice can create a single constant size decryption key KS that combines the decryption rights to each of the data classes in S, and then use a public key framework to broadcast this key to the target set of users ^ S in the form of a low overhead broadcast aggregate key $K_{(S,\ S\hat{})}$. This scheme is efficient, avoids the use of secret channels which are costly and difficult to realize in practice, and is scalable to any arbitrary number of data classes and data users. In this paper, we attempt to build precisely such a data sharing framework that is provably secure and at the same time, efficiently implementable.

Related work

Identity Based Encryption (IBE): IBE is belongs to a public-key encryption category. Identity string is situating for encryption which is nothing but user's public key. In IBE, master secret (ms) keys are generated by the private key generator and based on users identity secrete key will be provided. Sender wants to share files. So sender will encrypt the files by making use of user identity and public parameter and sends the files. Receiver will decrypt these files by making use of his secret key. But out of key-aggregation and IBE, only one assumes arbitrary oracles. Key aggregation is inhibited as keys to be aggregated will come from various identities. The drawback of these methods is cipher text size is non-constant and cost of storing cipher text and transmitting it expensive.

Attribute Based Encryption (ABE): In Attribute Based Encryption technique an attribute will be related with cipher text. From master secret key, the secret key will be derived. This secret key is used to decrypt the files only if all its related attributes go after the rules. Before Attribute Based Encryption technique was introduced, the user who wanted secret key must go to third party and proving he is real by providing his identity and then he was able to decrypt the file. Later in ABE scheme the secret key of user was not allowed to a single centre. Instead it was authorized by independent authorities. But still this scheme has disadvantages i.e. no solidity on secret key. In this method there is linear increase in key size, with the increase in attributes. Disadvantages are (a) Decryption key size is non-constant. (b) Requires more space to store keys. (c) Decryption key size increases linearly. (d) Expensive of managing keys.

Symmetric Key Encryption: Benaloh proposed an encryption scheme, where a huge number of keys can be sent rapidly in a broadcast scenario. The key origin is as follows. Initially choose two prime numbers p and q for a composite module. At random, master secret key will be chosen. Dissimilar prime numbers will be allied with each class. A public system parameter is considered for which all the prime numbers will be put. The outcome of this is a constant size key. This method is designed for symmetric-key setting. So here the sender should encrypt files with corresponding secret keys which will not be feasible. The disadvantages of this system are both encryption and decryption is done by same key and encryptor should get corresponding key to encrypt files.

Hierarchical Encryption: One of the most popular techniques for access control in online data storage is to use a pre-defined hierarchy of secret keys in the form of a tree-like structure, where access to the key corresponding to any node implicitly grants access to all the keys in the sub tree rooted at that node. For instance, uses repeated evaluations of a pseudo-random function/block ciphering fixed secret to generate a tree hierarchy of symmetric keys. Some more advanced schemes extend access control to cyclic and acyclic graphs. Several provably secure identity-based flavors of hierarchical encryption have also been studied extensively in. A major disadvantage of hierarchical encryption schemes is that granting access to only a selected set of branches within a given sub tree warrants an increase in the number of granted secret keys. This in turn blows up the size of the key shared.

Our Contribution

The most important contributions of this paper can be enumerated as follows:

1) In this paper we propose a basic key aggregate cryptosystem an efficiently implementable version by using asymmetric bilinear pairings. We prove our construction to be completely secure against a non-adaptive adversary in the standard model under appropriate security assumptions. We also create that the construction is collusion resistant against any number of colluding parties.

2) We propose a CCA-secure fully collusion resistant construction for the basic KAC scheme with low overhead cipher texts and aggregate keys. To the best of our knowledge, this is the first KAC construction in the cryptographic literature proven to be CCA secure in the standard model.

3) We show how the basic KAC structure may be efficiently extended and combined with broadcast encryption schemes for distributing the aggregate key among an arbitrary number of data users in a real-life data sharing environment. The extension has a secure channel requirement of $O(m + m')$ for m data users and m0 data owners, which is an improvement over the $O(mm')$ requirement reported in [6]. In addition, the extended construction continues to have the same overhead for the public parameters, cipher texts and aggregate keys, and does not require any secure storage for the aggregate keys, which are publicly broadcast.

4) Experimental results in an actual cloud environment are presented to validate the space and time complexity requirements, as well the network and communication requirements for our proposed constructions.

PRELIMINARIES

We commence by properly defining the framework key-aggregate cryptosystem. We depict the framework in two parts, for the transparency of presentation. The basic framework focuses on generating the aggregate key for any random subset of data classes, while the extended framework aims to transmit this aggregate key among arbitrarily large subsets of data users. We also draw the game based framework for formally proving the static security of these schemes. Finally, we state the difficulty assumptions used for proving the security of these schemes.

Key-Aggregate Cryptosystem:

The Framework of KAC is a cluster of five randomized polynomial-time algorithms. The system administrator is responsible for setting up the public parameters via the Set Up algorithm. A data owner ready to share her data using this system registers to receive her own public and private key pairs, created by using the Key Gen method. The data owner is responsible for classifying each of her data files/messages into a exact class i. Each message is consequently encrypted by an Encrypt method and stored online in the cloud. When delegating the decryption rights to a particular subset of message classes, the data owner uses the Extract operation to create a constant-size aggregate decryption key unique to that subset. Finally, an authorized data user can use this aggregate key to decrypt any message belonging to any class i ∈ S. We now describe each of the five algorithms involved in KAC:

1)Set up (1^{λ}, m): Takes the number of data classes n as the input and the security parameterλ. Output the public parameter param.

2) Key Gen (): The Key gen () method outputs the public key PK and the master secret key msk for a data owner registering in the system.

3) Encrypt (param, PK, i, M): The encrypt operation takes the public key parameter PK, the data class i and the plaintext data M as the input and Outputs the subsequent cipher text C.

4) Extract (param, msk, S): The extract () operation takes the master secret key and a subset of data classes S⊆ {1, 2....., n} as input and Computes the aggregate key KS for all encrypted messages belonging to these subset of classes.

5) Decrypt (*param, C, i, S, KS)* : Decrypt () operation takes the cipher text C, the data class i and the aggregate key KS corresponding to the subset S such that $i \in S$ as an input. Output the decrypted message.

A PROVABLY SECURE BASIC KAC USING ASYMMETRIC BILINEAR PAIRINGS

In this segment, we present the design of the basic key aggregate cryptosystem introduced using asymmetric bilinear pairings that are useful and efficiently

implementable, and properly prove its crypto graphic security. The basic KAC creation serves to demonstrate how a single data owner within different classes of encrypted data online, can generate a single decryption key corresponding to any arbitrary subset S ⊆{1,..., n} of these data classes. We prove our construction to be non-adaptively CPA secure and fully collusion resistant against any number of colluding parties, under the asymmetric n-BDHE exponent assumption.

Construction

This section presents the basic KAC construction for a single data owner using asymmetric bilinear pairings. we assume the existence of equi-prime order (for a ↗-bit prime q) elliptic curve subgroups G1 and G2, along with their generators P and Q. We also assume the existence of a multiplicative cyclic group GT, also of order q with identity element 1. Finally, we assume there exists an asymmetric bilinear pairing e : G1 ×G2→GT.

Setup $(1^{\nearrow}, m)$: arbitrarily take $\propto \in$Zq. Output the system parameter as param = $(P, Q, \gamma_{p,\alpha,n}, \gamma_{Q,\alpha,n})$ Discard α.

Key Gen (): arbitrarily pick$\gamma \epsilon z_q$. Put the master secret key msk toγ. Let PK1 = γP and PK2 = γQ. Set the public key PK =(PK1, PK2). Output (msk, PK) Encrypt (param, PK, i, M). For a message M∈G_T belonging to class i ∈{1,2,...,n,} arbitrarily choose t ∈ Zq. Output the cipher text C as
C= (c_0, c_1, c_2)= $(tQ, t(PK2 + Q_i), m \cdot e(P_n, tQ1)$

Extract (param, msk, S): For the subset of class indices S, the aggregate key is computed as
KS = msk $\sum_{j\epsilon s} P_{n+1-j} = \gamma \sum_{j\epsilon s} P_{n+1-j}$.

Decrypt (param, C, i, KS): Let $C = (c_0, c_1, c_2)$.
Decrypted message is

$$\widehat{M} = c_2 \cdot \frac{e \cdot (K_S + \alpha_s, c_0)}{e(b_s, c_1)}$$

Observe that the above KAC construction is independent of the manner in which a data owner chooses to organize her data classes. Any KAC construction can support hierarchical data structures, since a data owner can create an aggregate.

Correctness

$$\widehat{M} = c_2 \cdot \frac{e(K_S + \sum_{j\epsilon s, j\neq i} P_{n+1-j+i, c_0})}{e(\sum_{j\epsilon S} P_{n+1-j, c_1})}$$
$$= c_2 \cdot \frac{e(\sum_{j\epsilon s, j\neq i} \gamma P_{n+1-j} + \sum_{j\epsilon s, j\neq i} P_{n+1-j+i, tQ})}{e(\sum_{j\epsilon S} P_{n+1-j, t(PK_2+Q_i))}}$$
$$= c_2 \frac{e(\sum_{j\epsilon S} P_{n+1-j+i, tQ})}{e(P_{n+1, tQ}) e(\sum_{j\epsilon S} P_{n+1-j, tQ_i})}$$
$$= M \cdot \frac{e(P_n, tQ_1)}{e(P_{n+1}, tQ)}$$
$$= M.$$

Extended KAC with aggregate key broadcast

The KAC constructions require the aggregate keys to be transmitted to data users via secure channels. However, in a real world data sharing setup with m data users and m' data owners, such a solution requires the existence of $O(mm')$ secure channels, which is extremely costly. In addition, the dynamically increasing nature of the network implies that the requirement for secure channels increases in a multiplicative mode with every new data owner/user joining the network. This makes the basic KAC scheme difficult for large scale deployment regardless of its cryptographically secure aggregate key generation properties. In this section, we develop a novel mechanism for public key based aggregate key distribution that reduces the secure channel requirement to $O(m + m')$ from (mm') . We use broadcast encryption, which is a well known technique in public key cryptography, to efficiently distribute the aggregate keys among multiple users in a secure fashion. Our extended KAC construction combines the basic KAC instance with the public key based broadcast encryption system to construct a fully public key based online data sharing scheme.

The Framework for Extended KAC

The framework for extended KAC with aggregate key broadcast is presented below:

1) Setup $(1^{\nearrow}, n, m)$: Set up () operation takes the number of data classes n, the number of users m and the security parameter as the input. Output the public parameter param.

2) Owner Key Gen(): Outputs the public key PK, the master-secret key msk and the broadcast secret key bsk for a data owner registering in the system.

3) Owner Encrypt (param, PK, i, M): Takes as input a data class i ∈{1,..., n} and the plaintext data M. Outputs a partly encrypted cipher text C0. Note that C0 is not the final cipher text and is not exposed to the outside world. It is sent to the system administrator via a secure channel for additional medication as described next. Note here that any instantiation of this scheme must ensure that the partial ciphertextC0 is protected using appropriate randomizations so as to reveal nothing about the underlying plaintext data M during transmission to the system administrator.

4) System Encrypt (C0, msk, bsk): Takes as input the partially encrypted cipher text C0, the master secret key msk and the broadcast secret key bsk. Outputs the final cipher text C which is made available on the cloud. This step is carried out by the system administrator, who is a trusted third party.

5) User Key Gen (param, msk, $\hat{\imath}$): Takes as input the data user id $\hat{\imath} \in \{1,...,m\}$and outputs the corresponding secret key $d_{\hat{\imath}}$.

6) Extract (param, msk, S): Takes as input the master secret key msk and a subset of data classes $S \subseteq \{1,...,n\}$. Computes the aggregate key KS for all encrypted messages belonging to these subset of classes, and passes it as input to the Broadcast algorithm for generating the broadcast aggregate key.

7) Broadcast (param, KS, $\hat{S}$, PK, bsk): Takes as input the aggregate key KS and the target subset of users $\hat{s} \subseteq \{1,...,m\}$. Outputs a single broadcast aggregate key $K_{(S,\hat{S})}$that allows any user $\hat{\imath} \in \hat{s}$to decrypt all encrypted data/message classified into any class $i \in S$.

8) Decrypt (param, C,$K_{(S,\hat{S})}$, i,$\hat{\imath}$, $d_{\hat{\imath}}$, S, $\hat{s}$): The decryption algorithm now takes, besides the cipher text C and the corresponding data class $i \in S$, a valid user id $\hat{\imath} \in \hat{s}$. It also takes as input the broadcast aggregate key $K_{(S,\hat{S})}$ and the secret key $d_{\hat{\imath}}$. The algorithm outputs the decrypted message.

We note that the secure channel requirement is one per data owner (in Owner Encrypt) and one per data user (in User Key Gen). Thus the overall secure channel requirement is$O(m + m')$. Observe that the main challenge to be tackled in realizing this scheme is combining the original aggregate key KS with the broadcast secret to obtain the final broadcast aggregate key K

CONCLUSIONS AND DISCUSSIONS

In this paper, we have proposed an capably implementable version of the basic key-aggregate cryptosystem in with low overhead cipher texts and aggregate keys, using asymmetric bilinear pairings. Our creation serves as an efficient solution for several data sharing applications on the cloud, including collaborative data sharing, product license distribution and medical data sharing. We have proved our construction to be fully collusion resistant and semantically secure against a non-adaptive adversary under appropriate security assumptions. We have then demonstrated how this construction may be modified to achieve CCA-secure construction, which is, to the best of our knowledge, the first CCA secure KAC construction in the cryptographic literature. We have further demonstrated how the basic KAC framework may be efficiently extended and generalized for securely broadcasting the aggregate key among multiple data users in a real-life data sharing environment. This provides a crucial pathway in designing a scalable fully public key based online data sharing scheme for large-scale deployment on the cloud. We have presented simulation results to validate the space and time complexity requirements for our scheme. The results establish that KAC with aggregate key broadcast outperforms other existing secure data sharing schemes in terms of performance and scalability.

REFERENCES:

[1] F. Gens, "IT cloud services user survey, pt. 3: What users want from cloud services providers," Aug. 2008, http://blogs.idc. com/ie/?p=213

[2] S. S. Chow, Y.-J. He, L. C. Hui, and S. M. Yiu, "Spice-simple privacy-preserving identity-management for cloud environment," in Applied Cryptography and Network Security. Berlin, Germany: Springer, 2012, pp. 526–543.

[3] C. Wang, S. S.-M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacypreserving public auditing for secure cloud storage," Cryptology ePrint Archive, Report 2009/579, 2009. [Online]. Available: http:// eprint.iacr.org/

[4] S. S. 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. Berlin, Germany: Springer, 2012, pp. 442–464.

[5] E. C. Shallman, "Up in the air: Clarifying cloud storage protections," Intell. Property Law Bulletin, vol. 19, 2014, Art. no. 49.

[6] C.-K. Chu, S. S. Chow, W.-G. Tzeng, J. Zhou, and R. H. Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage," IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 2, pp. 468– 477, Feb. 2014.

[7] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in Advances in Cryptology. Berlin, Germany: Springer, 2005, pp. 258–275.

[8] S. G. Akl and P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," ACM Trans. Comput. Syst., vol. 1, no. 3, pp. 239–248, 1983.

[9] G. C. Chick and S. E. Tavares, "Flexible access control with master keys," in Advances in Cryptology. Berlin, Germany: Springer, 1990, pp. 316–322.

[10] W.-G. Tzeng, "A time-bound cryptographic key assignment scheme for access control in a hierarchy," IEEE Trans. Knowl. Data Eng., vol. 14, no. 1, pp. 182–188, Jan./Feb. 2002.

[11] G. Ateniese, A. de 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.

- ***Mr.Mohemmed Yousuf , Assistant Professor, MVJ College of Engineering, Bangalore, mohemmed.yousuf@mvjce.edu.in***
- ***Mr.Godhandaraman , Assistant Professor, MVJ College of Engineering, Bangalore ,gdraman84@gmail.com***
- ***Mr.Karthick Myilvahanan, Assistant Professor, MVJ College of Engineering, Bangalore, karthik.mj@mvjce.edu.in***