International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

A Flexible Crypto Mechanism for Secure Sharing In Cloud Environment

Majeti Abhinav[1], N. Bhagya Lakshmi[2]

[1]Computer Science Engineering, Guntur Engineering College, Guntur, India

[2]Computer Science Engineering, Guntur Engineering College, Guntur, India

Abstract: ***Cloud Computing is a big buzz word in today's technology uplift, storage of data in distributed network has been re-defined in the cloud environment and sharing the data stored is an important aspect of cloud storage systems which has raised many eyebrows with its security policies. In this paper, we present a flexible modeled crypto mechanism that produce constant-size cipher texts such that efficient delegation of decryption rights for any set of cipher texts 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 cipher texts 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. We provide formal security analysis of our schemes in the standard model. We also describe other application of our schemes. In particular, our schemes give the first public-key patient-controlled encryption for flexible hierarchy, which was yet to be known.***

Keywords: Cloud storage, data sharing, key-aggregate encryption, patient-controlled encryption

1. Introduction

CLOUD storage is gaining popularity recently. In enterprise settings, we see the rise in demand for data outsourcing, which assists in the strategic management of corporate data. It is also used as a core technology behind many online services for personal applications. Now days, it is easy to apply for free accounts for email, photo album, file sharing and/or remote access, with storage size more than 25 GB (or a few dollars for more than 1 TB). Together with the current wireless technology, users can access almost all of their files and emails by a mobile phone in any corner of the world.

Considering data privacy, a traditional way to ensure it is to rely on the server to enforce the access control after authentication (e.g., [1]), which means any un expected privilege escalation will expose all data. In a shared-tenancy cloud computing environment, things become even worse. Data from different clients can be hosted on separate virtual machines (VMs) but reside on a single physical machine. Data in a target VM could be stolen by instantiating another VM co resident with the target one [2]. Regarding avail-ability of files, there are a series of cryptographic schemes which go as far as allowing a third-party auditor to check.

In this paper, we attempt to facilitate, and ensure, the integrity of elections by introducing our M-Vote system, which is capable of performing tasks that can reduce the risk inherent in the voting process, such as the addition, deletion, and alteration of votes. Requirements are gathered and composed based on a list of questions asked of a specific category of people who have participated in elections and election officials, and on other electronic voting systems, in order to acquire precise ideas regarding their experience and to avoid their mistakes. This system is designed to accept only one vote from the voter, check on his/her eligibility, and prevent any attempt to manipulate the vote. It is possible to perform all these operations with nothing more than a mobile phone and an Internet connection.

The availability of files on behalf of the data owner without leaking anything about the data [3], or without compromising the data owner's anonymity [4]. Likewise, cloud users probably will not hold the strong belief that the cloud server is doing a good job in terms of confidentiality. A crypto-graphic solution, for example, [5], 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. These users are motivated to encrypt their data with their own keys before uploading them to the server.

Data sharing is an important functionality in cloud storage. For example, bloggers can let their friends view a subset of their private pictures; an enterprise may grant her employees access to a portion of sensitive data. The challenging problem is how to effectively share encrypted data. Of course users can download the encrypted data from the storage, decrypt them, then send them to others for sharing, 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. Below we will take Dropbox1.as an example for illustration.

1.1 Our Contributions

In modern cryptography, a fundamental problem we often study is about leveraging the secrecy of a small piece of knowledge into the ability to perform cryptographic functions (e.g., encryption, authentication) multiple times. In this paper, we study how to make a decryption key more powerful in the sense that it allows decryption of multiple cipher texts, without increasing its size.

We solve this problem by introducing a special type of public-key encryption which we call key-aggregate crypto-system (KAC). In KAC, users encrypt a message not only under a public-key, but also under an identifier of Cipher text called class. That means the cipher texts 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 cipher text classes.

With our solution, Alice can simply send Bob a single aggregate key via a secure e-mail. Bob can download the encrypted photos from Alice's Drop box space and then use this aggregate key to decrypt these encrypted photos. The scenario is depicted in Figure 1.

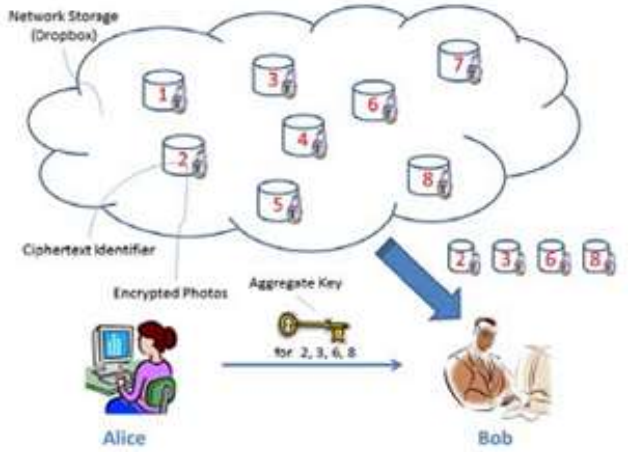

Figure 1: Alice shares files with identifiers 2, 3, 6 and 8 with Bob by sending him a single aggregate key

The sizes of cipher text, public-key, master-secret key and aggregate key in our KAC schemes are all of constant size. The public system parameter has size linear in the number of cipher text 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.

2. Key-Aggregate Encryption

We first give the framework and definition for key-aggregate encryption. Then we describe how to use KAC in a scenario of its application in cloud storage.

It is obvious that we are not proposing an algorithm to compress the decryption key. On one hand, cryptographic keys come from a high-entropy source and are hardly compressible. On the other hand, decryption keys for all possible combinations of cipher text classes are all in constant-size—information theoretically speaking such compression scheme cannot exist.

2.1 Framework

A key-aggregate encryption scheme consists of five polynomial-time algorithms as follows.

The data owner establishes the public system parameter via Setup and generates a public/master-secret3key pair via Key Gen. Messages can be encrypted via Encrypt by anyone who also decides what cipher text class is associated with the plaintext message to be encrypted. The data owner can use the master-secret to generate an aggregate decryption key for a set of cipher text classes via Extract. The generated keys can be passed to delegates securely (via secure e-mails or secure devices) finally; any user with an aggregate key can decrypt any cipher text provided that the cipher text's class is contained in the aggregate key via Decrypt4.

- $\text{Setup}(1^{\lambda}, n)$Executed by the data owner to setup an account on an un trusted server. On input a security level parameter $1^{\lambda}$and the number of cipher text classes n (i.e., class index should be an integer bounded by 1 and n), it outputs the public system parameter param, which is omitted from the input of the other algorithms for brevity.
- Key Gen: executed by the data owner to randomly generate a public/master-secret key pair (pk, msk).
- $\text{Encrypt}(\text{pK}, \text{i}, \text{m})$: executed by anyone who wants to encrypt data. On input a public-key pk, an index I denoting the cipher text class, and a message m,it outputs a cipher text C.
- $\text{Encrypt}(\text{msK}, \text{S})$Executed by the data owner for delegating the decrypting power for a certain set of cipher text classes to a delegate. On input the master-secret key msk and a set S of indices corresponding to different classes, it outputs the aggregate key for set S denoted by KS.
- $\text{Decrypt}(\text{K}_s, \text{S}, \text{i}, \text{C})$ executed by a delegate who received an aggregate key $\text{K}_s$generated by Extract. On input K_s, the set S, an index i denoting the cipher text class the cipher text C belongs to, and C ,it outputs the decrypted result m if i∈S.

There are two functional requirements:

Correctness. For any integers λ and n, any set $\text{S} \subseteq \{1; \ldots; \text{n}\}$ any index $\text{i} \in \text{S}$ and any message m,

$$\Pr\big[\text{Decrypt}(\text{K}_s, \text{S}, \text{i}, \mathfrak{C}) = \text{m}; \text{param} \leftarrow \text{Setup}(1^{\lambda}, \text{n}), (\text{PK}, \text{msK}) \leftarrow \text{Key Gen}(), \mathfrak{C} \leftarrow \text{Encrpt}(\text{pK}, \text{i}, \text{m})\text{K}_s \leftarrow \text{Extract}(\text{msK}, \text{S})\big] == 1$$

Compactness. For any integers λ, n, any set S, any index $\text{i} \in \text{S}$ and any message m; $\text{param} \leftarrow \text{Setup}(1^{\lambda}, \text{n})$ $(\text{PK}, \text{msK}) \leftarrow \text{Key Gen}()\text{K}_s \leftarrow \text{Extract}(\text{msK}, \text{S})$ $\text{C} \leftarrow \text{Encrpt}(\text{pK}, \text{i}, \text{m})$; $|\text{K}_s|$ and $|\text{C}|$ only depend on the security parameter λ but independent of the number of classes n.

We call this as master-secret key to avoid confusion with the delegated key we will explain later.

For simplicity, we omit the inclusion of a decryption algorithm for the original data owner using the master-secret key. In our specific constructions, we will show how the knowledge of the master-secret key allows a faster decryption than using Extract followed by Decrypt.

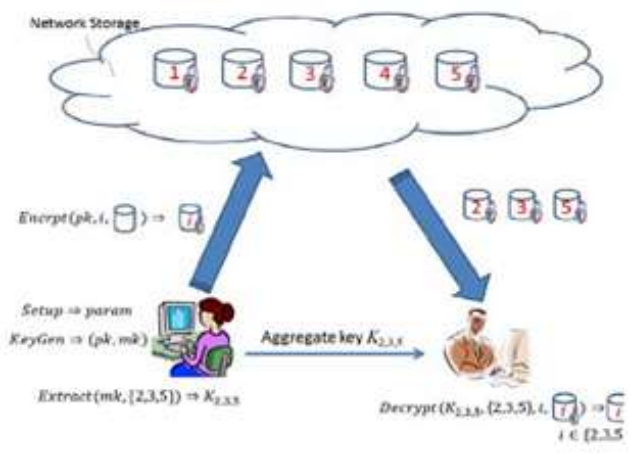

Figure 2: Using KAC for data sharing in Cloud Storage

2.2 Sharing Encrypted Data

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

Here, we describe the main idea of data sharing in cloud storage using KAC, illustrated in Fig. 2. Suppose Alice wants to share her data m_1; m_2;... m_v on the server. She first performs $Setup(1^\lambda, n)$: 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 (including Alice herself) can then encrypt each mi by $C_i = Encrpt(pK, i, m)$:. 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 K_s for Bob by performing $Extract(msK, S)$:. Since K_s is just a constant-size key, it is easy to be sent to Bob via a secure e-mail.

After obtaining the aggregate key, Bob can download the data he is authorized to access. That is, for each i∈S, Bob downloads C_i (and some needed values in param) from the server. With the aggregate key KS, Bob can decrypt each C_i by Decrypt (K_s,S_i, C_i) for each i∈S.

3. Related Work

This section we compare our basic KAC scheme with other possible solutions on sharing in secure cloud storage.

3.1 Cryptographic Keys for a Predefined Hierarchy

We start by discussing the most relevant study in the literature of cryptography/security. Cryptographic key assignment schemes (e.g., [11], [12], [13], [14]) aim to minimize the expense in storing and managing secret keys for general cryptographic use. Utilizing a tree structure, a key for a given branch can be used to derive the keys of its descendant nodes (but not the other way round). Just granting the parent key implicitly grants all the keys of its descendant nodes. Sandhu [15] proposed a method to generate a tree hierarchy of symmetric-keys by using repeated evaluations of pseudorandom function/block-cipher on a fixed secret. The concept can be generalized from a tree to a graph. More advanced cryptographic key assignment schemes support access policy that can be modelled by an acyclic graph or a cyclic graph [16], [17], [7]. Most of these schemes produce keys for symmetric-key crypto systems, even though the key derivations may require modular arithmetic as used in public-key cryptosystems, which are generally more expensive than "symmetric-key operations" such as pseudorandom function.

In Fig. 3a, if Alice wants to share all the files in the "personal" category, she only needs to grant the key for the node "personal," which automatically grants the delegate the keys of all the descendant nodes ("photo," "music"). This is the ideal case, where most classes to be shared belong to the same branch and thus a parent key of them is sufficient.

However, it is still difficult for general cases. As shown in Fig. 3b, if Alice shares her demo music at work("work"→"casual"→"demo"and"work"→""confidential" → "demo") with a colleague who also has the rights to see some of her personal data, what she can do is to give more keys, which leads to an increase in the total key size. One can see that this approach is not flexible when the classifications are more complex and she wants to share different sets of files to different people. For this delegate in our example, the number of granted secret keys becomes the same as the number of classes.

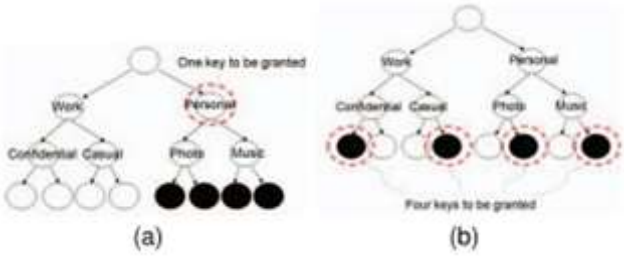

Figure 3: Compact Key is not always possible for a fixed hierarchy

In general, hierarchical approaches can solve the problem partially if one intends to share all files under a certain branch in the hierarchy. On average, the number of keys increases with the number of branches. It is unlikely to come up with a hierarchy that can save the number of total keys to be granted for all individuals (which can access a different set of leaf-nodes) simultaneously.

3.2 Compact Key in Symmetric-Key Encryption

Motivated by the same problem of supporting flexible hierarchy in decryption power delegation (but in sym-metric-key setting), Benaloh et al. [8] presented an encryp-tion scheme which is originally proposed for concisely transmitting large number of keys in broadcast scenario [18]. 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 as follows: A composite modulus=p.q is chosen where p and q are two large random primes. A

master-secret key Y is chosen at random from Z_N^*. Each class is associated with a distinct prime ei All these prime numbers can be put in the public system parameter5.A constant-size key for set S' can be generated (with the knowledge of $\emptyset(N)$)as

$$K_{S'} = Y^{1/\prod_{j\in S'(e_j)}} mod\ N.$$

For those who have been delegated the access rights for S where S'C S, $K_{s'}$can be computed by

$$K_S^{\prod j\in s\setminus S'^{(e_j)}}$$

As a concrete example, a key for classes represented bye1;e2;e3 can be generated $Y^{1/e_1.e_2.e_3}$, from which each ofY^{1/e_1} , Y^{1/e_2} , Y^{1/e_3}can easily be derived (while providing no information about keys for any other class, say, e4). This approach achieves similar properties and performances as our schemes. However, it is designed for the symmetric-key setting instead. The encrypt or needs to get the corresponding secret keys to encrypt data, which is not suitable for many applications. Since their 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

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. Encryption, for example, [19]. However, sharing of decryption power is not a concern in these schemes.

3.3 Compact Key in Identity-Based Encryption (IBE)

IBE (e.g., [20], [21], [22]) is a type of 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). There is a trusted party called private key generator in IBE which holds a master-secret key and issues a secret key to each user with respect to the user identity. The encrypt or can take the public parameter and a user identity to encrypt a message. The recipient can decrypt this cipher text by his secret key.

Guo et al. [23], [9] tried to build IBE with key aggregation. One of their schemes [23] assumes random oracles but another [9] does not. 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. Most importantly, their key-aggregation [23], [9] comes at the expense of O(n) sizes for both cipher texts and the public parameter, where n is the number of secret keys which can be aggregated into a constant size one. This greatly increases the costs of storing and transmitting cipher texts, which is impractical in many situations such as shared cloud storage. As we mentioned, our schemes feature constant cipher text size, and their security holds in the standard model.
In fuzzy IBE [21], one single compact secret key can decrypt cipher texts encrypted under many identities which hare 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.

4. Concrete Constructions of KAC

Let G and G_T be two cyclic groups of prime order p and

^e : G×G → G_T be a map with the following properties:

- $Bilinear : \forall_{g_1,g_2}\epsilon G, a, b\epsilon Z, e\hat{}(g_1^a, g_2^b) = e\hat{}(g_1, g_2)^{ab}$.
- Nondegenerate: for some g ∈ G, $e\hat{}(g,g) \neq 1$.G is a bilinear group if all the operations involved above are efficiently computable. Many classes of elliptic curves feature bilinear groups.

4.1 A Basic Construction

The design of our basic scheme is inspired from the collusion-resistant broadcast encryption scheme proposed by Boneh et al. [24]. Although their scheme supports constant-size secret keys, every key only has the power for decrypting cipher texts associated to a particular index. We, thus, need to devise a new Extract algorithm and the corresponding Decrypt algorithm

- $Setup(1^\lambda, n)$: Randomly pick a bilinear group G of prime order p where $2^\lambda \leq p \leq 2^{\lambda+1}$, a generator g ∈G and $\alpha\epsilon R\ Z_p$. Compute $g_{i=g^{\propto i}} \in G$for i = 1. . . ;n; n +2; . . . ; 2n. Output the system parameter as param= $\langle g; g_1; \ldots; g_n; g_{n+2}; \ldots; g_{2n}\rangle$ ($\propto$ can be safely deleted after Setup). Note that each cipher text class is represented by an index in the integer set {1; 2; . . . ; n}, where n is the maximum number of cipher text classes.
- Key Gen(): Pick $\gamma\epsilon R$ Zp, output the public and Master secret key pair: ($pk = v = g^\gamma$; msk= γ).
- $Encrpt(pK, i, m)$: For a message m∈ $G_T$and an index i ∈ {1; 2; . . . ; n}, randomly pick t ∈R Zp and compute the cipher text as $C = \langle g^t, (vgi)^t, m.e\hat{}(g_1, g_n)^t\rangle$'
- $Extract(msk = \gamma, S)$: For the set S of indices j's, the aggregate key is computed as $K_S = \prod_{j\in S} g_{n+1-j}^\gamma$Since S does not include 0, $g_{n+1-j} = g^{\alpha^{n+1-j}}$can always be retrieved from param.
- $Decrypt(K_S, S, i, C = \langle c_1, c_2, c_3\rangle)$: If i ¢ S, output ⊥.Otherwise, return Q the message

$$m = c_3 . e\hat{}(K_S.\prod_{j\in S, j\neq i} g_{n+}\ 1-j + i, c_1)/e\hat{}(\prod_{j\in S} g_{n+1-j}, c_2)$$

For the data owner, with the knowledge of γ, the term $e\hat{}(g_1, g_n)^t$ can be easily recovered by $e\hat{}(c_1, g_n)^\gamma = e\hat{}(g^t, g_n)^t = e\hat{}(g_1, g_n)^t$For correctness, we can see that

$$c_3 . e\hat{}(K_S.\prod_{j\in S, j\neq i} g_{n+}\ 1-j+i, c_1)/e\hat{}(\prod_{j\in S} g_{n+1-j}, c_2)$$

$$= c_3 . \frac{e\hat{}(\prod_{j\in S} g_{n_1-j}^\gamma \cdot \prod_{j\in S.j\neq i} g_{n+1-j+i,g^t})}{e\hat{}(\prod_{j\in S} g_{n+1-j}, (vg_i)^t)}$$

$$= c_3 . e\hat{}(\prod_{j\in S, j\neq i} g_{n+1-j+i,g^t})/e\hat{}(\prod_{j\in S} g_{n+1-j}, g_i^t)$$

$$= c_3 . \frac{e\hat{}(\prod_{j\in S} g_{n+1-j+i,g^t,})/e\hat{}(g_{n+1}, g^t)}{e\hat{}(\prod_{j\in S} g_{n+1-j+i}, g^t)}$$

$=m.e\hat{}(g_1, g_n)^t/e\hat{}(g_{n+1}, g^t) = m$

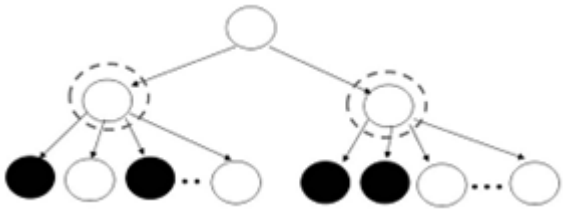

Figure 4: Key Assignment in our approach

4.2 Public-Key Extension

If a user needs to classify his cipher texts into more than n classes, he can register for additional key pairs (pK_2, msK_2);...;(pK_l, msK_l). Each class now is indexed by a two-level index in $\{(i,j)|1 \le i \le l, 1 \le j \le n\}$ and the number of classes is increased by n for each added key.

Since the new public-key can be essentially treated as a new user, one may have the concern that key aggregation across two independent users is not possible. It seems that we face the problem of hierarchical solution as reviewed in Section 1, but indeed, we still achieve shorter key size and gain flexibility as illustrated in Fig. 4. Fig. 4 shows the flexibility of our approach. We achieve "local aggregation, "which means the secret keys under the same branch c an always be aggregated. We use a quaternary tree for the last level just for better illustration of our distinctive feature. Our advantage is still preserved when compared with quaternary trees in hierarchical approach, in which the latter either delegates the decryption power for all four classes (if the key for their parent class is delegated) or the number of keys will be the same as the number of classes. For our approach, at most two aggregate keys are needed in our example. Below we give the details on how encryption and decryption work when the public-key is extended, which is similar to the " "$\sqrt{n}$-approach" [31].

- Setup and KeyGen: Same as the basic construction.
- Extend(pK_l, msK_l): Execute KeyGen () to get$(v_{l+1}, \gamma_{l+1}) \in G \times Z_p$, output the extended public and Master secret keys $pK_{l+1} = (pK_l, v_{l+1}), msK_{l+1} = (msK_l, \gamma_{l+1})$.
- $Encrypt(pK_l, (a,b), m)$: LetpK_l={v_1,...v_l} For an index (a; b); 1 ≤ a ≤ l; 1 ≤ b ≤ n, pick t ∈R Zp, output the cipher text as$C = \langle g^t, (v_a g_b)^t, m.e\hat{}(g_1, g_n)^t \rangle$.
- $Extract(msK_l, S_l)$: Let msK_l= {γ_1,γ_2,... γ_l}. For a setS_l of indices (i; j); 1 ≤ i ≤ l; 1 ≤j ≤ n, get $g_{n+1-j=g^{\alpha^{n+1-j}}}$ from param, output:

$$K_{S'} = (\prod_{(1,j)\in S_1} g^{\gamma_1}_{n+1-j}, \prod_{(2,j)\in S_1} g^{\gamma_2}_{n+1-j}, \dots, \prod_{(l,j)\in S_1} g^{\gamma_l}_{n+1-j})$$

- $Decrypt(K_S, S_l, (a,b), C)$:: If (a; b) ¢ S_l, output ⊥.Otherwise, let K_{Sl} = {d1; . . . ; dl} and C = $\langle c_1, c_2, c_3 \rangle$ Output the message:

We can also prove the semantic security of this extended scheme. The proof is very similar to that for the basic scheme and therefore is omitted. The public-key of our CCA construction to be presented below can also be extended using the same Extend algorithm.

$$m = \frac{c_3 . e\hat{}(d_a . \prod_{(a.j)\in S_{1,j\neq b}} g_{n+1-j+b}, c_1}{e\hat{}(\prod_{(a,j)\in S_1} g_{n+1-j}, c_2)}$$

Just like the basic construction, the decryption can be done more efficiently with the knowledge of γ'_i's.

Correctness is not much more difficult to see

$c_3.e\hat{}\left(d_a.\prod_{(a,j)\in S_{1,j\neq b}} g_{n+1-j+b}, c_1\right)/e\hat{}(\prod_{(a,j)\in S_l} g_{n+1-j}, c_2)$

$=$

$c_3.e\hat{}(\prod_{(a,j)\in S_l} g^{\gamma\alpha}_{n+1-j} \cdot \prod_{(a,j)\in S_{l,j\neq b}} g_{n+1-j+b}, g^t)/$

$e\hat{}(\prod_{(a,j)\in S_l} g_{n+1-j}, (v_a g_b)^t)$

$= c_3.e\hat{}(\prod_{(a,j)\in S_{l,j\neq b}} g_{n+1-j+b}, g^t)/e\hat{}(\prod_{(a,j)\in S_l} g_{n+1-j}, (g_b)^t)$

$= m.e\hat{}(g_1, g_n)^t / e\hat{}(g_{n+1}, g^t) = m$

We can also prove the semantic security of this extended scheme. The proof is very similar to that for the basic scheme and therefore is omitted. The public-key of our CCA construction to be presented below can also be extended using the same Extend algorithm.

5. Conclusion and Future Work

How to protect users' data privacy is a central question of cloud storage. With more mathematical tools, crypto graphic schemes are getting more versatile and often involve multiple keys for a single application. In this paper, we consider how to "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 delegate 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.

A limitation in our work is the predefined bound of the number of maximum cipher text classes. In cloud storage, the number of cipher texts usually grows rapidly. So we have to reserve enough cipher text classes for the future extension. Otherwise, we need to expand the public-key as we described in Section 4.2.

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. On the other hand, when one carries the delegated keys around in a mobile device without using special trusted hardware, the key is prompt to leakage, designing a leakage-resilient crypto system [22] yet allows efficient and flexible key delegation is also an interesting direction.

References

[1] S.S.M. Chow, Y.J. He, L.C.K. Hui, and S.-M. Yiu, "SPICE - SimplePrivacy-Preserving Identity-Management for Cloud Environ-ment,"Proc. 10th Int'l Conf. Applied Cryptography and Network Security (ACNS),vol. 7341, pp. 526-543, 2012.

[2] L. Hardesty, Secure Computers Aren't so Secure.MIT press http://www.physorg.com/news176107396.html, 2009.

[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, Feb. 2013.

[4] 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.
[5] S.S.M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R.H. Deng, "Dynamic Secure Cloud Storage with Provenance,"Cryptographyand Security,pp. 442-464, Springer, 2012.
[6] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps,"Proc. 22ndInt'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '03),pp. 416-432, 2003.
[7] M.J. Atallah, M. Blanton, N. Fazio, and K.B. Frikken, "Dynamicand Efficient Key Management for Access Hierarchies," ACMTrans. Information and System Security,vol. 12, no. 3, pp. 18:1-18:43,2009.
[8] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "PatientControlled Encryption: Ensuring Privacy of Electronic MedicalRecords," Proc. ACM Workshop Cloud Computing Security(CCSW '09),pp. 103-114, 2009.
[9] F. Guo, Y. Mu, Z. Chen, and L. Xu, "Multi-Identity Single-KeyDecryption without Random Oracles,"Proc. Information Security and Cryptology (Inscrypt '07),vol. 4990, pp. 384-398, 2007.
[10] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data,"Proc. 13th ACM Conf. Computer and Comm. Security (CCS '06), pp. 89-98, 2006.
[11] S.G. Akl and P.D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy,"ACM Trans. Computer Systems,vol. 1, no. 3, pp. 239-248, 1983.
[12] G.C. Chick and S.E. Tavares, "Flexible Access Control with MasterKeys,"Proc. Advances in Cryptology (CRYPTO '89), vol. 435,pp. 316-322, 1989.
[13] W.-G. Tzeng, "A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy," IEEE Trans. Knowledge and Data Eng.,vol. 14, no. 1, pp. 182-188, Jan./Feb. 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 Hierarchyfor 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," Proc. IEEE INFOCOM '04, 2004.
[17] Q. Zhang and Y. Wang, "A Centralized Key Management Scheme for Hierarchical Access Control," Proc. IEEE Global Telecomm. Conf.(GLOBECOM '04), pp. 2067-2071, 2004.
[18] J. Benaloh, "Key Compression and Its Application to Digital Fingerprinting," technical report, Microsoft Research, 2009.
[19] B. Alomair and R. Poovendran, "Information Theoretically Secure Encryption with Almost Free Authentication," J. UniversalComputer Science,vol. 15, no. 15, pp. 2937-2956, 2009.
[20] D. Boneh and M.K. Franklin, "Identity-Based Encryption from the Weil Pairing," Proc. Advances in Cryptology (CRYPTO '01), vol. 2139, pp. 213-229, 2001.
[21] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption,"Proc.22nd Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '05), vol. 3494, pp. 457-473, 2005.
[22] S.S.M. Chow, Y. Dodis, Y. Rouselakis, and B. Waters, "Practical Leakage-Resilient Identity-Based Encryption from Simple Assumptions," Proc. ACM Conf. Computer and Comm. Security, pp. 152-161, 2010.
[23] F. Guo, Y. Mu, and Z. Chen, "Identity-Based Encryption: How to Decrypt Multiple Cipher texts Using a Single Decryption Key," Proc. Pairing-Based Cryptography Conf. (Pairing '07), vol. 4575,pp. 392-406, 2007.
[24] D. Boneh, C. Gentry, and B. Waters, "Collusion Resistant Broad-cast Encryption with Short Cipher texts and Private Keys,"Proc. Advances in Cryptology Conf. (CRYPTO '05), vol. 3621, pp. 258-275, 2005.

Author Profile

Majeti. Abhinav Obtained the B.Tech degree in Information Technology (IT) from Guntur Engineering College, Guntur. At present he is pursuing the M.Tech in Computer Science and Engineering (CSE) Department at Guntur Engineering College, Guntur.

N. Bhagya Lakshmi obtained the B.Tech Degree from Prathyusha Engineering College, Anna University and M. Tech (CSE) from Nimra College of Engineering, JNTU Kakinada. She has 5 years of teaching experience and working in Computer Science and Engineering (CSE) Department at Guntur Engineering College, Guntur.