Scalable and Secured Data Storage in Cloud Environment using Leakage Resilient Key Cryptosystem

S. Gomathi
Department of Computer Science and Engineering,
J.K.K. Nattraja College of Engineering and Technology,
Kumarapalayam, Tamilnadu, India

P. Ramya
Department of Computer Science and Engineering,
J.K.K. Nattraja College of Engineering and Technology,
Kumarapalayam, Tamilnadu, India

Abstract- **Data sharing is an important functionality in cloud storage. In this previous work, it shows how to securely, efficiently, and flexibly share data with others in cloud storage. The existing work presents the Key-Aggregate Cryptosystem used for conveniently sent to others or be stored in a smart card with very limited secure storage. A limitation of existing work is the predefined bound of the number of maximum cipher text classes and key is prompt to leakage. The proposed work mainly concentrates on above two problems. The first work is to dynamically reserve number of maximum cipher text classes in cloud storage. It describes new cryptosystem which produce cipher text of dynamic size such that decryption rights can be assigned on them. Additionally designing a Leakage-Resilient Identity-Based Encryption (LR-IBE) allows efficient and flexible key delegation. LR-IBE system, based on Boneh-Boyen IBE, is only selectively secure under the simple Decisional Bilinear Diffie-Hellman assumption (DBDH), and serves as a stepping stone to the second fully secure construction. The proposed scheme gives an efficient key encryption scheme for efficient and flexible.**

Keywords—cloud storage, data sharing, key-aggregate encryption, patient-controlled encryption

I. INTRODUCTION

Cloud storage is gaining popularity recently. In enterprise settings, it is seen that 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. Nowadays, 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, which means any unexpected 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 coincident with the target one. 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, for example, 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 it will take Dropbox1 as an example for illustration.

Assume that Alice puts all her private photos on Dropbox, and she does not want to expose her photos to everyone. Due to various data leakage possibility Alice cannot feel relieved by just relying on the privacy protection mechanisms provided by Dropbox, so she encrypts all the photos using her own keys before uploading. One day, Alice's friend, Bob, asks her to share the photos taken over all these years which Bob appeared in. Alice can then use the share function of Dropbox, but the problem now is how to delegate the decryption rights for these photos to Bob. A possible option Alice can choose is to securely send Bob the secret keys involved. Naturally, there are two extreme ways for her under the traditional encryption paradigm:

- Alice encrypts all files with a single encryption key and gives Bob the corresponding secret key directly.
- Alice encrypts files with distinct keys and sends Bob the corresponding secret keys.

Obviously, the first method is inadequate since all unchosen data may be also leaked to Bob. For the second method, there are practical concerns on efficiency. The number of such keys is as many as the number of the shared photos, say, a thousand. Transferring these secret keys inherently requires a secure channel, and storing these keys requires rather expensive secure storage.

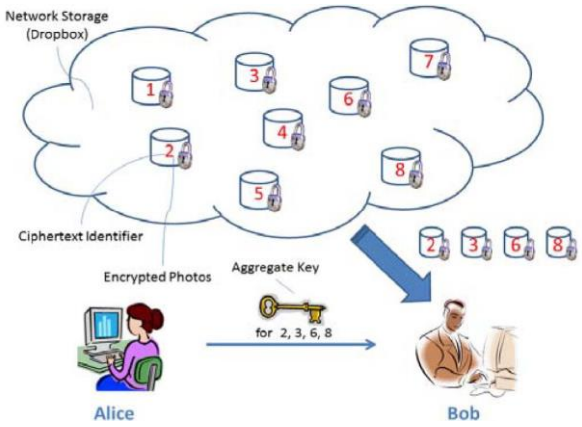

Fig. 1. Alice shares files with identifiers 2, 3, 6, and 8 with Bob by sending him a single aggregate key.

The costs and complexities involved generally increase with the number of the decryption keys to be shared. In short, it is very heavy and costly to do that. Encryption keys also come with two flavors—symmetric key or asymmetric (public) key. Using symmetric encryption, when Alice wants the data to be originated from a third party, she has to give the encryptor her secret key; obviously, this is not always desirable. By contrast, the encryption key and decryption key are different in publickey encryption. The use of public-key encryption gives more flexibility for our applications. For example, in enterprise settings, every employee can upload encrypted data on the cloud storage server without the knowledge of the company's master-secret key.

Therefore, the best solution for the above problem is that Alice encrypts files with distinct public-keys, but only sends Bob a single (constant-size) decryption key. Since the decryption key should be sent via a secure channel and kept secret, small key size is always desirable. For example, we cannot expect large storage for decryption keys in the resource-constraint devices like smart phones, smart cards, or wireless sensor nodes. Especially, these secret keys are usually stored in the tamper-proof memory, which is relatively expensive.

II. LITERATURE SURVEY

A. *Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage*

Cloud storage is gaining popularity recently. Data sharing is an important functionality in cloud storage. The challenging problem is how to effectively share encrypted data. The paper, shows how to make a decryption key more powerful in the sense that it allows decryption of multiple ciphertexts, without increasing its size. A key-aggregate encryption scheme consists of five polynomial-time algorithms as follows. 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 Decrypt

There may be possible of leakage in the network environment where the key will be transmitted in the unknown media. The constant number cipher texts lead to a limitation of extending the data size in the real time

B. *Privacy-Preserving Public Auditing for Secure Cloud Storage*

The system is among the first few ones to support privacy-preserving public auditing in cloud computing, with a focus on data storage. As the individual auditing of these growing tasks can be tedious and cumbersome, a natural demand is then how to enable the TPA to efficiently perform multiple auditing tasks in a batch manner, i.e., simultaneously. This scheme enables an external auditor to audit user's cloud data without learning the data content. To the best of the knowledge, this scheme is the first to support scalable and efficient privacy-preserving public storage auditing in cloud. To fully ensure the data integrity and save the cloud users' computation resources as well as online burden, it is of critical importance to enable public auditing service for cloud data storage, so that users may resort to an independent Third-Party Auditor (TPA) to audit the outsourced data when needed. In a word, enabling public auditing services will play an important role for this nascent cloud economy to become fully established, where users will need ways to assess risk and gain trust in the cloud.

One disadvantage of the Merkle Signature Scheme is the still limited number of signatures. Both time and space requirements of hash traversal technique is maximum. It is inefficient algorithm that generates a sequence of leaves along with their associated authentication paths.

C. *Patient Controlled Encryption : Ensuring Privacy of Electronic Medical Records*

Encryption schemes with strong security properties will guarantee that the patient's privacy is protected. In particular, they would like to employ encryption, yet support such desirable functions as allowing users to share partial access rights with others and to perform various searches over their records. In what follows, they consider encryption schemes

that enable patients to delegate partial decryption rights, and that allow patients (and their delegates) to search over their health data. They shall propose a design, refered to as Patient Controlled Encryption (PCE) as a solution to secure and private storage of patients' medical records. PCE allows the patient to selectively share records among doctors and healthcare providers. The patient is required to store a root secret key, from which a tree of subkeys is derived. The patient can selectively distribute subkeys for decryption of various portions of the record and also generate and distribute trapdoors for selectively searching portions of the record. Their design prevents unauthorized access to patients' medical data by data storage providers, healthcare providers, pharmaceutical companies and insurance companies.

The hierarchy is fixed in that there is only one way in which They can partition the record. The medical records category might be further broken down into subcategories for basic medical information.

III. PROBLEM DEFINITION

In cloud if data owner want to share his data or image files to user to securely send is possible in two ways: 1. Data owner encrypts all files with a single encryption key and gives user the corresponding secret key directly, 2. Data owner encrypts files with distinct keys and sends user the corresponding secret keys. Obviously, the first method is inadequate since all un-chosen data may be also leaked to user For the second method, there are practical concerns on efficiency. The number of such keys is as many as the number of the shared files, say, a thousand. Transferring these secret keys inherently requires a secure channel, and storing these keys requires rather expensive secure storage.

The costs and complexities involved generally increase with the number of the decryption keys to be shared. In short, it is very heavy and costly to do that. The best solution for the above problem is required for sharing data in cloud computing.When using the individual cipher text class id's for the individual blocks, the transferring of corresponding access control key to the user which will enable the decryption access to the users will be more difficult. It needs to be concentrated more in order to evaluate the reduce the communication burden when transferring all the keys to the corresponding users individually. This problem can be overcome by introducing the aggregated key concept in which, keys will be send in the aggregated manner.

IV. EXISTING SYSTEM

Encryption keys also come with two flavours — symmetric key or asymmetric (public) key. Using symmetric encryption, when Alice wants the data to be originated from a third party, she has to give the encryptor her secret key; obviously, this is not always desirable. By contrast, the encryption key and decryption key are different in public key encryption. The use of public-key encryption gives more flexibility for the applications. Introducing a special type of public-key encryption which it call Key-Aggregate Cryptosystem (KAC). In KAC, users encrypt a message not only under a public-key, but also under an identifier of ciphertext called class. That means the ciphertexts are further categorized into different classes.

The key owner holds a master-secret called master-secret key, which can be used to extract secret keys for different classes. More importantly, the extracted key have can be an aggregate key which is as compact as a secret key for a single class, but aggregates the power of many such keys, i.e., the decryption power for any subset of ciphertext classes. The sizes of ciphertext, public-key, master-secret key, and aggregate key in KAC schemes are all of constant size. The public system parameter has size linear in the number of 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.

A. Disadvantages

In the existing system the disadvantage is the predefined bound of the number of maximum cipher text classes. When one carries the delegated keys around in a mobile device without using special trusted hardware, the key is prompt to leakage.

V. PROPOSED SYSTEM

In proposed system it improve the Key-Aggregate Cryptosystem by determining the number of cipher text classes dynamically and applying ID based cryptosystem under Decisional Bilinear Diffie-Hellman assumption to protect leak information in cryptosystem which is called leak resilient cryptosystem. In the first work the classes of cipher text is decide based on the feedback of the current cipher delivered. Here a well suited software implementations is carried out to produce sequences of large period; thee repetitions will takes place after a large number of times and it should also have good statistical properties. It will take up this issue in subsequent class on pseudo randomness. So the proposed system number of classes is determined based on the above statistical properties.

In the second proposal it use the hash proof system that build a Leakage-Resilient IBE (LR-IBE) .In proposed system it use identity based encryption with Decisional Bilinear Diffie-Hellman assumption (DBDH). In this method for each identity id, there are many valid secret keys skid and also two kinds of ciphertexts: valid and invalid. The leakage resilient under the decisional bilinear Diffie-Hellman assumption (DBDH). The idea is to add another degree of randomness to the identity-based secret keys, called the "tag" t, coupled with some master secret key terms. This is done in a way that the secret-key holder can now only re-randomize the key along the original degree of freedom (which is needed for the original proof), but cannot re-randomize the key along the new "tag-dimension" anymore. This will let us define invalid ciphertexts which decrypt to random values when the tag t is random, and yet decrypt to the same value when the tag t is kept the same, but the key is re-randomized along the original degree of freedom.

A. *Advantages*

To protect against weak key-leakage attacks. The number of ciphertexts classes reserve dynamically. efficient and flexible for key delegation.

VI. EXPERIMENTS

A. *Create Cloud Setup*

Initially the basic network model for the cloud data storage is developed in this module using java. Three different network entities can be identified as follows: Data Owner: an entity, who has data to be stored in the cloud and relies on the cloud for data storage and computation, can be either enterprise or individual customers. Cloud Server (CS): an entity, which is managed by Cloud Service Provider (CSP) to provide data storage service and has significant storage space and computation resources (it will not differentiate CS and CSP hereafter). Data User: Can access the data from cloud and send/ receive data stored in the CS.

B. *Network Setup*

In the setup module, the public key and secret key parameters of data owners will be created. The public key, secret key pair is created under decisional bilinear diffie hell-man assumption. The Diffie hellman bilinear assumption is used to create the secured public and secret key pair. This is done by selecting two random group generator parameters g1, g2. This group generated is used to create the group of parameter which will have the public and secret key parameters.

Algorithm:

Let $(p, G = \langle g \rangle, G_T, e(.,.)) \leftarrow g(1^{\lambda})$

Set $mpk = (p, G, G_T, e(.,.), g, u, h, e(g,g)^{\beta})$

Where $u, h \leftarrow G$ and $\alpha, \beta \leftarrow Z_p$

$Msk = (g^{\alpha}, g^{\beta})$

1) *Key Generation*

In this module the secret identity key is generated for the decryption purpose. The secret identity key for the individual cipher text classes will be generated by using the corresponding cipher text class id and the master secret key. This cipher text class id generated dynamically by using the feedback of the previously generated class id's in order to support the large text files.

In this work bilinear Diffie hellman assumption is used to generate the public key and secret key. In this work, it "tag" a user secret key with an integer tag t by introducing a factor of $g^{\beta t}$ ($\beta \in Zp$ is a new secret parameter) to the term which already has g^{α} ($\alpha \in Zp$ is an existing secret parameter). Intuitively, for an attacker who only gets one secret key for each identity, deriving a new key of a different tag requires the knowledge of g^{α} or $g^{\beta t}$. To offset the effect of $g^{\beta t}$ in decryption, the ciphertext requires a new component of e(g, g)_z where $z \in Zp$ is its randomness.

Algorithm:

For $id \epsilon Z_p$,

Choose $t, r \leftarrow Z_p$

Generate $sk_{id} = (S_1, S_2, S_3) = (g^{\alpha} g^{-\beta t} (u^{id}h)^r, g^{-r}, t)$

C. *Encryption*

In this module, the encryption of data and the keys are taken place in order to prevent it from the leakage. The encryption is done to provide the better security mechanism. The encryption process consists of two phases. Those are encapsulation and the decapsulation. The encapsulation is done with the original message whereas the decapsulation is done with the duplicate message in order to confuse the hackers from downloading the original content.

1) *Valid Encapsulation*

In module, the encapsulation is done for the original cipher texts. The encapsulation is done for the valid cipher text as follows: Encap (c, k). Here C represents the valid cipher text message which is obtained by encrypting the original message m with the corresponding cipher text class id. K is encapsulation key of the corresponding cipher text.

*Algorithm***:**

Choose $z \leftarrow Z_p$.

Output $C = (C_1, C_2, C_3) = (g^z, (u^{id}h)z, e(g,g)^{\beta z})$

$K = e(g, g)^{\alpha z}$

2) *Invalid Encapsulation*

In the invalid encapsulation duplicate cipher text that is invalid cipher text will be encapsulated with the corresponding encapsulation key k which is used by the valid encapsulation phase.

Algorithm:

Choose $z, z' \leftarrow Z_p$

Subject to the constraint $z \neq z'$.

Output $C = (C_1, C_2, C_3) = (g^z, (u^{id}h)z, e(g,g)^{\beta z'})$

D. *Decryption*

In the decryption phase, the contents will be decrypted for accessing. The decryption can be done only if the users are

having the proper delegation access which was sent by the data owner. If the users don't have the proper decryption permission, then he cannot download the entire content from the net and he will be limited with the data access permission.

1) Decapsulation

In this module, the cipher text will be decrypted with the secret identity key that is generated by the owner and will output the encapsulation key. By using that encapsulation key the message will be decrypted.

Algorithm:

Output e $(C_1, S1)$ e$(C_2, S_2)C_3^{s3}$

E. Performance Evaluation

It conducted a thorough experimental evaluation of the proposed Diffie Hellman leakage resilient based hash encryption with the key aggregate crypto system. The performance evaluation is done by comparing the proposed algorithm with the existing algorithm based on the parameters called data confidentiality, compression ratio, data integrity and the number of granted keys. The experimental tests that were conducted prove the proposed methodology improves in all of the terminologies mentioned above than the existing methods.

VII. CONCLUSION

In cloud computing, the numbers of user are increasing considerably due to the convenience of sharing data's through cloud considerably. While data sharing, the security is the biggest concern which has to be provided in order to gain the users trust. The key used to encrypt the data has to share with the users to enable them to decrypt the content of data's. Here need to 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. The approach is more flexible than hierarchical key assignment which can only save spaces if all key-holders share a similar set of privileges. The experimental results proves that the proposed approach is better than the existing methodology

In all leakage-resilient IBE systems, including the ones presented here in the paper, leakage is allowed from only one secret key per identity. Although, this can be easily achieved by generating the randomness of the secret-key algorithm using a pseudo-random generator, leakage from multiple keys might be useful in HIBE (Hierarchical Identity-Based Encryption) and ABE (Attribute-Based Encryption) systems, based on IBE constructions. In these cases different secret keys have to be generated for the same identities, and as a result it is more difficult to apply leakage resilient techniques.

REFERENCES

[1] Cheng – Kang Chu, Sherman S.M. Chow, Weng Guey Tzeng, Jianying Zhou, and Robert H.Deng, Senior Member, IEEE, "Key – Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage", IEEE Transactions On Parallel And Distributed Systems, Vol. 25, No. 2, February 2014.

[2] Amit Sahai Brent R. Waters, "Fuzzy Identity Based Encryption", Lecture Notes in Computer Science Volume 3494, 2005, pp 457-473.

[3] Cong Wang, Sherman S.M. Chow, Qian Wang, Kui Ren, and Wenjing Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage", IEEE Transactions On Computers, Vol. 62, No. 2, February 2013.

[4] Giuseppe Ateniese. Kevin Fu. Matthew Green. Susan Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage", ACM Transactions on Information and System Security (TISSEC), Volume 9 Issue 1, Pages 1-30, February 2006.

[5] Josh Benaloh, Melissa Chase, Eric Horvitz, and Kristin Lauter, "Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records", Proceedings of the 2009 ACM workshop on Cloud computing security, Pages 103-114, 2009.

[6] Melissa Chase, Sherman S.M. Chow, "Improving Privacy and Security in Multi-Authority Attribute-Based Encryption", Proceedings of the 16th ACM conference on Computerand communications security, Pages 121-130, 2009.

[7] Mikhail J. Atallah, Keith B. Frikken, and Marina Blanton, "Dynamic and Efficient Key Management for Access Hierarchies", ACM Transactions on Information and System Security (TISSEC), Volume 12 Issue 3, January 2009.

[8] Sherman S.M. Chow1, Cheng-Kang Chu2, Xinyi Huang, "Dynamic Secure Cloud Storage with Provenance", Lecture Notes in Computer Science Volume 6805, 2012, pp 442-464.

[9] Tatsuaki Okamoto, "Achieving Short Ciphertexts or Short Secret-Keys for Adaptively Secure General Inner-Product Encryption", Lecture Notes in Computer Science Volume 7092, 2011, pp 138-159.

[10] Vipul Goyal, Omkant Pandeyy Amit Sahaiz Brent Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data", Proceedings of the 13th ACM conference on Computer and communications security, Pages 89 – 98, 2006