IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555
Vol.6, No3, May-June 2016

Online Key-Aggregated Cloud Based Cryptosystem

[1]Athunya C B, [2]Kadheeja Fathima M S M,[3]Reshma R,[4]Saranya R S

Mrs. Lekshmy P.L(Professor)

[1234]B.Tech,Department of Computer Science and Engineering

[1234]LBS Institute of Technology For Women,Trivandrum,Kerala,India.

athunyajinu@gmail.com,kadheejafathima.kf@gmail.com,reshmachinku@gmail.com,rssaranya8@gmail.com

Abstract—Cloud computing is a new paradigm and emerged technology for hosting and delivering resources over a network. Many challenging issues are still unclear in cloud-based environments. User Authentication is one of the most challenging issues in cloud-based environments and hence an efficient user authentication model that involves both of defined phases during registration and accessing processes is proposed. Cryptography is an effective way of protecting sensitive information as it is stored on media or transmitted through network communication paths.Data sharing is an important functionality in cloud storage. Here how to securely, efficiently, and flexibly share data with others in cloud storage is shown. A new public-key cryptosystems which produce constant-size cipher texts such that efficient delegation of decryption rights for any set of cipher texts are possible is described. 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 text 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. The first public-key client-controlled encryption for flexible hierarchy, which was yet to be known, is given. De-duplication concept is used to prevent same content being uploaded repeatedly by the same user or by different users.

Keywords-Cloud Storage, Data Sharing, Key Aggregate Encryption, De-duplication.

I. 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. Nowadays, it is easy to apply for free accounts for email, photo album, file sharing and/or remote access, with storage size more than 25GB (or a few dollars for more than 1TB). 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 co-resident 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 owner's anonymity. Likewise, cloud users probably will not hold the strong belief that the cloud server is doing a good job in terms of confidentiality. A cryptographic solution, with proven security relied on number-theoretic assumptions is more desirable, whenever the user is not perfectly happy with trusting the security of the VM or the honesty of the technical staff. 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 picture; an enterprise may 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.

II. RELATED WORK

The public cloud is a challenge to data security, since the data is no longer under your control. The cloud provider may have access to your data and the potential to share it with others. Industries which are heavily regulated, such as health care and financial industries must be doubly careful about sending data into the cloud.

A. Cryptographic Keys for a Predefined Hierarchy

Cryptographic key assignment schemes works on the basis of minimize the expense in storing and managing secret keys for general cryptographic use by using a tree structure . By using hierarchical tree structure [2], a key for a given branch can be used to derive the keys of its descendant nodes. This can solve the problem partially if one intends to share all files under a certain branch in the hierarchy which alternatively means that the number of keys increases with the number of branches. So it is difficult to create a hierarchy that can save the number of total keys to be granted for all individuals simultaneously.

B. Compact Key in Symmetric-Key Encryption

This method is used to generate a secret value instead of a pair of public/secret keys. It is designed for the symmetric-key setting in which the encryption gets the corresponding secret keys to encrypt data [5]. Thus it is unclear how to apply this idea for public key encryption scheme.

C. Compact Key in Identity-Based Encryption (IBE)

In this encryption, there is a trusted party called private key generator in IBE [6][7] which holds a master-secret key and gives a secret key to each user with respect to the user identity. The encryption can take the public parameter and a user identity to encrypt a message. The receiver can decrypt this cipher text by his secret key. Some tried to build IBE with key aggregation. But their key-aggregation compensate the expense of O (n) sizes for both cipher text and the public parameter, where n is the number of secret keys. This greatly increases the costs to store and transmit cipher text.

D. Attribute-based encryption (ABE)

This scheme [4] maintains each cipher text to be associated with an attribute, and the master-secret key holder can extract a secret key for a policy of these attributes so that a cipher text can be decrypted by this key. But the size of the key often increases linearly with the number of attributes it encompasses, or the cipher text-size is not constant.

E. Proxy Re-Encryption

Proxy re-encryption (PRE) [3] allows a proxy to convert a cipher text encrypted under one key into an encryption of the same message under another key. The main idea is to place as little trust and reveal as little information to the proxy as necessary to allow it to perform its translations. At the very least, the proxy should not be able to learn the keys of the participants or the content of the messages it re-encrypts.

III. Problem Definition

"To design an efficient encryption scheme which supports flexible delegation in the sense that any subset of the cipher texts (produced by the encryption scheme) is decryptable by a constant-size decryption key (generated by the owner of the master-secret key)."

To solve this problem aggregate keys are used. The files are encrypted using a symmetric encryption scheme. A set of requested files can be decrypted using a single aggregate key derived from the request id (request number). So this aggregate key can be used only for that particular request. Also the key used for encrypting the files is also stored in encrypted form to protect it. For encrypting the key, asymmetric encryption is used. The corresponding private and public keys are generated using identifiers called classes which can be specified by the data owner.

With our solution, owner can simply send client a single aggregate key via a secure e-mail. Client can download the encrypted files and then use this aggregate key to decrypt these encrypted files. The scenario is depicted in Fig.1. In our scheme, all the user's secret keys are tied to his identifier to resist the collusion attacks while the other users cannot know anything about the user's identifier. Notably, each user can join or leave the system freely without the need of reinitializing the system and there is no central authority. Furthermore, any access structure can be expressed in our scheme using the access tree technique.

IV. Key Aggregated Cryptosystem(KAC)

A key-aggregate encryption scheme [1] consists of five polynomial-time algorithms as follows. The data owner establishes the public system parameter via Setup and generates a public/master-secret key pair via KeyGen. Messages can be encrypted via Encrypt by anyone who also decides what ciphertext 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 ciphertext classes via Extract. The generated keys can be passed to delegatees securely (via secure e-mails or secure devices). Finally, any user with an aggregate key can decrypt any ciphertext provided that the ciphertext's class is contained in the aggregate key via Decrypt.

- Setup (1^{λ}, n): executed by the data owner to setup an account on an untrusted server. On input a security level parameter 1^{λ} and the number of ciphertext 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.

- KeyGen: executed by the data owner to randomly generate a public/master-secret key pair (pk,msk).

- Encrypt (pk, i, m): executed by anyone who wants to encrypt data. On input a public-key pk, an index i denoting the ciphertext class, and a message m, it outputs a ciphertext C.

- Extract (msk, S): executed by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegatee. 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 K_s.

- Decrypt (K_s, S, i, C): executed by a delegatee who received an aggregate key K_s generated by Extract. On input K_s, the set S, an index i denoting the ciphertext class the ciphertext C belongs to, and C, it outputs the decrypted result m if $i \in S$.

In our system, private key cryptography is adopted for encryption of the files. Public key encryption is used for encrypting the keys of each file. Also the classes created are used for creating the public and private keys for encrypting and decrypting the key with which each file is encrypted. The aggregate key is generated on the basis of request id or request number for the particular files requested. Hence the generated aggregate key cannot be used for decrypting the same set of files again through another request. So even if the aggregate key is leaked, it will not be useful to another user in the system.

V. ALGORITHMS

A. RIJNDAEL

Rijndael is a fast algorithm that can be implemented easily on simple processors. Although it has a strong mathematical foundation, it primarily uses substitution; transposition; and the shift, exclusive OR, and addition operations. Like DES, AES uses repeat cycles. There are 10,12, or 14 cycles for keys of 128, 192, and 256 bits, respectively. In Rijndael, the cycles are called "rounds". Each cycle consists of four steps.

- *Byte substitution:* This step uses a substitution box structure similar to the DES,substituting each byte of a 128-bit block according to a substitution table. This is a straight diffusion operation.
- *Shift row:* A transposition step. For 128- and 192-bit block sizes, row *n* is shifted left circular (*n* - 1) bytes; for 256-bit blocks, row 2 is shifted 1 byte and rows 3 and 4 are shifted 3 and 4 bytes, respectively. This is a straight confusion operation.
- *Mix column:* This step involves shifting left and exclusive-ORing bits with themselves.These operations provide both confusion and diffusion.
- *Add subkey:* Here, a portion of the key unique to this cycle is exclusive-ORed with the cycle result. This operation provides confusion and incorporates the key.

The Round Keysare derived from the Cipher Keyby means of a key schedule. The number of Round Keysnecessary to encrypt one block of information depends on the block length and key length as this determines the number of rounds. For a block length of 128 bits, 11 Round Keys(1 for initial round, 9 for standard rounds and 1 for the final round) are needed.

B. RSA

RSA encrypts messages through the following algorithm:

- Choose two distinct prime numbers p and q.
- Find n such that n = pq ,n will be used as the modulus for both the public and private keys.
- Find the totient of n, $\varphi(n)\varphi(n)=(p-1)(q-1)$.
- Choose e such that $1 < e < \varphi(n)$, and such that e and $\varphi(n)$ share no divisors other than 1 (e and $\varphi(n)$ are relatively prime), e is kept as the public key exponent.
- Determine d (using modular arithmetic) which satisfies the congruence relation,$de \equiv 1 (\mathrm{mod} \varphi(n))$.
- Cipher text c is computed by the expression, $c = m^e \pmod n$.
- Plain text m is computed by the expression, $m = c^d \pmod n$.

VI. PROPOSED FRAMEWORK

The proposed framework is applicable in a business enterprise which consists of the following modules:

- *Account:*In this module, both owners and clients can create an account by specifying their details. These details will be saved on the database. After registration they can login using their account.Login verification is done by comparing the entered details with that saved on the database.
- *Owner:***:**Here owners denote the ones who own the data which is uploaded to the cloud. They are the higher officials such as managers in the enterprise. This is the module where owners can upload their official data to cloud with all security. They have the right to generate their own individual cloud accounts. This module uses Rijndael managed encryption security to encrypt confidential files. To protect the key, keys are encrypted using RSA Algorithm. Owners can also search clients/staffs under them to assign appropriate work. Data sharing is done in a high secure manner.

  Also de-duplication is included. Uploading a file with content that has already been uploaded to cloud by that owner or some other owner is prevented. This is done to ensure that same job assignments are not given repeatedly to staffs. The entire content of a file is checked with that of already uploaded files using pattern matching.
- *Client:*These are users or staffs in enterprise. They can search work from cloud. If they want to access a particular file then they have to send a request to data owner. If data owner wants to share that file then an

aggregate key is provided based on the request id or request number for those particular files. Using this single key multiple files requested can be decrypted.

The overall flow of the proposed system is shown in the Fig. 1.

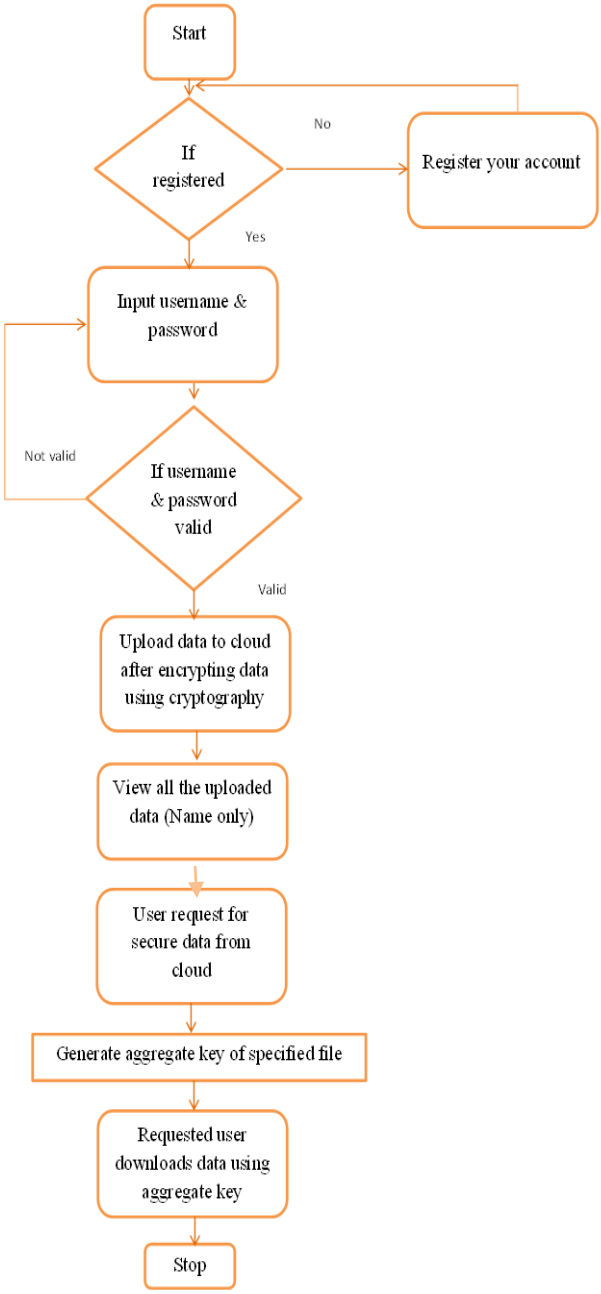

Fig.1: System Flow

VII. FUTURE WORK

To provide a secure environment where a data owner can share data with members of his group while preventing outsides from gaining any data access in case of malicious activities such as data loss and theft. Auditing and accountability in the cloud is a potential for future research in the context of data sharing in the cloud. Many users, in particular organizations and enterprises, benefit from data sharing in the cloud. However, there is always a likely chance that members of the group can carry out illegal operations on the data such as making illegal copies and distributing copies to friends, general public etc. in order to profit. A future research direction would be to find ways for a data owner to hold accountable any member that carries out malicious activities on their data. Another research direction would be to give the data owner physical access control over his data. Instead of accountability, the data owner can create a set of access control rules on his data and send the data along with the access control policy. In this way, any member with access to the data can only use the data in such a way that abides by the access control policy. If a member attempts to make illegal copies of the data, the access control policy should "lock" the data to prevent the member from doing so.

VIII. CONCLUSION

How to protect users' data privacy is a central question of cloud storage. With more mathematical tools, cryptographic schemes are getting more versatile and often involve multiple keys for a single application. In our project, we consider how to "compress" secret keys in public-key cryptosystems which support delegation of secret keys for different ciphertext classes in cloud storage. No matter which one among the power set of classes, the delegatee can always get an aggregate key of constant size. Our approach is more flexible than hierarchical key assignment which can only save spaces if all key-holders share a similar set of privileges. We proved security of the system in a model that gives the adversary his choice of public key and messages forge. An efficient client based encryption with the power of single aggregate key is adopted. Our project ensures strong belief in terms of confidentiality. The encrypted storage of keys provides additional security to the files uploaded. The use of request numbers for generating aggregate keys helps to ensure no further misuse of the aggregate key by others. Also the constant small key size decreases the cost and complexity of sharing keys over a channel.

REFERENCES

[1] Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage Cheng-Kang Chu, Sherman S. M. Chow, Wen-Guey Tzeng, Jianying Zhou, and Robert H. Deng, Senior Member, IEEE

[2] Flexible Access Control with Master Keys Gerald C. Chick and Stafford E. Tavares Queen's University at Kingston, G.Brassard (Ed.): Advances in Cryptology – CRYPTO '89,1 NCS 435, pp. 316-322, 1990

[3] Efficient Unidirectional Proxy Re-Encryption, Sherman S.M. Chow, Jian Weng, Yanjiang Yang, and Robert H. Deng

[4] Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data
Vipul Goyal, Omkant Pandey, Amit Sahai, Brent Waters

[5] Information Theoretically Secure Encryption with Almost Free Authentication Basel Alomair (Center of Excellence in Information Assurance (CoEIA) King Saud University, Riyadh, Saudi Arabia & Network Security Lab (NSL) University of Washington, Seattle, WA, USA alomair@uw.edu), Radha Poovendran (Network Security Lab (NSL) University of Washington, Seattle, WA, USA rp3@u.washington.edu), Journal of Universal Computer Science, vol. 15, no. 15 (2009), 2937-2956 submitted: 1/2/09, accepted: 29/8/09, appeared: 1/9/09

[6] 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.

[7] T. H. Yuen, S. S. M. Chow, Y. Zhang, and S. M. Yiu "Identity Based Encryption Resilientto Continual Auxiliary Leakage," in Proceedings of Advances in Cryptology-EUROCRYPT'12, ser.LNCS, vol.7237, 2012, pp. 117–134.