Key Aggregate Cryptosystem Method for Secure Data Sharing in Cloud Storage

A. V. Jadhav[1], B. M. Kore[2], V. V. Pottigar[3]
[1,2,3] ME Department of Computer Science & Engineering, Solapur University, Solapur
[1,2,3] SKN Sinhgad College of Engineering, Korti, Solapur, MS, India
[1]archu.j43@gmail.com

Abstract - ***Cloud Computing is a technology that uses the internet and central remote servers to maintain data and applications. Cloud computing allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access. In the key aggregate cryptosystem for cloud data sharing efficient public key encryption scheme which support flexible delegation in the sense that any subset of the cipher texts is decryptable by a constant-size decryption key. The secret key holder can release a constant size aggregate key for flexible choices of cipher text in cloud storage. This paper describes cryptographic technique for securely and efficiently data sharing in cloud storage using constant size aggregate key.***

Keywords - cloud storage, data sharing, data security, aggregate key, cryptography.

I. INTRODUCTION

Cloud system can be used to enable data sharing capabilities and this can proven abundant of benefits to the user. There is currently a push for IT organization to increase their data sharing efforts. In enterprise settings, there is 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. With current technology user can access almost all of their files or emails by mobile phone or computer from any corner of the world.

Cloud storage is a model of data storage where the digital data is stored in logical pools, the physical storage spans multiple servers (and often locations), and the physical environment is typically owned and managed by a hosting company. These cloud storage providers are responsible for keeping the data available and accessible, and the physical environment protected and running. People and organizations buy or lease storage capacity from the providers to store end user, organization, or application data. Cloud storage services may be accessed through a co-located cloud compute service, a web service application programming interface (API) or by applications that utilize the API, such as cloud desktop storage, a cloud storage gateway or Web-based content management systems. Considering data privacy, a traditional way to ensure it is to rely on the server to enforce the access control after authentication (e.g., [6]), which means any unexpected privilege escalation will expose all data. In a shared-tenancy cloud computing environment, things become even worse.

In the cloud storage efficient public key encryption scheme which support flexible delegation in the sense that any subset of the cipher texts is decryptable by a constant-size decryption key. In other words, the secret key holder can release a constant-size aggregate key for flexible choices of ciphertext set in cloud storage [1].

In KAC user can encrypt message not only under a public-key but also under an identifier of cipher texts called class. The ciphertexts are further categorized into different classes. The key owner holds a master-secret key called master secret key. The extracted key have can be an aggregate key which is as compact as a secret key for a single class, but aggregate the power of many such keys, i.e., the decryption power of any subset of cipher text classes.

Cryptography helps the data owner to share the data to in safe way. Cryptography is the practice and study of hiding information. It is the Art or Science of converting a plain intelligible data into an unintelligible data (i.e. encryption) and again retransforming that message into its original form (i.e. decryption). It provides Confidentiality, Integrity, and Accuracy [7].

A cryptographic solution, with proven security relied on number-theoretic assumptions is more desirable Data sharing is important functionality in cloud storage [8]. For example bloggers can let their friends view private data or an enterprise may grant their employee access to important data. But the problem is how to effectively share encrypted data. Of course users can download the encrypted data from the storage, decrypt them and then send them to others for sharing, but it loses the value of cloud storage. So user should be able to give access rights of sharing data to others so that they can access these data from server directly.

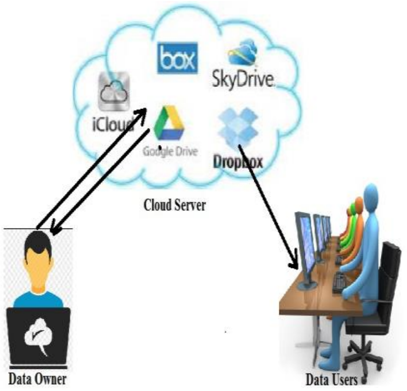

Figure 1: Cloud System

II. RELATED WORK

Key assignment scheme aim to minimize the expense in storing and managing secret keys for general cryptographic use. Only hash functions are used for a node to derive a descendant’s key from its own key. The space complexity of the public information is the same as that of storing hierarchy and is asymptotically optimal; the private information at a node consists of a single key associated with that node and updates are handled locally in the hierarchy [2].

Presented an encryption scheme which is originally proposed for concisely transmitting large number of keys in broadcast scenario. In this paper build an efficient system that allows patients both to share partial access rights with others, and to perform searches over their records. They formalize the requirements of a Patient Controlled Encryption scheme, and give several instances, based on existing cryptographic primitives and protocols, each achieving a different set of properties. The encryptor needs to get the secret keys to encrypt data which is not suitable for many applications. It is unclear how to apply this method for public key encryption scheme [3].

Identity-based encryption (IBE) 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 (PKG) in IBE which holds a master-secret key and issues a secret key to each user with respect to the user identity. The encryptor can take the public parameter and a user identity to encrypt a message. The recipient can decrypt this ciphertext by his secret key [4].

Attribute-based encryption (ABE) allows each ciphertext 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 ciphertext can be decrypted by this key if its associated attribute conforms to the policy. However, the major concern in ABE is collusion-resistance but not the compactness of secret keys. Indeed, the size of the key often increases linearly with the number of attributes it encompasses or the ciphertext-size is not constant [5].

III. PROPOSED ALGORITHM

This project consists of five algorithms which are used to perform the above operations. These algorithms are as follow:

Setup: The account is created on the untrusted server for sharing of data. This account is generated by data owner.

KeyGen: This algorithm is use for the generation of public key. The data owner generates a public secrete key to encrypt the data over cloud. He also creates an aggregate key to access the block of ciphers of limited size.

Encrypt: This algorithm encrypts the data provided by the data owner by using the secrete key. This encrypted data is then share among the cloud.

Extract: The aggregate key is use for extracting the particular block of the ciphers from the cipher file. But other encrypted data remains secure.

Decrypt: The encrypted data is then decrypted by using the same secrete key which is use for encryption. As the above figure shows, the key assignment is done in dynamic way. The aggregate key is use to decrypt only those ciphers which user wants. This key will not decrypt the other remaining ciphers. The main encryption and decryption is done by the secrete key. If any user enters the wrong secrete key or wrong aggregate key then the user contains will be blocked by the data owner. And the information which that user tries to retrieve is then added into non confidential storage. Only data owner can unblock that user contents and he may transfer the information from non-confidential storage to confidential storage. The user can only access the data on cloud if he has secreted key and the aggregate key, otherwise he will be block forever.

MD5: MD5 algorithm can be used as a digital signature mechanism. It takes as input a message of arbitrary length and produces as output a 128 bit fingerprint or message digest of the input. It intended where a large file must be compressed in a secure manner before being encrypted with a private key under a public-key cryptosystem such as PGP.

The main steps of MD5 algorithm to generate the hash value are given as below:

1. Append padding bits so message becomes 448 modules 512.
2. Append length to the input message so that it becomes exact 64-bit in length.
3. Initialize the 32 bit MD buffer A, B, C, D.
4. Process the message in 16-word block,

F (X, Y, Z) = XY or not (X) Z
G (X, Y, Z) = XZ or Y not (Z)
H (X, Y, Z) = X xor Y xor Z
I (X, Y, Z) = Y xor (X or not (Z))

5. The final digest message will be stored in buffer.

IV. SYSTEM ARCHITECTURE

The proposed system is basically design on the basis of key aggregation encryption. Here we are using two keys to encrypt and decrypt the data which are secret key and its aggregate key.

The data owner creates the public system parameter and generates a secrete key which is public key pair. Data can be encrypted by any user and he may decide cipher text block associated with the plaintext file which wants to be encrypted. The data owner have rights to use the secret key from which he can generate an aggregate key which is use for decryption of a set of cipher text blocks. The both keys can be sent to end user in very secure manner. The authenticated user having an aggregate key can decrypt any block of cipher text.

In the proposed system there are three main parts i.e. data owner, data user and cloud service providers. User will get blocked if he enter wrong aggregate key for three times. The following figure shows the working of proposed system In the aggregate key cryptosystem authentication is very important if user fails to provide valid credentials then data owner may be block to the particular user. User either one of sender or receiver. Permission function is the functions such as read write and update. Encryption function encrypt data using public key that key size is fixed for every user but it can be generated dynamically by using MD5 algorithm. The above figure 2 shows the aggregate key master key generation location. Each user will get different and unique key as per the request generated by user. Initially the public key is used to encrypt the file and then it get merge into the master key. When any user request for the data then he need to provide the index of particular file. Then the keys are get extracted from the master key and again we form aggregate key from this keys. This aggregate key is of constant size. To generate this aggregate key we have used MD5 algorithm.

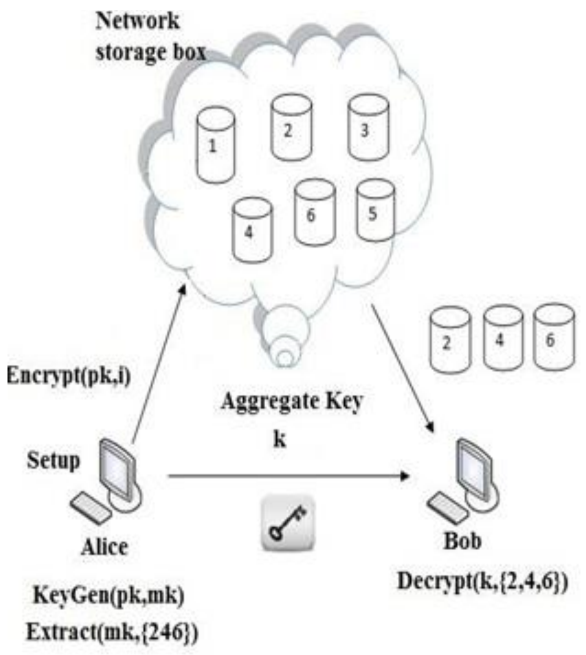

Figure 2: System Architecture.

KAC is developed for the secure data sharing. Data owner can send his data with secure and confidently. KAC is very secure and reliable method for sharing data in cloud computing. The aim of KCA is illustrated in Figure 2. For sharing the selected file with user cloud service provider first checks the rights of particular user. If he having rights for that file then only user can perform particular office. Later the public/master key pair (pk, mk) is generated by executing the KeyGen. The master key is kept secret and the public key pk and param are made public to access the file.

SYSTEM DESIGN

1 User Module
Set (C) = {c0, c1, c2, c3, c4}
C0= User Registration
C1= Upload file.
C2= Generate secret key.
C3= Encrypt or decrypt files.
C4= Download file.

2 Cryptographic modules
Set (G)={g1, g2, g3, c2, c3, c4}
G1=Secrete key generation.
G2= Encrypt and share file.
G3=Decrypt file using secrete key.

3 Extraction Modules
Set (E) = {e0, e1, e2, c3, g3}
E0=Receive key.
E1=Extract key data.
E2=Decrypt Data.

4 Union and Intersection
Set (G) = {g1, g2, g3, c2, c3, c4}
Set (E) = {e0, e1, e2, c3, g3}
C Intersection G= {c2, c3, c4}

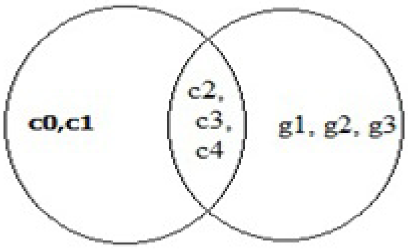

Figure 3: Set Representation

V. RESULT ANALYSIS

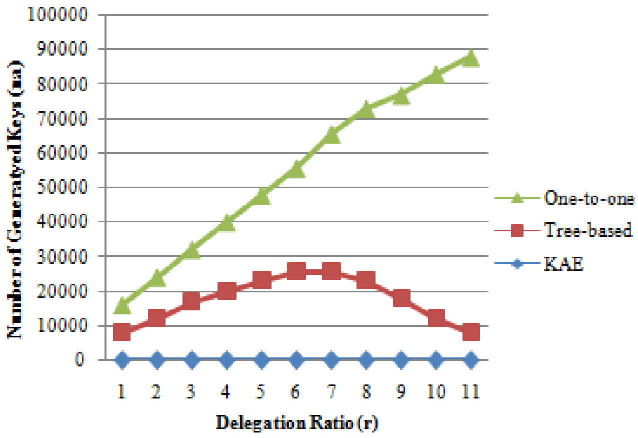

Figure 4: Number of granted keys (*na*) required for different approaches

Looking at the performance analysis, a comparison of the number of keys granted between three methods is shown in the Fig. 4. Here we can see, in one by one key granting, the number of granted keys will be same as the number of ciphertext delegate classes. With the tree based structure, the number of keys granted can be saved depending on the delegation ratio. Whereas in KAC scheme, it is efficiently implemented with the fixed size aggregate key. The constant-size aggregate key and constant-size ciphertext is the greatest advantage of this scheme. The Key Aggregate Cryptosystem (KAC) is the most efficient scheme when compared to the tree based structure and one by one granting of the keys.

VI. CONCLUSIONS

To share data flexibly and securely in cloud computing is vital thing. Users always prefer to upload there data on cloud and share the uploaded data among different users. The main drawback of cloud computing is the security issue. Cryptography is a one of best solution which provides security to share selected data with desired cloud data users. Sharing of decryption keys in secure way plays important role. The proposed Public-key cryptosystems provides delegation or leader key of secret keys for different cipher text classes in cloud storage. The proposed system creates user group and can share file to all group members simultaneously. Cryptographic schemes are getting more versatile and trustable, it involve multiple keys for a single application. In this paper, we consider how we can ”compress” secret keys by combining the multiple keys which support delegation or aggregation of secret keys for different cipher text classes in cloud storage system.

REFERENCES

[1] Cheng-Kang Chu et.al, “Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage”, IEEE Transactions on Parallel and Distributed Systems. Volume: 25, Issue: 2. Year :2014.

[2] M. J. Atallah et.al, “Dynamic and Efficient Key Management for Access Hierarchies,” ACM Transactions on Information and System Security (TISSEC), vol. 12, no. 3, 2009.

[3] J. Benaloh et.al, “Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records,” in Proceedings of ACM Workshop on Cloud Computing Security (CCSW ’09). ACM, 2009, pp. 103–114.

[4] S. S. M. Chow et.al, “Practical Leakage-Resilient Identity-Based Encryption from Simple Assumptions,” in ACM Conference on Computer and Communications Security, 2010, pp. 152–161.

[5] M. Chase and S. S. M. Chow, “Improving Privacy and Security in Multi-Authority Attribute-Based Encryption,” in ACM Conference on Computer and Communications Security, 2009, pp. 121–130.

[6] S. S. M. Chow, Y. J. He, L. C. K. Hui, and S.-M. Yiu, “SPICE – Simple Privacy-Preserving Identity-Management for Cloud Environment,” in Applied Cryptography and Network Security – ACNS 2012, ser. LNCS, vol. 7341. Springer, 2012, pp. 526–543.

[7] S. S. M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R. H. Deng, “Dynamic Secure Cloud Storage with Provenance,” in Cryptography and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday, ser. LNCS, vol. 6805. Springer, 2012, pp. 442–464.

[8] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, “Privacy- Preserving Public Auditing for Secure Cloud Storage,” *IEEE Trans. Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[9] B. Wang, S. S. M. Chow, M. Li, and H. Li, “Storing Shared Data on the Cloud via Security-Mediator,” in International Conference on Distributed Computing Systems - ICDCS 2013. IEEE, 2013.

[10] L. Hardesty, “Secure computers aren’t so secure,” MIT press, 2009, http://www.physorg.com/news176107396.html

[11] D. Boneh, R. Canetti, S. Halevi, and J. Katz, “Chosen-Ciphertext Security from Identity-Based Encryption,” *SIAM Journal on Computing (SIAMCOMP)*, vol. 36, no. 5, pp. 1301–1328, 2007.

[12] F. Guo, Y. Mu, and Z. Chen, “Identity-Based Encryption: How to Decrypt Multiple Ciphertexts Using a Single Decryption Key,” in Proceedings of Pairing-Based Cryptography (Pairing ’07), ser. LNCS, vol. 4575. Springer, 2007, pp. 392–406.

[13] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, “Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.