iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** II **Month of publication:** February 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Extended Cryptographic Mechanism for Secure and Scalable Data Sharing in Cloud using Multiple Keys

Swati Kutke[#1], Shraddha Karmase[*2], Prachi Bodake[#3] ,Vrushali Tajane[#4]
[#]*Computer Department, Savitribai fule pune University*

Abstract - Cloud computing technology is widely used now-a-days. Data sharing is an important functionality in cloud storage. Data can be outsourced on cloud and can access easily. Different users can share that data through different virtual machines but present on single physical machine. But there is one loop hole in this i.e users don't have physical control over the outsourced data. In this project, we show how to securely, efficiently, and flexibly share data with others in cloud storage. The data owner can share the data to other users in safe way using cryptography. So sender encrypts data and uploads it on cloud. Different encryption and decryption keys are generated for different data. The encryption and decryption keys are different for different set of data. Only those set of decryption keys are shared that the selected data can be decrypted. Here a public-key cryptosystems produces a ciphertext which is of constant size so that delegation of decryption rights for any set of ciphertexts is possible. One can combine a set of secret keys to form a single key of small size which has the power of all the keys being combined. This compact aggregate key of constant size can be efficiently sent to others via secure channel like email or can be stored in a smart card.
Keywords— Cloud storage, Attribute base encryption, Identity base encryption, Cloud storage, data sharing, key-aggregate encryption.

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. Its flexible and cost optimizing characteristic motivates the end user as well as enterprises to store the data on cloud. The insider attack is one of security concern which's needs to be focused. Cloud Service provider need to make sure whether audits are held for users who have physical access to the server. As cloud service provider stores the data of different users on same server it is possible that user's private data is leaked to others. The public auditing system of data storage security in cloud computing provides a privacy-preserving auditing protocol. It is necessary to make sure that the data integrity without compromising the anonymity of the data user. To ensure the integrity the user can verify metadata on their data, upload and verify metadata.

The main concern is how to share the data securely, the answer is cryptography. The question is how can the encrypted data is to be shared. The user must provide the access rights to the other user as the data is encrypted and the decryption key should be send securely. For an example Alice keeps her private data i.e. photos on dropbox and she doesn't want to share it with everyone. As the attacker may access the data so it is not possible to rely on predefine privacy preserving mechanism so she all the photos were encrypted by her on encryption key while uploading it.

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. 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. 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. The present research efforts mainly focus on minimizing the communication requirements (such as bandwidth, rounds of communication) like aggregate signature.This is shown in figure1

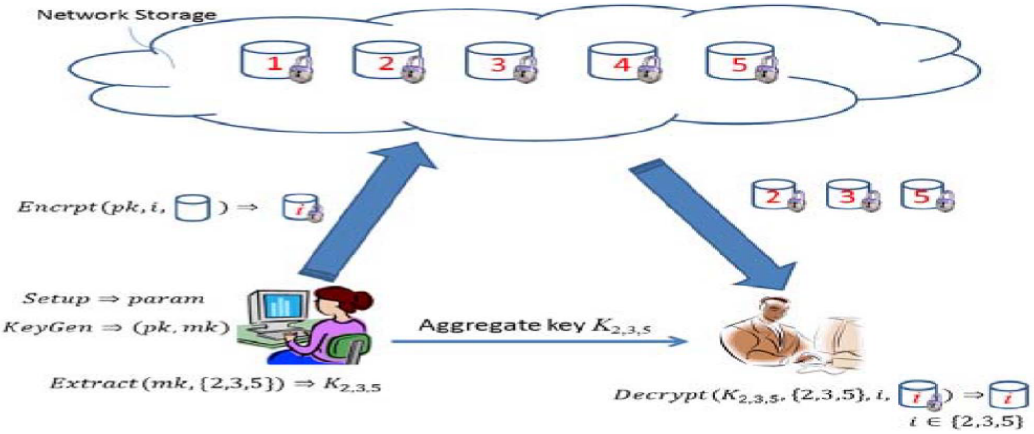
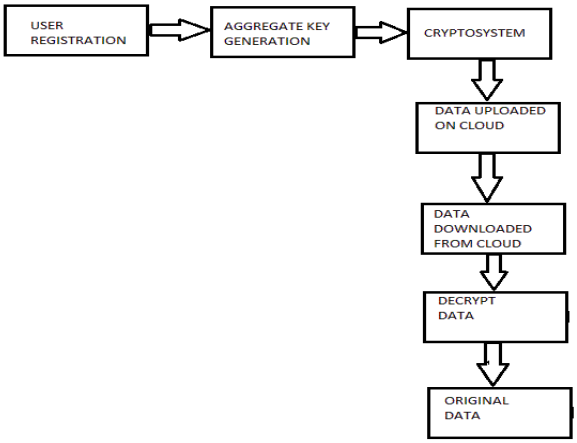


Figure 1

The block diagram of our project is given below.



II. LITERATURE SURVEY

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. For example, with the secret key for the policy (2v3v6v8), one can decrypt ciphertext tagged with class 2, 3, 6, or 8. 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.To delegate the decryption power of some ciphertexts without sending the secret key to the delegatee, a useful primitive is proxy re-encryption (PRE). A PRE scheme allows sender to delegate to the server(proxy) the ability to convert the ciphertexts encrypted under his public-key into ones for receiver. Nevertheless, sender has to trust the proxy that it only converts ciphertexts according to her instruction ,which is what we want to avoid at the first place. Even worse, if the proxy colludes with receiver, some form of senders secret key can be recovered which can decrypt senders(convertible) ciphertexts without receivers further help. That also means that the transformation key of proxy should be well protected. Using PRE just moves the secure key storage requirement from the delegatee to the proxy. It is, thus, undesirable to let the proxy reside in the storage server. That

will also be inconvenient since every decryption requires separate interaction with the proxy.
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 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. In fuzzy IBE [21], one single compact secret key can decrypt ciphertexts encrypted under many identities which are 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.

III. KEY-AGGREGATE ENCRYPTION

A key aggregate encryption has five polynomial-time algorithms as:

A. Setup Phase

The data owner executes the setup phase for an account on server which is not trusted. The setup algorithm only takes implicit security parameter.

B. KeyGen Phase

This phase is executed by data owner to generate the public or the master key pair (pk, msk).

C. Encrypt Phase

This phase is executed by anyone who wants to send the encrypted data. Encrypt (pk, m, i), the encryption algorithm takes input as public parameters pk, a message m, and i denoting ciphertext class. The algorithm encrypts message m and produces a ciphertext C such that only a user that has a set of attributes that satisfies the access structure is able to decrypt the message.

1) Input= public key pk, an index i, and message m
2) Output = ciphertext C.

D. Extract Phase

This is executed by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegate.

1) Input = master-secret key mk and a set S of indices corresponding to different classes
2) Outputs = aggregate key for set S denoted by k_S.

E. Decrypt Phase

This is executed by the candidate who has the decryption authorities. Decrypt (kS, S, i, C), the decryption algorithm takes input as public parameters pk, a ciphertext C, i denoting ciphertext classes for a set S of attributes.

1) Input = kS and the set S, where index i = ciphertext class
2) Outputs = m if i element of S

IV. AGGREGATE KEY GENERATION ALGORITHM

A. First Setup Data

	USER1	USRE2	USER3	PUBLIC KEY	
FILE1	SEND	NO	NO	K1	→ PUBLIC +PRIVATE
FILE2	SEND	SEND	NO	K2	→ PUBLIC +PRIVATE
FILE3	NO	SEND	NO	K3	→ PUBLIC +PRIVATE

B. All the key like k1, k2, k3 are in string format and then it will be converted into bytes using Byte Encoder.
C. Then every string is converted into number , for eg.,

K1=12356
K2=56423
K3=35641

D. All these keys are combined but a 0 is inserted between them as a separator, like,
12356 '0' 56423 '0' 35641

E. Secret key ie S.

F. Key convolution : we use the quardratic equation,
f(x) = (n1x + n2x + S) / n1=94, n2/66
here the x is consider as 2 or any number.

G. After calculation we get a number like this, 254631, then that number is again converted in String.

H. Display String format of key.

V. CONCLUSION

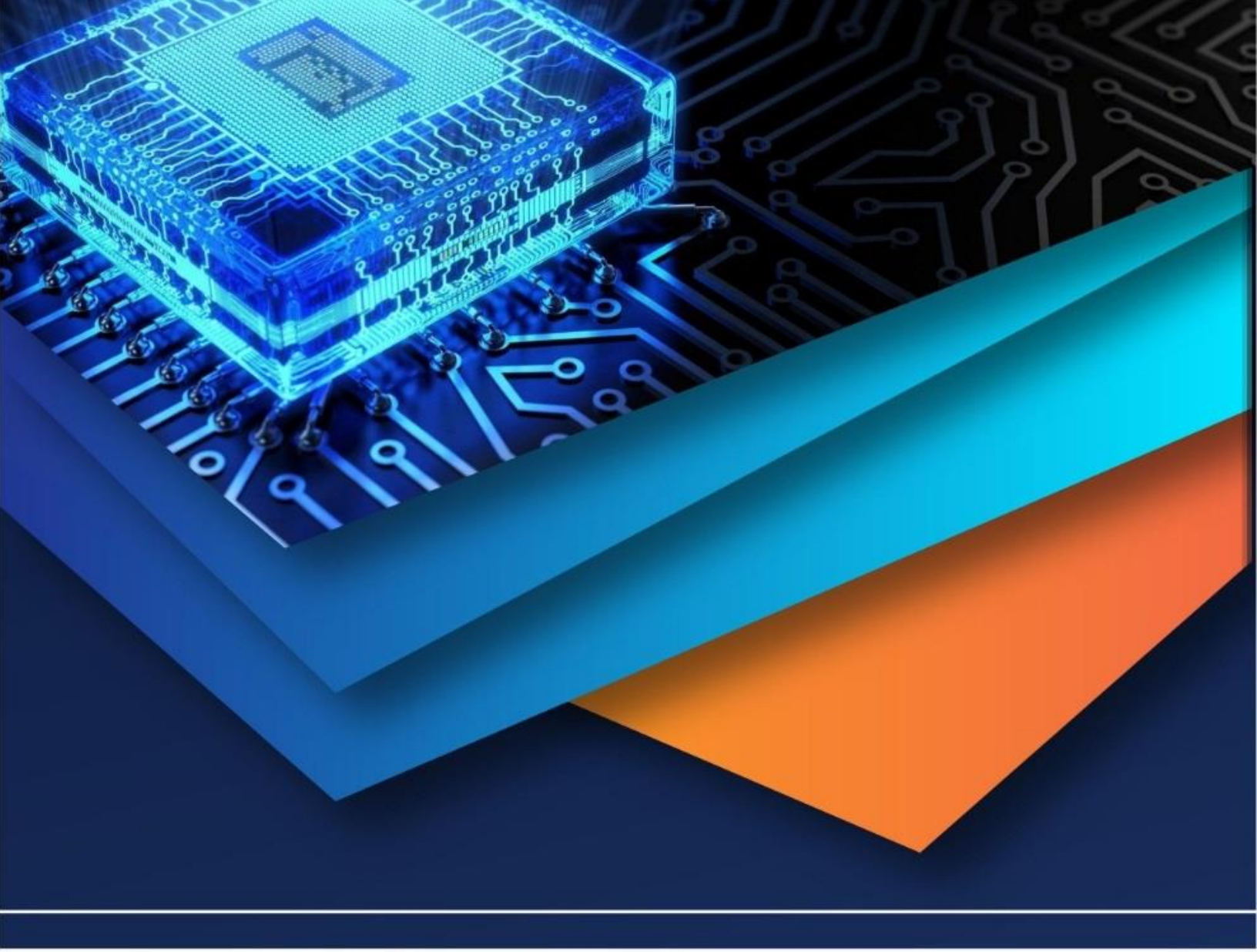
How to protect user's data privacy is a central question of cloud storage. With more mathematical tools, cryptographic schemes are getting more versatile and often involve multiple keys for a single application. To share data flexibly is vital thing in cloud computing. Users prefer to upload there data on cloud and among different users. Outsourcing of data to server may lead to leak the private data of user to everyone. Encryption is a one solution which provides to share selected data with desired candidate. Sharing of decryption keys in secure way plays important role. Public-key cryptosystems provides delegation of secret keys for different ciphertext classes in cloud storage. The delegatee gets securely an aggregate key of constant size. Thus we have partially build our system that provides scalable data sharing in cloud. We are providing security using multiple key encryption. Thus, with our system, data can be shared in cloud very securely, efficiently and flexibly.

VI. ACKNOWLEDGEMENTS

First and foremost, I would like to thank my guide, Prof. Archana Vaidya, for her guidance and support. I will forever remain grateful for the constant support and guidance extended by her in making this project. Through our many discussions, she helped me to form and solidify ideas. The invaluable discussions we had with her, the penetrating questions she has put to me and the constant motivation, has all led to the development of this project. I wish to express my sincere thanks to the Head of department, Prof. N.V.Alone and also grateful thanks to Prof.Lahane Sir and the departmental staff members for their support. I would also like to thank to my friends for listening to my ideas, asking questions and providing feedback and suggestions for improving my ideas.

REFERENCES

[1] Cheng-Kang Chu, Sherman S.M. Chow, Wen-Guey Tzeng, Jianying Zhou, and Robert H. Deng, Senior Member, IEEE Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage
[2] T. Okamoto and K. Takashima, "Achieving Short Ciphertexts or Short Secret-Keys for Adaptively Secure General Inner-Product Encryption," Proc. 10th Int'l Conf. Cryptology and Network Security (CANS '11), pp. 138-159, 2011.
[3] S.S.M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R.H. Deng, "Dynamic Secure Cloud Storage with Provenance," Cryptography and Security, pp. 442-464, Springer, 2012.
[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, J. Weng, Y. Yang, and R.H. Deng, "Efficient Unidirectional Proxy Re-Encryption," Proc. Progress in Cryptology (AFRICACRYPT '10), vol. 6055, pp. 316-332, 2010..
[6] T.H. Yuen, S.S.M. Chow, Y. Zhang, and S.M. Yiu, "Identity-Based Encryption Resilient to Continual Auxiliary Leakage," Proc. Advances in Cryptology Conf. (EUROCRYPT '12), vol. 7237, pp. 117-134, 2012
[7] 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. ,
[8] 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.
[9] S.S.M. Chow, Y.J. He, L.C.K. Hui, and S.-M. Yiu, ?SPICE - Simple Privacy- Preserving Identity-Management for Cloud Environment,?Proc. 10th Int?l Conf. Applied Cryptography and Network Security (ACNS), vol. 7341, pp. 526- 543, 2012
[10] 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.
[11] 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.,



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY