ONLINE DATA SHARING USING SECURE KEY AGGREGATE CRYPTOSYSTEM ON CLOUD

[1]Supriya M,[2]Mrs.Pushpalatha R

[1]Mtech in CS&E,[2]Assistant Professor, DOS in CS&E
[1]Department of Studied in Computer Science and Engineering,
[1] VTU PG Centre, Mysore, India

Abstract: *Sharing of data in cloud is widely used now a day. Cloud is used for storing data and applications on servers and accessing them through internet. Online data sharing for improved productivity and efficiency is one of the primary requirements today for any organization. In cloud data is stored on single machine and this stored data shared among multiple users by different machines. To store and also for sharing data securely cryptosystem is used. The data owner encrypt the data before it is upload to the cloud and then data decryption is done when user want to access it. In the existing system owner needs to generate individual key to the individual user. To overcome this problem in proposed system we generate a aggregate key and broadcast.*

Index Terms - **Cloud storage, data sharing, key-aggregate encryption, Cloud Computing, Key Aggregate Cryptosystem**

I. INTRODUCTION

Outsourcing of data is increasingly demanded in enterprise settings. In outsourcing of data there are chances of stolen data from virtual machine. On separate virtual machine is used to access the data of cloud stored on single physical machine. Proposed system supports KAC scheme which contains various security levels. In 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. When the user is not perfectly happy with trusting the security of the virtual machine. The shared data in cloud servers, however usually contains user's sensitive information such as personal profile, financial data health records etc…and needs to be well protected. Data cryptography mainly is the scrambling of the content of the data, such as text, image, audio, video and so forth to make the data unreadable, invisible or meaningless during transmission or storage is termed encryption. The main aim of cryptography is to take care of data secure from attacker. 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.

II. RELATED WORK

2.1 Spice–simple privacy-preserving identity-management for cloud environment:

According to Sherman SM Chow, Yi-Jun He Identity security and privacy have been regarded as one of the top seven cloud security threats. There are a few identity management solutions proposed recently trying to tackle these problems. However, none of these can satisfy all desirable properties. In particular, *unlinkability* ensures that none of the cloud service providers (CSPs), even if they collude, can link the transactions of the same user. On the other hand, *delegatable authentication* is unique to the cloud platform, in which several CSPs may join together to provide a packaged service, with one of them being the source provider which interacts with the clients and performs authentication while the others will be transparent to the clients. Note that CSPs may have different authentication mechanisms that rely on different attributes. Moreover, each CSP is limited to see only the attributes that it concerns. This paper presents SPICE – the first digital identity management system that can satisfy these properties in addition to other desirable properties. The novelty of our scheme stems from combining and exploiting two group signatures so that we can randomize the signature to make the same signature look different for multiple uses of it and hide some parts of the messages which are not the concerns of the CSP. Our scheme is quite applicable to cloud systems due to its simplicity and efficiency.

2.2 Dynamic secure cloud storage with provenance:

According to Sherman SM Chow, Cheng-Kang Chu One concern in using cloud storage is that the sensitive data should be confidential to the servers which are outside the trust domain of data owners. Another issue is that the user may want to preserve his/her anonymity in the sharing or accessing of the data (such as in Web 2.0 applications). To fully enjoy the benefits of cloud storage, we need a confidential data sharing mechanism which is fine-grained (one can specify *who* can access *which classes* of his/her encrypted files), dynamic (the total number of users is not fixed in the setup, and any new user can decrypt previously encrypted messages), scalable (space requirement does not depend on the number of decryptors), accountable (anonymity can be revoked if necessary) and secure (trust level is minimized).This paper addresses the problem of building a secure cloud storage system which supports dynamic users and data provenance. Previous system is based on specific constructions and does not offer all of the aforementioned desirable properties. Most importantly, dynamic user is not supported. We study the various features offered by cryptographic anonymous authentication and encryption mechanisms; and instantiate our design with verifier-local revocable group signature and identity-based broadcast encryption with constant size cipher texts and private keys. To realize our concept, we equip the broadcast encryption with the dynamic cipher text update feature, and give formal security guarantee against adaptive chosen-cipher text decryption and update attacks.

2.3 Privacy-preserving public auditing for secure cloud storage:

According to Cong Wang, Sherman S.-M. Chow Using cloud storage, users can remotely store their data and enjoy the on-demand high-quality applications and services from a shared pool of configurable computing resources, without the burden of local data storage and maintenance. However, the fact that users no longer have physical possession of the outsourced data makes the data integrity protection in cloud computing a formidable task, especially for users with constrained computing resources. Moreover, users should be able to just use the cloud storage as if it is local, without worrying about the need to verify its integrity. Thus, enabling public audit ability for cloud storage is of critical importance so that users can resort to a third-party auditor (TPA) to check the integrity of outsourced data and be worry free. To securely introduce an effective TPA, the auditing process should bring in no new vulnerabilities toward user data privacy, and introduce no additional online burden to user. In this paper, we propose a secure cloud storage system supporting privacy-preserving public auditing. We further extend our result to enable the TPA to perform audits for multiple users simultaneously and efficiently. Extensive security and performance analysis show the proposed schemes are provably secure and highly efficient. Our preliminary experiment conducted on Amazon EC2 instance further demonstrates the fast performance of the design.

2.4 Key-aggregate cryptosystem for scalable data sharing in cloud storage:

According to Cheng-Kang Chu, Sherman SM Chow Data sharing is an important functionality in cloud storage. In this paper, we show how to securely, efficiently, and flexibly share data with others in cloud storage. We describe new public-key cryptosystems that produce constant-size cipher texts such that efficient delegations 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 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. 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.

2.5 Collusion resistant broadcast encryption with short cipher texts and private keys:

According to Dan Boneh, Craig Gentry We describes two new public key broadcast encryption systems for stateless receivers. Both systems are fully secure against any number of colluders. In our first construction both cipher texts and private keys are of constant size (only two group elements), for any subset of receivers. The public key size in this system is linear in the total number of receivers. Our second system is a generalization of the first that provides a tradeoff between cipher text size and public key size. For example, we achieve a collusion resistant broadcast system for n users where both cipher texts and public keys are of size $O(N−−√)$ for any subset of receivers. We discuss several applications of these systems.

III. PROPOSED SYSTEM

In this paper, we attempt to build precisely such a data sharing framework that is provably secure and at the same time, efficiently implementable. In this paper we propose an efficiently implementable version of the basic key-aggregate cryptosystem (KAC) using asymmetric bilinear pairings. We propose a CCA-secure fully collusion resistant construction for the basic KAC scheme with low overhead cipher texts and aggregate keys. We demonstrate how the basic KAC frame work may be efficiently extended and combined with broadcast encryption schemes for distributing the aggregate key among an arbitrary number of data users in a real-life data sharing environment. The extension has a secure channel requirement of O(m + m0) for m data users and m0 data owners. In addition, the extended construction continues to have the same overhead for the public parameters, cipher texts and aggregate keys, and does not require any secure storage for the aggregate keys, which are publicly broadcast.

3.1 ADVANTAGES OF PROPOSED SYSTEM:

- In this paper, we attempt to build precisely such a data sharing framework that is provably secure and at the same time, efficiently implementable.
- In this paper we propose an efficiently implementable version of the basic key-aggregate cryptosystem (KAC) using asymmetric bilinear pairings.

- ❖ We propose a CCA-secure fully collusion resistant construction for the basic KAC scheme with low overhead cipher texts and aggregate keys.
- ❖ We demonstrate how the basic KAC frame work may be efficiently extended and combined with broadcast encryption schemes for distributing the aggregate key among an arbitrary number of data users in a real-life data sharing environment. The extension has a secure channel requirement of O(m + m0) for m data users and m0 data owners.

3.2 MODULES OF DECRIPTIONS:

File Upload:

Whenever a need to share data among the group arises, the owner of the file sends the encryption request to the CS. The request is accompanied by the file (*F*) and a list (*L*) of users that are to be granted access to the file. *L* also contains the access rights for each of the users. The users may have READ-only and/or READ–WRITE access to the file. Other parameters can be also set to enforce fine-grained access control over the data. *L* is used to generate the ACL for the data by the CS. *L* is sent to the CS only if the data are to be shared with a new proposed group. If the group already exists, the encryption request will not contain *L*; rather, the group ID of the existing group will be sent. The CS, after receiving the encryption request for the file, generates the ACL from the list and creates a group of the users. The ACL is separately maintained for each file. The ACL contains information regarding the file such as its unique ID, size, owner ID, the list of the user IDs with whom the file is being shared, and other metadata. If the group already existed, only the ACL for the file is created. Next, the CS generates *K* according to the procedure defined in Section III-B and encrypts the file with an appropriate symmetric block cipher (we have used the AES for encryption purposes). The result is an encrypted file (*C*). Subsequently, the CS generates *Ki* and *K_ i* for every user and deletes *K* by secure overwriting. Secure overwriting is a concept in which the bits in the memory are constantly flipped to make sure that a memory cell never grips a charge for enough duration for it to be remembered and recovered. The *Ki* for each user is inserted into the ACL for later use.

File Download:

The authorized user sends a download request to the CS or downloads the encrypted file (*C*) from the cloud and sends the decryption request to the CS. The cloud verifies the authorization of the user through a locally maintained ACL. The decryption request is accompanied by the user portion of the key, i.e., *K_ i*, along with other authentication credentials. The CS computes *K* by applying XOR operation over *K_ i* and the corresponding *Ki* from the ACL. As each of the users correspond to a different pair of *Ki* and *K_ i*, none of the users can use other users' *K_ i* to masquerade identity. Subsequently, the CS proceeds with the decryption process after verifying the integrity of the file. If the correct *K_ i* is received by the CS, the result will be a successful decryption process; otherwise, the decryption will fail. After successful decryption, the file is sent to the requesting user through a secure communication channel that could be Secure Sockets Layer (SSL) or Internet Protocol Security (IPSec) channels. *K* is deleted via secure overwriting from the CS after decryption. The users are authenticated before the request processing according to standard procedures. Similar to the file upload process, the downloading of the file can be also done by the CS on behalf of the user. In the aforesaid case, the decryption request is sent to the CS. The CS, after authenticating the user, sends the download request to the cloud for the specified file. The cloud sends the encrypted file (*C*) to the CS. The rest of the process for the decryption is the same.

File Update:

Updating the file has a similar procedure to that of uploading the file. The difference is that, while updating, all of the activities related to the creation of the ACL and key generation are not carried out. The user, who has downloaded the file and made any changes, sends an update request to the CS. The request contains the group ID, the file ID, and *K_i*, along with the file to be encrypted after changes. The CS verifies that the user has the WRITE access to the file from the corresponding ACL. In the case of a valid update request, the CS computes *K* by XORing *Ki* and *K_ i*, encrypts the file, and performs the HMAC calculations. The encrypted file is sent to the user or uploaded to the cloud. *K* is deleted afterward.

3.3 ARCHITECTURE:

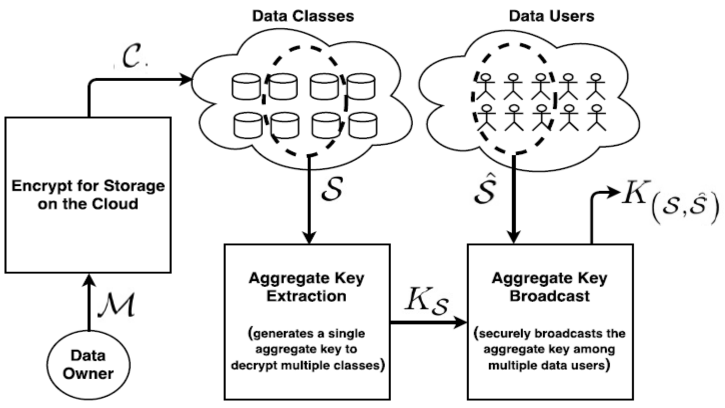
Data Classes
Data Users
C.
Encrypt for Storage on the Cloud
S
Ŝ
K(S,Ŝ)
M
Data Owner
Aggregate Key Extraction
(generates a single aggregate key to decrypt multiple classes)
K_S
Aggregate Key Broadcast
(securely broadcasts the aggregate key among multiple data users)

Figure1: Shows System Architecture

IV. CONCLUSION

In this paper, we addressed an important issue of secure data sharing on untrusted storage. We have proposed an efficiently implementable version of the basic key aggregate cryptosystem in with low overhead cipher texts and aggregate keys using asymmetric bilinear pairings. We have proved our construction to be fully collusion resistant and semantically secure against a non-adaptive adversary under appropriate security assumptions. We have then demonstrated how this construction may be modified to achieve CCA-secure construction, which is, to the best of our knowledge, the first CCA secure KAC construction in the cryptographic literature. We have further demonstrated how the basic KAC framework may be efficiently extended and generalized for securely broadcasting the aggregate key among multiple data users in a real-life data sharing environment.

REFERENCES

[1] Cong Wang, Sherman S.-M. Chow, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for secure cloud storage. Cryptology ePrint Archive, Report 2009/579, 2009.

[2] Sherman SM Chow, Cheng-Kang Chu, Xinyi Huang, Jianying Zhou, and Robert H Deng. Dynamic secure cloud storage with provenance. In Cryptography and Security: From Theory to Applications, pages 442–464. Springer, 2012.

[3] Erik C Shallman. Up in the air: Clarifying cloud storage protections. Intell. Prop. L. Bull., 19:49, 2014.

[4] Cheng-Kang Chu, Sherman SM Chow, Wen-Guey Tzeng, Jianying Zhou, and Robert H Deng. Key-aggregate cryptosystem for scalable data sharing in cloud storage. Parallel and Distributed Systems, IEEE Transactions on, 25(2):468–477, 2014.

[5] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Advances in Cryptology–CRYPTO 2005, pages 258–275. Springer, 2005.

[6] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, “Aggregate and Verifiably Encrypted Signatures from Bilinear Maps,” in ProceedingsofAdvancesinCryptology-EUROCRYPT’03,ser.LNCS, vol. 2656. Springer, 2003, pp. 416–432.

[7] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, “Dynamic and Efficient Key Management for Access Hierarchies,” ACM Transactions on Information and System Security (TISSEC), vol. 12, no. 3, 2009.

[8] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, “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.

[9] F. Guo, Y. Mu, Z. Chen, and L. Xu, “Multi-Identity Single-Key Decryption without Random Oracles,” in Proceedings of Information Security and Cryptology (Inscrypt ’07), ser. LNCS, vol. 4990. Springer, 2007, pp. 384–398.

[10] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-Based Encryption for Fine-Grained Access Control of Encrypted data,” in Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS ’06). ACM, 2006, pp. 89–98.