AUTHENTICATED KEY EXCHANGE PROTOCOLS FOR FILE SHARING SYSTEMS

[1]Subashini T, [2]Preethi P, [3]Priyanka K, [4]Sowmiya R,
[1]Assistant professor, Department of Computer Science and Engineering
Velammal Engineering College,
Chennai, India.
[1]subhatanish@gmail.com ,[2] preethiprakash17@gmail.com,[3] vkpriyanka95@gmail.com,[4]sowmiyavec@gmail.com

ABSTRACT

To implement how to securely, efficiently, and flexibly share data with others in cloud storage.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 their 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.

I. INTRODUCTION

Cloud computing, with the characteristics of intrinsic data sharing and low maintenance, provides a better utilization of resources. In cloud computing, cloud service providers offer an abstraction of infinite storage space for clients to host data. It can help clients reduce their financial overhead of data managements by migrating the local managements system into cloud servers. However, security concerns become the main constraint as we now outsource the storage of data, which is possibly sensitive, to cloud providers. To preserve data privacy, a common approach is to encrypt data files before the clients upload the encrypted data into the cloud. Unfortunately, it is difficult to design a secure and efficient data sharing scheme, especially for dynamic groups in the cloud.The main aim of our project is to sharing the data in cloud. In this paper, we show that how to securely, efficiently and flexibly share data with others in cloud storage. For that we propose Key-Aggregate Cryptosystem which produces cipher text of constant size such that decryption rights can be assigned on them. By combining a set of secret key, we can make a compact single key. By using this compact key, we can send others or can be store in a very limited secure storage. First, owner of the data Setup the public system next Key-Gen algorithm generates a public or master/secret key. By using this key, user can convert plain text to cipher text. Next user will give input as master secret key by Extract function; it will produce output as aggregate decryption key. This generated key is safely sent to the receiver. Then the user with aggregate key can decrypt the cipher text through the use of Decrypt function. 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. 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.

II EXISTING SYSTEM

The existing system of cloud storage bloggers can let their friends view a subset of their private pictures or data; 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 thesedata from the server directly. However, finding an efficient and secure way to share partial data in cloud storage is not trivial. The receiver decrypting the original Message using symmetric key algorithm.

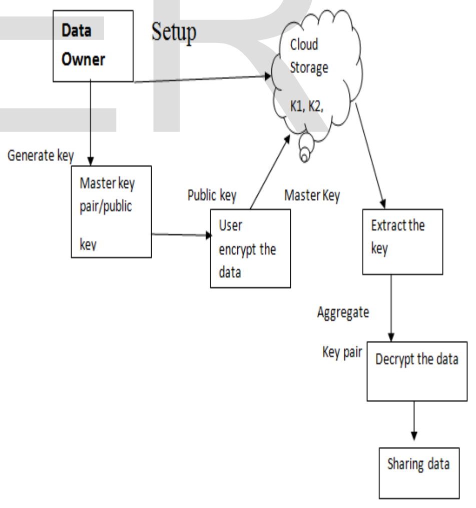
LIMITATIONS:

- Increases the costs of storing and transmitting cipher texts.
- Secret keys are usually stored in the tamper-proof memory, which is relatively expensive.
- This is a versatile approach.
- The costs and complexities involved generally increase with the number of the decryption keys to be shared.

III PROPOSED SYSTEM

In this paper, we make a decryption key as more powerful in the sense that it allows decryption of multiple cipher-texts, without increasing its size. We are introducing a public-key encryption which we call key-aggregate cryptosystem (KAC) they using AES algorithm. In KAC, users encrypt a message not only under a public-key, but also under an identifier of cipher-text called class. That means the cipher-texts 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 cipher-text classes. The sizes of cipher-text, public-key, and master-secret key and aggregate key in our KAC schemes are all of constant size. Thepublic 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 cloud storage. Previous results may achieve a similar property featuring a constant-sizedecryption key, but the classes need to conform to some pre-defined hierarchical relationship. Our work is flexible in thesense that this constraint is eliminated, that is, nospecial relation is required between the classes.

IV SYSTEM-ARCHITECTURE

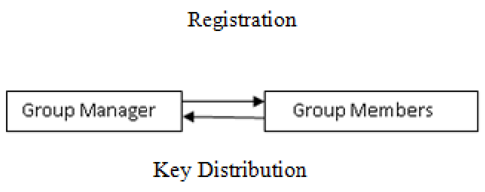


V MODULES

1. User Registration
2. User Revocation
3. File Uploading and Deletion
4. File Access and Traceability

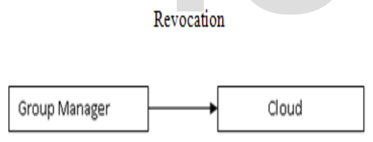
1. User Registration:

For the registration of user with identity ID the group manager randomly selects a number. Then the group manager adds into the group user list which will be used in the traceability phase. After the registration, user obtains a public key which will be used for group signature generation and file decryption.



2. User Revocation:

User revocation is performed by the group manager via a public available. Revocation list, based on which group members can encrypt their data files and ensure the confidentiality against the revoked users. Group manger update the revocation list each day even no user has being revoked in the day. In other words, the others can verify the freshness of the revocation list from the contained current date.



3. File Generation and Deletions:

To store and share a data file in the cloud, a group member performs to getting the revocation list from the cloud. In this step, the member sends the group identity IDgroup as a request to the cloud.Verifying the validity of the received revocation list. File stored in the cloud can be deleted by either the group manager or the data owner.

4. File Access and Traceability:

To access the cloud, a user needs to compute a group signature for his/her authentication. The employed group signature scheme can be regarded asa variant of the short group signature which inherits theinherent unforgetability property, anonymous authentication, and tracking capability. When a data dispute occurs, the tracing operation is performed by the group manager to identify the real identity of the data owner.

VI AES

The Advanced Encryption Standard (AES) is an encryption algorithm for securing sensitive but unclassified material by U.S. Government agencies and, as a likely consequence, may eventually become the de facto encryption standard for commercial transactions in the private sector. (Encryption for the US military and other classified communications is handled by separate, secret algorithms.)In January of 1997, a process was initiated by the National Institute of Standards and Technology (NIST), a unit of the U.S. Commerce Department, to find a more robust replacement for the Data Encryption Standard (DES) and to a lesser degree Triple DES. The specification called for a symmetric algorithm (same key for encryption and decryption) using block encryption (see block cipher) of 128 bits in size, supporting key sizes of 128, 192 and 256 bits, as a minimum. The algorithm was required to be royalty-free for use worldwide and offer security of a sufficient level to protect data for the next 20 to 30 years. It was to be easy to implement in hardware and software, as well as in restricted environments (for example, in a smart card) and offer good defenses against various attack techniques.The entire selection process was fully open to public scrutiny and comment, it being decided that full visibility would ensure the best possible analysis of the designs. In 1998, the NIST selected 15 candidates for the AES, which were then subject to preliminary analysis by the world cryptographic community, including theNational Security Agency. On the basis of this, in August 1999, NIST selected fivealgorithms for more extensive analysis. These were:

- MARS, submitted by a large team from IBM Research
- RC6, submitted by RSA Security
- Rijndael, submitted by two Belgian cryptographers, Joan Daemen and Vincent Rijmen
- Serpent, submitted by Ross Andersen, Eli Biham and Lars Knudsen
- Twofish, submitted by a large team of researchers including Counterpane's respected cryptographer, Bruce Schneier

Implementations of all of the above were tested extensively in ANSI C and Java languages for speed and reliability in such measures as encryption and decryption speeds, key and algorithm set-up time and resistance to various attacks, both in hardware- and software-centric systems. Once again, detailed analysis was provided by the global cryptographic community (including some teams trying to break their own submissions). The end result was that on October 2, 2000, NIST announced that Rijndael had been selected as the proposed standard. On December 6, 2001, the Secretary of Commerce officially approved Federal Information Processing Standard (FIPS) 197, which specifies that all sensitive, unclassified documents will use Rijndael as the Advanced Encryption Standard.Also see cryptography, data recovery agent (DRA).

Explanations

AES is based on a design principle known as a Substitution permutation network. It is fast in both software and hardware. Unlike its predecessor, DES, AES does not use a Feistel network.AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits, whereas Rijndael can be specified withblock and key sizes in any multiple of 32 bits, with a minimum of 128 bits. Theblocksize hasamaximum of 256 bits, but the key-size has no theoretical maximum. AES operates on a 4×4 column-major order matrix of bytes, termedthe *state* (versions of Rijndael with a larger block size have additional columns in the state). Most AES calculations are done in a special finite field.The AES cipher is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of ciphertext. Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key.

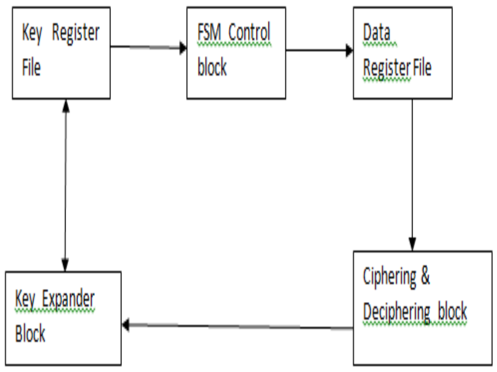
High-level description of the algorithm

1. KeyExpansion—round keys are derived from the cipher key using Rijndael's key schedule.
2. Initial Round
    1. AddRoundKey—each byte of the state is combined with the round key using bitwise xor.
3. Rounds
    1. SubBytes—a non-linear substitution step where each byte is replaced with another according to lookup.
    2. ShiftRows—a transposition step where each row of the state isshifted cyclically a certain number of steps.
    3. MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
    4. AddRoundKey

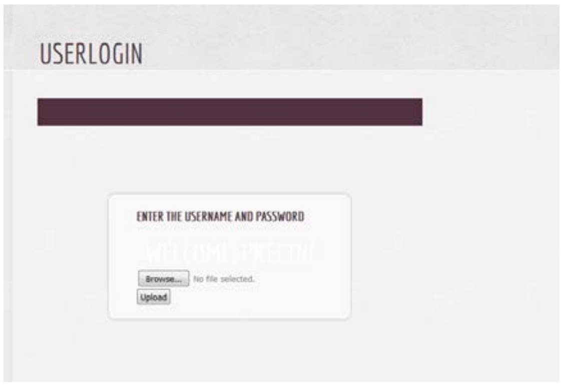
4. Final Round (no MixColumns)
    1. SubBytes
    2. ShiftRows

3.AddRoundKey

DIAGRAM:



VII EXPERIMENT RESULTS



VIII ADVANTAGES

- The delegation of decryption can be efficiently implemented with the aggregate key, which is only of fixed size.
- Number of ciphertext classes is large.
- It is easy to key management for encryption and decryption

IX CONCLUSION

In this paper, we design Key-Aggregate Cryptosystem For Scalable Data Sharing In Cloud Storage. In, a user is able to share data with others in the group without revealing identity privacy to the cloud. Additionally, supports efficient user revocation and new user joining. More specially, efficient user revocation can be achieved through a public revocation list without updating the private keys of the remaining users, and new users can directly decrypt files stored in the cloud before their participation. Moreover, the storage overhead and the encryption computation cost are constant. Extensive analyses show that our proposed scheme satisfies the desired security requirements and guarantees efficiency as well. proposed a cryptographic storage system that enables secure file sharing on untrusted servers, named Plutus. By dividing files into filegroups and encrypting each filegroup with a unique file-block key, the data owner can share the file groups with others through delivering the corresponding lockbox key, where the lockbox key is used to encrypt the file-block keys. However, it brings about a heavy key distribution overhead for large-scale file sharing. Additionally, the file-block key needs to be updated and distributed again for a user revocation.

REFERENCES

[1] M. Abd-El-Malek, W. V. Courtright II, C. Cranor, G. R. Ganger, J. Hendricks, A. J. Klosterman, M. P. Mesnier, M. Prasad, B. Salmon, R. R. Sambasivan, S. Sinnamohideen, J. D. Strunk, E. Thereska, M. Wachs, and J. J. Wylie, “Ursa minor: Versatile cluster-based storage,” in Proc. 4th USENIX Conf. File Storage Technol., Dec. 2005,pp. 59–72.
[2] C. Adams, “The simple public-key GSS-API mechanism (SPKM),”

Internet Eng. Task Force (IETF), RFC 2025, Oct. 1996.

[3] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. Wattenhofer, “FARSITE: Federated, available, and reliable storage for an incompletely trusted environment,” in Proc. 5th Symp. Oper. Syst. Des. Implementation, Dec. 2002, pp. 1–14.

[4] M. K. Aguilera, M. Ji, M. Lillibridge, J. MacCormick, E. Oertli, D. G. Andersen, M. Burrows, T. Mann, and C. A. Thekkath, “Block-level security for network-attached disks,” in Proc. 2nd Int. Conf. File Storage Technol., Mar. 2003, pp. 159–174.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “A view of cloud computing,” Commun. ACM, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[6]Amazon simple storage service (Amazon S3) [Online]. Available: http://aws.amazon.com/s3/, 2014.

[7] M. Bellare, D. Pointcheval, and P. Rogaway, “Authenticated key exchange secure against dictionary attacks,” in Proc. 19th Int. Conf. Theory Appl. Cryptographic Techn., May 2000, pp. 139–155.

[8] D. Boneh, C. Gentry, and B. Waters, “Collusion resistant broadcast encryption with short ciphertexts and private keys,” in Proc. 25th Annu. Int. Conf. Adv. Cryptol., Aug. 2005, pp. 258–275.

[9] B. Callaghan, B. Pawlowski, and P. Staubach, “NFS version 3 protocol specification,“ Internet Eng. Task Force (IETF), RFC 1813, Jun. 1995.

[10] R. Canetti and H. Krawczyk, “Analysis of key-exchange protocols and their use for building securechannels,” in Proc. Int. Conf. Theory Appl. Cryptographic Techn.: Adv. Cryptology, May 2001,