ABSTRACT

In the cloud environment data owner uploads the information to the Remote Cloud Server for data access. Data owner appoints members for data utility. Members have to get permission for the data updation from the data owner. Members will have their user name, keys, and group key for access. In the social network there is no adequate security. To enhance the security for data in that situation, introducing new technology here that is either if existing member is removed from that group, Group Key is automatically changed and updated to all the members of that group. There is change in group key in case of new member is added in that group also. Member can resign from the group by themselves or data owner can terminate the member or can be cloud terminates the member in case of misbehavior.

KEYWORDS:

INTRODUCTION

Cloud computing is an innovative service mode. It enables users to get almost unlimited computing power and abundant a variety of information services from internet. They are distributed computing, parallel computing and grid computational evolution. This kind of new pattern refers the integration and expansion to the IT infrastructure, through the network to the required resources (hardware, platform, and software), virtual integration into a reliable and high performance computing platform. In cloud computing, all users data are stored in the cloud resources Nodes. The results distribute to the user through the network when the user needed.

Although cloud computing has become a mature service model, and have large commercial, cloud computing is still facing many problems. In 2009, the well-known research institutions IDC release an IT report that cloud computing service is facing three major challenges: safety, stability and performance issue. Including the security problem concerns the most. A real cloud computing security incidents have profound reveals the urgency of cloud security issues, such as the 2009 Microsoft SIDEKICK service was interrupted for a week, a large number of users cannot access to their email and other personal data. More seriously, due to the technical personnel not to make backups of their data, resulting in Microsoft cannot recover data. Although the cloud storage service can realize multi copy of fault tolerance and backup automatically, it is also cannot do guarantee 100% security. In 2009, Google expose to the risk of mixed date by unauthorized accessing in cloud platform and unauthorized sharing user's spreadsheets date and document date.

In the most basic cloud-service model & according to the IETF (Internet Engineering Task Force), providers of IaaS offer computers – physical or (more often) virtual machines and other resources. IaaS clouds often offer additional resources such as a virtual-machine disk image library, raw block storage, and file or
object storage, firewalls, load balancers, IP addresses, virtual local area networks (VLANs), and software bundles.

Encryption is the conversion of data into a form, called a cipher text, that cannot be easily understood by unauthorized people. Decryption is the process of converting encrypted data back into its original form, so it can be understood. The use of encryption/decryption is as old as the art of communication. In wartime, a cipher, often incorrectly called a code, can be employed to keep the enemy from obtaining the contents of transmissions. Simple ciphers include the substitution of letters for numbers, the rotation of letters in the alphabet, and the "scrambling" of voice signals by inverting the sideband frequencies. More complex ciphers work according to sophisticated computer algorithms that rearranges the data bits in digital signals.

A group key management (GKM) protocol supports protected communication between members of a secure group. A secure group is a collection of principals, called members, who may be senders, receivers, or both receivers and senders to other members of the group. Group membership may vary over time. A group key management protocol helps to ensure that only members of a secure group can gain access to group data (by gaining access to group keys) and can authenticate group data. The goal of a group key management protocol is to provide legitimate group members with the up-to-date cryptographic state they need for secrecy and authentication. Key management is the management of cryptographic keys in a cryptosystem. This includes dealing with the generation, exchange, storage, use, and replacement of keys. It includes cryptographic protocol design, key servers, user procedures, and other relevant protocols. We proposed scheme greatly reduces the user’s computation and storage overhead and makes full use of cloud server to achieve an efficient group key management for the cryptographic cloud storage applications

I. Related Methods:

[1] We introduce a model for provable data possession (PDP) that allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof. The challenge/response protocol transmits a small, constant amount of data, which minimizes network communication.

[2] We present two provably-secure PDP schemes that are more efficient than previous solutions, even when compared with schemes that achieve weaker guarantees. In particular, the overhead at the server is low (or even constant), as opposed to linear in the size of the data. Experiments using our implementation verify the practicality of PDP and reveal that the performance of PDP is bounded by disk I/O and not by cryptographic computation.

[2] Proxy re-encryption allows a proxy to transform a ciphertext computed under Alice’s public key into one that can be opened by Bob’s secret key. There are many useful applications of this primitive. For instance, Alice might wish to temporarily forward encrypted email to her colleague Bob, without giving him her secret key. In this case, Alice the delegator could designate a proxy to re-encrypt her incoming mail into a format that Bob the delegatee can decrypt using his own secret key. Alice could simply provide her secret key to the proxy, but this requires an unrealistic level of trust in the proxy.

[2] It presents several efficient proxy re-encryption schemes that offer security improvements over earlier approaches. The primary advantage of our schemes is that they are unidirectional (i.e., Alice can delegate to Bob without Bob having to delegate back), and do not require delegators to reveal all of their secret key to anyone or even interact with the delegatee in order to allow a proxy to re-encrypt their ciphertexts. In our schemes, only a limited amount of trust is placed in the proxy.

[2] For example, it is not able to decrypt the ciphertexts it re-encrypts, and we prove our schemes secure even when the proxy publishes all the re-encryption information it knows. This enables a number of applications that would not be practical if the proxy needed to be fully trusted.

[3] Cloud Computing has been envisioned as the next-generation architecture of IT Enterprise. It moves the application software and databases to the centralized large data centers, where the management of the data and services may not be fully trustworthy. This unique paradigm brings about many new security challenges, which have not been well understood. This work studies the problem of ensuring the integrity of data storage in Cloud Computing.

[3] The task of allowing a third party auditor (TPA), on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. The introduction of TPA eliminates the involvement of client through the auditing of whether his data stored in the cloud is indeed intact, which can be important in achieving economies of scale for Cloud Computing. The support for data dynamics via the most general forms of data operation, such as block modification, insertion and deletion, is also a significant step toward practicality, since services in Cloud Computing are not limited to archive or backup data only. While prior works on ensuring remote data integrity often lack the support of either public verifiability or dynamic data operations, this paper achieves both.
It first identifies the difficulties and potential security problems of direct extensions with fully dynamic data updates from prior works and then show how to construct an elegant verification scheme for seamless integration of these two salient features in our protocol design. In particular, to achieve efficient data dynamics, we improve the Proof of Retrievability model [1] by manipulating the classic Merkle Hash Tree (MHT) construction for block tag authentication. Extensive security and performance analysis show that the proposed scheme is highly efficient and provably secure.

With data services in the cloud, users can easily modify and share data as a group. To ensure data integrity can be audited publicly, users need to compute signatures on all the blocks in shared data. Different blocks are signed by different users due to data modifications performed by different users. For security reasons, once a user is revoked from the group, the blocks, which were previously signed by this revoked user must be re-signed by an existing user.

The straightforward method, which allows an existing user to download the corresponding part of shared data and re-sign it during user revocation, is inefficient due to the large size of shared data in the cloud. In this project, it propose a novel public auditing mechanism for the integrity of shared data with efficient user revocation in mind. By utilizing proxy re-signatures, we allow the cloud to re-sign blocks on behalf of existing users during user revocation, so that existing users do not need to download and re-sign blocks by themselves. In addition, a public verifier is always able to audit the integrity of shared data without retrieving the entire data from the cloud, even if some part of shared data has been re-signed by the cloud. Experimental results show that our mechanism can significantly improve the efficiency of user revocation.

II. Proposed System:

A Time-based Group Key Management algorithm for cryptographic cloud storage applications, which uses the proxy re-encryption algorithm to transfer major computing task of the group key management to the cloud server. We present our lightweight end-to-end key management protocol.

The proposed TGKM scheme greatly reduces the user’s computation and storage overhead and makes full use of cloud server to achieve an efficient group key management for the cryptographic cloud storage applications.

We introduce a key seed mechanism to generate a time-based dynamic group key which effectively strengthens the cloud data security. Our security analysis and performance evaluations both show that the proposed TGKM scheme is a secure and efficient group key management protocol for the cloud storage applications with low overheads of computation and communication.

A. Architecture:

B. Key Management Protocol:

Symmetric encryption algorithms can be divided into stream ciphers and block ciphers. Stream ciphers encrypt a single bit of plaintext at a time, whereas block ciphers take a number of bits (typically 64 bits in modern ciphers), and encrypt them as a single unit.

1. Asymmetric encryption algorithms (public key algorithms) use different keys for encryption and decryption, and the decryption key cannot (practically) be derived from the encryption key.
2. Public key methods are important because they can be used for transmitting encryption keys or other data securely even when the parties have no opportunity to agree on a secret key in private.
3. A new lightweight key management protocol. This protocol is based on collaboration to establish a secured communication channel between a highly resource constrained node and a remote entity (i.e. server).
The secured channel allows the constrained node to transmit captured data while ensuring confidentiality and authentication.

Our protocol offloads highly consuming cryptographic primitives to third parties which are not necessarily trusted. Consequently, constrained nodes obtain assistance from more powerful entities in order to securely establish a shared secret with any remote entity. The secure channel allows the constrained node to transmit captured data while ensuring confidentiality and authentication. To achieve this goal, we propose offloading highly consuming cryptographic primitives to third parties.

C. Proxy Re-Encryption:
A proxy re-encryption is generally used when one party, say Bob, wants to reveal the contents of messages sent to him and encrypted with his public key to a third party, Chris, without revealing his private key to Chris. Bob does not want the proxy to be able to read the contents of his messages.
1. Bob could designate a proxy to re-encrypt one of his messages that is to be sent to Chris. This generates a new key that Chris can use to decrypt the message.
2. If Alice sends Chris a message that was encrypted under Bob's key, the proxy will alter the message, allowing Chris to decrypt it.
3. This method allows for a number of applications such as e-mail forwarding, law-enforcement monitoring, and content distribution.

A weaker re-encryption scheme is one in which the proxy possesses both parties' keys simultaneously. One key decrypts a plaintext, while the other encrypts it. Since the goal of many proxy re-encryption schemes is to avoid revealing either of the keys or the underlying plaintext to the proxy, this method is not ideal.

D. Data Blind Storage:
A blind storage system is built on the cloud server to support adding, updating and deleting documents and concealing the access pattern of the search user from the cloud server.
1. In the blind storage system, all documents are divided into fixed-size blocks. These blocks are indexed by a sequence of random integers generated by a document-related seed.
2. In the view of a cloud server, it can only see the blocks of encrypted documents uploaded and downloaded.

The blind storage system leaks little information to the cloud server. Specifically, the cloud server does not know which blocks are of the same document, even the total number of the documents and the size of each document. Moreover, all the documents and index can be stored in the blind storage system to achieve a searchable encryption scheme.

E. Time-Based Group Key Management Algorithm:
The cloud offers data storage and sharing services to users. It follows our proposed protocol in general, but also tries to find out as much secret information as possible. TGKM uses two steps to share the GKfile in the authenticated group users.
1. GKfile is not fixed in various phases, so even GKfile is disclosed during any period, other GKfile is still secure. In the first step,
2. TGKM shares the key seed Skey based on proxy re-encryption mechanism in the authorized group users
   The key can compute file encryption group key. Then data owner and authorized users further compute time-based group keys from Skey to enable forward security and back ward security

F. Data Sharing To Group Users:
When the data owner wants to upload a shared file, he gets the current time as the timestamp and computes encrypted file group key in the time phase by key seed, and encrypts file by key. Finally, the data owner uploads the encrypted file and the timestamp to the cloud server.
1. Cryptographic cloud storage server builds the authorized tree to share resources in the authorized users.
   Each data owner has an authorized tree to describe the shared relationship of his resources.
2. In the authorized tree, the root node stores key seed which is encrypted by the data owner’s public key, each the child node presents a user group which stores key seed encrypted by the group public key, and every leaf node presents an authorized user stores the key seed encrypted by user’s public key.

G. Forward Security And Backward Security:
We survey the development of forward security and relate it to other concepts and trends in modern cryptography.
1. Ordinary digital signatures have an inherent weakness: if the secret key is leaked, then all signatures, even the ones generated before the leak, are no longer trustworthy.
2. Forward-secure digital signatures were proposed to address this weakness: they ensure that past signatures remain secure even if the current secret key is leaked.
3. Similarly for the case of ordinary encryption, adversary that successfully exposed a secret key is typically able to expose even the old messages sent long before exposure. Forward-secure encryption ensures that the past messages are protected even if the current secret key is exposed. We discuss forward security as a special case of key-evolving cryptography.

**Conclusion:**

In this paper, we propose a novel time-based group key management (TGKM) in cryptographic cloud storage. TGKM transfers much workload of key management to the cloud and prevents the cloud to master any group key. Furthermore, to enhance the scalability of TGKM with dynamic group, we propose the key seed mechanism to enable a time based key management. Even if an attacker gets a file encryption key $GK_{file}$, he still can’t decrypt any other files out of the time window. Through experiments, we find TGKM can greatly improve the efficiency of key management and can be applied to the cryptographic cloud storage applications.

**REFERENCES**