RESEARCH ARTICLE OPEN ACCESS

Secure Multiparty Computation on Multiple Clouds

[1]K.Ravikumar, [2]S. Thamizharasi
[1]Asst.professor, Dept.of.Computer science, Tamil University (Established by the Govt.of.Tamilnadu), Thanjavur.
[2]Research Scholar, Dept.of.Computer Science, Tamil University, Thanjavur.

----------**********----------

Abstract:

Cloud Computing is a model for delivering information technology services in which resources are retrieved from the internet through web-based tools and applications, rather than a direct connection to a server. A distributed cloud storage system contains collection of storage servers which continuously provides storage services to the third party or clients. The distributed cloud storage system must maintain the data confidentiality over the stored data in the storage server. This can be done by encoding over the encrypted data on the storage server. The distributed cloud storage system also maintains the robustness and functionality over the encoded and encrypted data. The distributed cloud storage system support data forwarding operations over encoded and encrypted messages.

.

Keywords **— Proxy Encryption, Cloud.**

----------**********----------

I. INTRODUCTION

Storing data in a third party's cloud system causes serious concern over data confidentiality and there are some functionality restrictions on the storage system. We focus on designing a cloud storage system for data robustness, confidentiality, and improve the functionality of the storage server. These all can be achieved through a threshold proxy re-encryption scheme and integrate it with a decentralized erasure code such that a secure distributed storage system is formulated.

2.PROPOSED SYSTEM TECHNIQUES

Proxy Re-Encryption Schemes:

In a proxy re-encryption scheme, a proxy server can transfer a cipher text under a public key PKA to a new one under another public key PKB by using the re-encryption key RKA→B. The server does not know the plaintext during transformation. When user A wants to store messages are first encrypted by the owner and then stored in a storage server. When a user wants to share his messages, he sends a re-encryption key to the storage server. The storage server re-encrypts the encrypted messages for the authorized user. Thus, their system has data confidentiality and supports the data forwarding function.

System setup:

Login/Register
Key generator (PK and SK)
Share to Key server

Data storage:

Storing data in the storage server

Data forwarding:

Forward data to another user

Data retrieval:

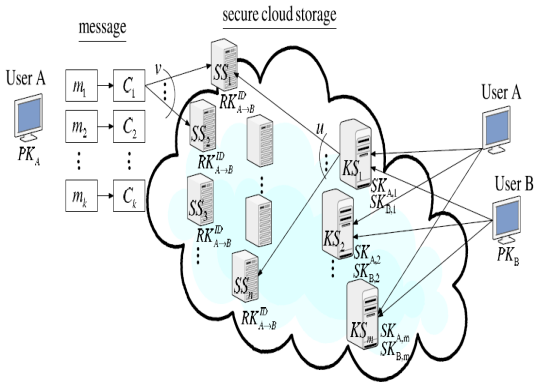
By data owner
Received data

Advantages

The storage server can able to transfer the stored user's data into another user by analyzing the stored user ID and our system is distributed storage system it can perform independently.

Our cloud storage system maintains the robustness, confidentiality and functionality.

To store data on the storage server performs encryption and encoding so that it maintains the data confidentiality.



3. MODULES DIAGRAM AND DESCRIPTION

System setup:

- **Login/Register:**

In Login Form module presents users a form with username and Password fields. If the user enters a valid username/password combination they will be granted to access data. If the user enter invalid username and password that user will be considered as unauthorized user and denied access to that user.

If he is a new user he needs to enter the required data to register the form and the data will be stored in server for future authentication purpose.

Key generator (PK and SK)

The key generator generates the public key and secret key for the new user. These public and private or secret keys are used to encrypt and decrypt the messages for data confidential purpose. Usually public key is used to encrypt the data and secret key or private key is used to decrypt the cipher text to get the original plain text.

Share to Key server

The user has to share his secret key to randomly chosen key server. This secret key is used to decrypt the encoded message when the authenticated person wants to share his data or retrieve his data.

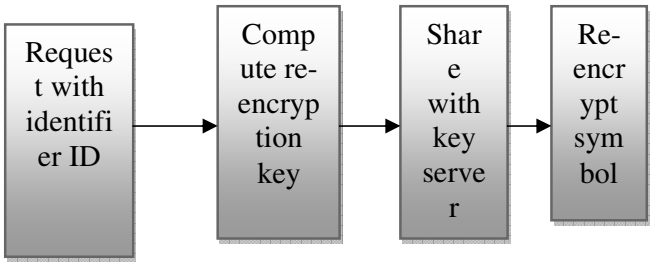
4. DATA STORAGE

- **Storing data in the storage server**

In the data storage phase, user A encrypts his message M and dispatches it to storage servers. A message M is decomposed into k blocks m_1, m_2, ……., m_k and has an identifier ID. User A encrypts each block m_i into a cipher text C_i and sends it to v randomly chosen storage servers. Upon receiving cipher texts from a user, each storage server linearly combines them with randomly chosen coefficients into a codeword symbol and stores it.

5. FORWARD DATA TO ANOTHER USER

When user A wants to forwards his encrypted message with an identifier ID stored in storage servers to user B such that B can decrypt the forwarded message by his secret key. To do so, A uses his secret key SK_A and B's public key PK_B to compute a re-encryption key $RK^{ID}_{A\to B}$ and then sends $RK^{ID}_{A\to B}$ to all storage servers. Each storage server uses the re-encryption key to re-encrypt its codeword symbol for later retrieval requests by B. The re-encrypted codeword symbol is the combination of cipher texts under B's public key.



By data owner

The first case is that a user A retrieves his own message. When user A wants to retrieve the message with the identifier ID, he informs all key servers with the identity token. A key server first retrieves original codeword symbols from randomly chosen storage servers and then performs partial decryption on every retrieved original codeword symbol. The key server sends the partially decrypted codeword symbols to user A. Then user A applies decryption on collected cipher text to recover the blocks and then combines them to get original data.

- **Received data**

The second case is that a user B retrieves a message forwarded to him. User B sends the request to key server with identifier ID. After authenticating the user B key server decode the re-

encrypted codeword symbol. The key server sends the partially decrypted codeword symbols to user A. Then user A applies decryption on collected cipher text to recover the blocks and then combines them to get original data.

6. GIVEN INPUT AND EXPECTED OUTPUT

System setup:

Login/Register

Input: User has to give required data to login or register to access the cloud storage

Output: Storage system permits them to access the cloud storage

Key generator (PK and SK)

Input: User details

Output: Generate the SK and PK

Share to Key server

Input: SK to the Key server

Output: SK is stored in Key server

Data storage:

Storing data in the storage server

Input: Encrypted message to the Storage Server

Output: Encode and store it

Data forwarding:

Forward data to another user

Input: Request with identification ID

Output: Apply Re-encryption scheme and store it

Data retrieval:

By data owner

Input: Request with Identification ID

Output: Original Message

7. CONCLUSION

The threshold proxy re-encryption scheme supports encoding, forwarding, and partial decryption operations in a distributed way. To decrypt a message of k blocks that are encrypted and encoded to n codeword symbols, each key server only has to partially decrypt two codeword symbols in our system. By using the threshold proxy re-encryption scheme, we present a secure cloud storage system that provides secure data storage and secure data forwarding functionality in a decentralized structure. Moreover, each storage server independently performs encoding and re-encryption and each key server independently perform partial decryption.

8. FUTURE ENHANCEMENT

Our Key server performs the main role in our distributed storage system. This Key server performs the important role key management. But our proposed system doesn't provide any security over this Key Server. The attacker or intruder can attack the key server to get the secret key because there is no security provided to the Secret Key. So as a future work we focus on key server for giving more secure to our storage system. To overcome this problem we are going apply the encoding over the secret key before it store in the Key server. The Key server can decode this encoded Secret Key.

9.REFERENCES

1. 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," Proc. Fifth Symp. Operating System Design and Implementation (OSDI) , pp. 1-14, 2002.
2. M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable Secure File Sharing on Untrusted Storage," Proc. Second USENIX Conf. File and Storage Technologies (FAST), pp. 29- 42, 2003.
3. H.-Y. Lin and W.-G. Tzeng, "A Secure Decentralized Erasure Code for Distributed Network Storage," IEEE Trans. Parallel and Distributed Systems, vol. 21, no. 11, pp. 1586-1594, Nov. 2010.
4. R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G.M. Voelker, "Total Recall: System Support for Automated Availability Management," Proc. First Symp. Networked Systems Design and Implementation (NSDI), pp. 337-350, 2004.
5. A.G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Ubiquitous Access to Distributed Data in Large-Scale Sensor Networks through Decentralized Erasure Codes," Proc. Fourth Int'l Symp. Information Processing in Sensor Networks (IPSN), pp. 111-117, 2005.

6. Q. Tang, "Type-Based Proxy Re-Encryption and Its Construction," Proc. Ninth Int'l Conf. Cryptology in India: Progress in Cryptology (INDOCRYPT), pp. 130-144, 2008.
7. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS), pp. 598-609, 2007.
8. G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. Fourth Int'l Conf. Security and Privacy in Comm. Netowrks (SecureComm), pp. 1-10, 2008.
9. H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. 14th Int'l Conf. Theory and Application of Cryptology and Information Security (ASIACRYPT), pp. 90-107, 2008.
10. K.D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Proc. 16th ACM Conf. Computer and Comm. Security (CCS), pp. 187-198, 2009.