International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 25521-25529

http://www.ripublication.com

Secure Multi Owner Data Sharing In Cloud

Abdul Ahad[1], G.Sarma[2], V.Sai Mounica[3], P.CH.S.B.Sruthi[4], D.Sai Sirisha[5]

Abstract

With the character of low maintenance, cloud computing provides an economical and efficient solution for sharing group resource among cloud users. Unfortunately, sharing data in a multi-owner manner while preserving data and identity privacy from an untrusted cloud is still a challenging issue, due to the frequent change of the membership. In this paper, we propose a secure multi owner data sharing scheme, named Mona, for dynamic groups in the cloud. By leveraging group signature and dynamic broadcast encryption techniques, any cloud user can anonymously share data with others. Meanwhile, the storage overhead and encryption computation cost of our scheme are independent with the number of revoked users. In addition, we analyze the security of our scheme with rigorous proofs, and demonstrate the efficiency of our scheme in experiments.

Keywords: Data Sharing, Signature Verification, Security

I. INTRODUCTION

CLOUD computing is recognized as an alternative to traditional information technology due to its intrinsic resource-sharing and low-maintenance characteristics. In cloud computing, the cloud service providers (CSPs), such as Amazon, are able to deliver various services to cloud users with the help of powerful datacenters. By migrating the local data management systems into cloud servers, users can enjoy high-quality services and save significant investments on their local infrastructures. One of the most fundamental services offered by cloud providers is data storage. Let us consider a practical data application. A company allows its staffs in the same group or department to store and share files in the cloud. By utilizing the cloud, the staffs can be completely released from the troublesome local data storage and maintenance. However, it also poses a significant risk to the confidentiality of those stored files. Specifically, the cloud servers managed by cloud providers are not fully trusted by users while the data files stored in the cloud may be sensitive and confidential, such as business plans. To preserve data privacy, a basic solution is to encrypt data files, and then upload the encrypted data into the cloud. Unfortunately, designing an efficient

and secure data sharing scheme for groups in the cloud is not an easy task due to the following challenging issues.

First, identity privacy is one of the most significant obstacles for the wide deployment of cloud computing. Without the guarantee of identity privacy, users may be unwilling to join in cloud computing systems because their real identities could be easily disclosed to cloud providers and attackers. On the other hand, unconditional identity privacy may incur the abuse of privacy. For example, a misbehaved staff can deceive others in the company by sharing false files without being traceable. Therefore, traceability, which enables the group manager (e.g., a company manager) to reveal the real identity of a user, is also highly desirable. Second, it is highly recommended that any member in a group should be able to fully enjoy the data storing and sharing services provided by the cloud, which is defined as the multiple-owner manner. Compared with the single-owner manner, where only the group manager can store and modify data in the cloud, the multiple-owner manner is more flexible in practical applications. More concretely, each user in the group is able to not only read data, but also modify his/her part of data in the entire data file shared by the company. Last but not least, groups are normally dynamic in practice, e.g., new staff participation and current employee revocation in a company. The changes of membership make secure data sharing extremely difficult. On one hand, the anonymous system challenges new granted users to learn the content of data files stored before their participation, because it is impossible for new granted users to contact with anonymous data owners, and obtain the corresponding decryption keys. On the other hand, an efficient membership revocation mechanism without updating the secret keys of the remaining users is also desired to minimize the complexity of key management.

II. REQUIREMENT ANALYSIS

SOFTWARE COMPONENTS

Operating System : Windows XP.
Coding Language : JAVA/JSP
Database : MYSQL
Server : Apache Tomcat

HARDWARE COMPONENTS

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Monitor : 15 inch VGA Colour.
- Mouse : Logitech Mouse.
- Ram : 512 MB
- Keyboard : Standard Keyboard

ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a

de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

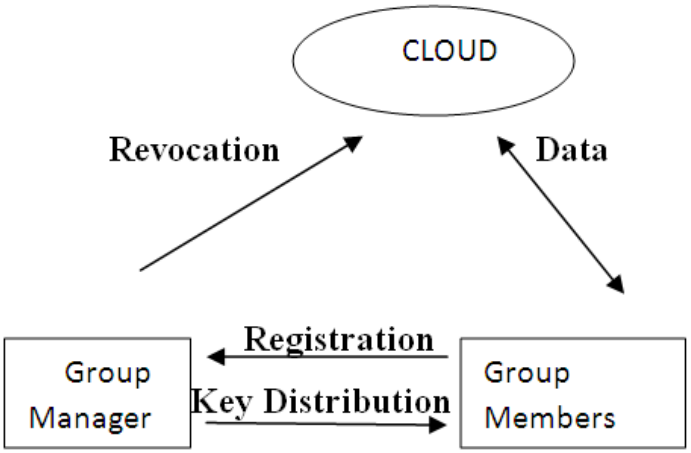
JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

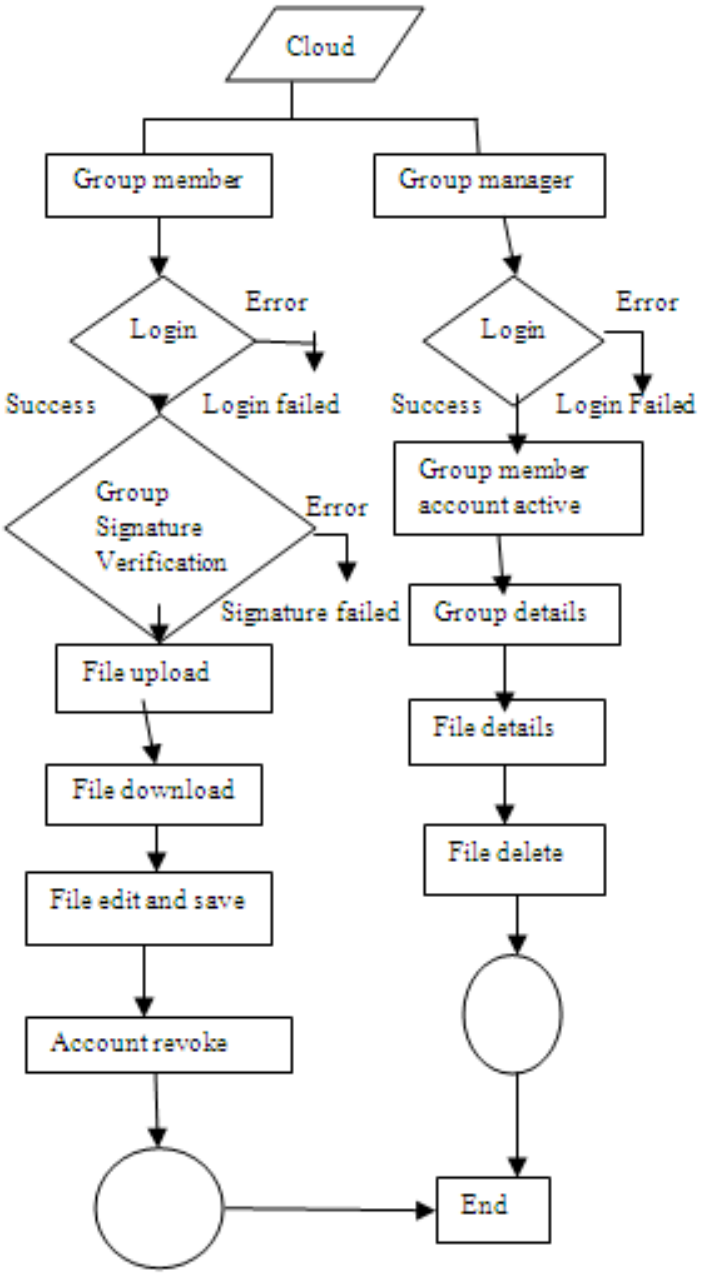
To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after. The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

III. DESIGN



- Group manager can access everything
- New member first registers, then the manager has the power to activate or to deactivate
- If the manager activates, then the member can login but the person will get a signature to the mail, with that signature he can access
- Here we can update or download the file
- If there is any missuse of file by the member, he has the power to deactivate that member
- Security is more



IV. ANALYSIS

EXISTING SYSTEM:

To preserve data privacy, a basic solution is to encrypt data files, and then upload the encrypted data into the cloud. Unfortunately, designing an efficient and secure data sharing scheme for groups in the cloud is not an easy task.

In the existing System data owners store the encrypted data files in untrusted storage and distribute the corresponding decryption keys only to authorized users. Thus, unauthorized users as well as storage servers cannot learn the content of the

data files because they have no knowledge of the decryption keys. However, the complexities of user participation and revocation in these schemes are linearly increasing with the number of data owners and the number of revoked users, respectively.

DISADVANTAGES OF EXISTING SYSTEM:

- In the existing Systems, identity privacy is one of the most significant obstacles for the wide deployment of cloud computing. Without the guarantee of identity privacy, users may be unwilling to join in cloud computing systems because their real identities could be easily disclosed to cloud providers and attackers. On the other hand, unconditional identity privacy may incur the abuse of privacy. For example, a misbehaved staff can deceive others in the company by sharing false files without being traceable.
- Only the group manager can store and modify data in the cloud
- The changes of membership make secure data sharing extremely difficult the issue of user revocation is not addressed

PROPOSED SYSTEM:

1. We propose a secure multi-owner data sharing scheme. It implies that any user in the group can securely share data with others by the untrusted cloud.
2. Our proposed scheme is able to support dynamic groups efficiently. Specifically, new granted users can directly decrypt data files uploaded before their participation without contacting with data owners. User revocation can be easily achieved through a novel revocation list without updating the secret keys of the remaining users. The size and computation overhead of encryption are constant and independent with the number of revoked users.
3. We provide secure and privacy-preserving access control to users, which guarantees any member in a group to anonymously utilize the cloud resource. Moreover, the real identities of data owners can be revealed by the group manager when disputes occur.
4. We provide rigorous security analysis, and perform extensive simulations to demonstrate the efficiency of our scheme in terms of storage and computation overhead.

ADVANTAGES OF PROPOSED SYSTEM:

- ✓ Any user in the group can store and share data files with others by the cloud.
- ✓ The encryption complexity and size of ciphertexts are independent with the number of revoked users in the system.
- ✓ User revocation can be achieved without updating the private keys of the remaining users.
- ✓ A new user can directly decrypt the files stored in the cloud before his participation.

V. TESTING

VI. FEATURES

Any user in the group can store and share data files with others by the cloud

The encryption complexity and size of ciphertexts are independent with the number of revoked users in the system

User revocation can be achieved without updating the private keys of the remaining users.

A new user can directly decrypt the files stored in the cloud before his participation

VII. APPLICATIONS

- Railways
- Collegues
- Offices

VIII. CONCLUSION

Here mona is used to share the data securely with the signature key

IX. FUTURE SCOPE

Files stored on the untrusted server include two parts: file metadata and file data. The file metadata implies the access control information including a series of encrypted key blocks, each of which is encrypted under the public key of authorized users. Thus, the size of the file metadata is proportional to the number of authorized users. The user revocation in the scheme is an intractable issue especially for large-scale sharing, since the file metadata needs to be updated. In their extension version, the NNL construction is used for efficient key revocation. However, when a new user joins the group, the private key of each user in an NNL system needs to be recomputed, which may limit the application for dynamic groups. Another concern is that the computation overhead of encryption linearly increases with the sharing scale.

References

[1] 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," Comm. ACM, vol. 53, no. 4, pp. 50-58, Apr. 2010.

[2] S. Kamara and K. Lauter, "Cryptographic Cloud Storage," Proc. Int'l Conf. Financial Cryptography and Data Security (FC), pp. 136-149, Jan. 2010.

[3] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing," Proc. IEEE INFOCOM, pp. 534-542, 2010.

[4] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable Secure File Sharing on Untrusted Storage," Proc. USENIX Conf. File and Storage Technologies, pp. 29-42, 2003.

[5] E. Goh, H. Shacham, N. Modadugu, and D. Boneh, "Sirius: Securing Remote Untrusted Storage," Proc. Network and Distributed Systems Security Symp. (NDSS), pp. 131-145, 2003.

[6] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," Proc. Network and Distributed Systems Security Symp. (NDSS), pp. 29-43, 2005.

[7] R. Lu, X. Lin, X. Liang, and X. Shen, "Secure Provenance: The Essential of Bread and Butter of Data Forensics in Cloud Computing," Proc. ACM Symp. Information, Computer and Comm. Security, pp. 282-292, 2010.

[8] B. Waters, "Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization," Proc. Int'l Conf. Practice and Theory in Public Key Cryptography Conf. Public Key Cryptography, http://eprint.iacr.org/2008/290.pdf, 2008.

[9] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," Proc. ACM Conf. Computer and Comm. Security (CCS), pp. 89-98, 2006.

[10] D. Naor, M. Naor, and J.B. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers," Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO), pp. 41-62, 2001.

[11] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," Proc. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO), pp. 213-229, 2001.

[12] D. Boneh, X. Boyen, and H. Shacham, "Short Group Signature," Proc. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO), pp. 41-55, 2004.

[13] D. Boneh, X. Boyen, and E. Goh, "Hierarchical Identity Based Encryption with Constant Size Ciphertext," Proc. Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT), pp. 440-456, 2005.

[14] C. Delerablee, P. Paillier, and D. Pointcheval, "Fully Collusion Secure Dynamic Broadcast Encryption with Constant-Size Ciphertexts or Decryption Keys," Proc. First Int'l Conf. Pairing-Based Cryptography, pp. 39-59, 2007.

[15] D. Chaum and E. van Heyst, "Group Signatures," Proc. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT), pp. 257-265, 1991.