International Journal of Research in Engineering, Science and Management
Volume-2, Issue-3, March-2019
www.ijresm.com | ISSN (Online): 2581-5792

# A Robust Identity based System for Secure Data Sharing in Cloud Computing

Niranjini[1], S. Balanarayanan[2], A. Ajay Kumar[3], H. Arvind[4], M. P. Gokul Balaji[5]
[1]*Assistant Professor, Dept. of Computer Science and Engg., Sri Eshwar College of Engg., Coimbatore, India*
[2,3,4,5]*Student, Dept. of Computer Science and Engg., Sri Eshwar College of Engg., Coimbatore, India*

***Abstract*: With the popularity of cloud computing, mobile devices can store or retrieve personal data from anywhere at any time. Consequently, the data security problem in mobile cloud becomes more severe and prevents further development of mobile cloud. There are related studies that have been conducted to improve the cloud security. However, most of them are not suitable for mobile cloud since mobile devices only have limited computing resources and power. Solutions with low computational overhead are in need for mobile cloud applications. In this paper, we propose a secure data sharing scheme (SDSS) for mobile cloud computing. It adopts IBE (Identity Based Encryption), an access control technology used in normal cloud environment, but changes the structure of access control tree to make it suitable for mobile cloud environments. SDSS moves a large portion of the computational intensive access control tree transformation in IBE from mobile devices to external proxy servers. Furthermore, to reduce the user revocation cost, it introduces attribute description fields to implement lazy-revocation, which is a thorny issue in program based IBE systems. The experimental results show that SDSS can effectively reduce the overhead on the mobile device side when users are sharing data in mobile cloud environments.**

***Keywords*: Java1.7, jsp, servlet, My SQL, Glass Fish.**

## 1. Introduction

Various cloud mobile applications have been widely used. In these applications, people (data owners) can upload their photos, videos, documents and other files to the cloud and share these data with other people (data users) they like to share. CSPs also provide data management functionality for data owners. Personal data files are important to users, so data admins are allowed to choose whether to make their data files public or can only be shared with specific data users. Clearly, data privacy of the personal important data is a big concern for many data owners. The state-of-the-art privilege management/access control mechanisms provided by the CSP are not sufficient. They cannot meet all the requirements of data owners. First, when people upload their data files onto the cloud, they are leaving the data in a place where is out of their control, and the CSP may spy on user data for its commercial interests and/or other reasons. Second, people have to send password to each data user if they only want to share the encrypted data with certain users, which is very difficult to use. To simplify the privilege management, the data owner can divide data users into different groups and send password to the groups which they want something to share the data. However, this approach requires fine-grained access control. In both cases, password management is a big issue.  Apparently, to solve the above problems, personal important data should be encrypted before uploaded onto the cloud so that the data is secure against the CSP. However, the data encryption brings new problems. How to provide efficient access control mechanism on ciphertext decryption so that only the authorized users can access the plaintext data is challenging. In addition, system must offer data owners effective user privilege management capability, so they can grant or revoke data access privileges easily on the data users. There have been related researches on the issue of data access control over ciphertext. In these researches, they have the following common assumptions. First, the CSP is considered honest and curious. Second, all the important data are encrypted before uploaded to the Cloud.

Third, user authorization on particular data is achieved through encryption/decryption key distribution. In general, we can divide these approaches into four categories: simple cipher text access control, hierarchical access control, access control based on fully homomorphic encryption and access control based on attribute-based encryption (ABE). All these proposals are designed for non-mobile cloud environment. It consumes huge amount of storage and computation resources, which are not available for mobile devices. According to the experimental results in, the basic ABE operations take much longer time on mobile devices than laptop or desktop computers. It is at least 27 times longer to execute on a smart phone than a personal computer (PC). It means that an encryption operation which takes 1 minute on a Personal computer will take a half an hour to finish on a mobile device.

Furthermore, current solutions don't solve the user privilege change problem very well. Such an operation could result in very high revocation cost. This is not applicable for mobile devices as well. Clearly, there is no proper solution which can effectively solve the secure data sharing problem in mobile cloud. As the mobile cloud becomes more and more popular, providing an efficient secure data sharing mechanism in mobile cloud is in urgent need.

## 2. Literature survey

*Gentry C, Halevi S. Implementing Gentry's Fully-*

*Homomorphic Encryption Scheme. In: Advances in Cryptology–Eurocrypt 2011. Berlin, Heidelberg: Springer Press, Pp. 129-148, 2011.*

We describe a working implementation of a variant of Gentry's fully homomorphic encryption scheme (STOC 2009), similar to the variant used in an earlier implementation effort by Smart and Vercauteren (PKC 2010). Smart and Vercauteren implemented the underlying "somewhat homomorphic" scheme, but were not able to implement the bootstrapping functionality that is needed to get the complete scheme to work. We show a number of optimizations that allow us to implement all aspects of the scheme, including the bootstrapping functionality. Our main optimization is a key-generation method for the underlying somewhat homomorphism encryption, that does not require full polynomial inversion. This reduces the asymptotic complexity from ˜O(n2:5) to ˜O (n1:5) when working with dimension-n lattices (and practically reducing the time from many hours/days to a few seconds/minutes). Other optimizations include a batching technique for encryption, a careful analysis of the degree of the decryption polynomial, and some space/time trade-offs for the fully-homomorphism scheme. We tested our implementation with lattices of several dimensions, corresponding to several security levels. From a "toy" setting in dimension 512, to "small,""medium," and "large" settings in dimensions 2048, 8192, and 32768, respectively. The public-key size ranges in size from 70 Megabytes for the "small" setting to 2.3 Gigabytes for the "large" setting. The time to run one bootstrapping operation (on a 1-CPU 64-bit machine with large memory) ranges from 30 seconds for the "small" setting to 30 minutes for the "large" setting.

*A. Disadvantage*

- lattice problems
- sparse-subset-sum problem
-

*Brakerski Z, Vaikuntanathan V. Efficient Fully Homomorphic Encryption from (Standard) Lwe. In: Proceeding of IEEE Symposium On Foundations of Computer Science. California, Usa: IEEE Press, Pp. 97-106, Oct. 2011.*

We present a fully homomorphic encryption scheme that is based solely on the (standard) learning with errors (LWE) assumption. Applying known results on LWE, the security of our scheme is based on the worst-case hardness of \short vector problems" on arbitrary lattices. Our construction improves on previous works in two aspects: 1. We show that \somewhat homomorphic" encryption can be based on LWE, using a new re- linearization technique. In contrast, all previous schemes relied on complexity assumptions related to ideals in various rings. 2. We deviate from the \squashing paradigm" used in all previous works. We introduce a new dimension-modulus reduction technique, which shortens the cipher texts and reduces the decryption complexity of our scheme, without introducing additional assumptions. Our scheme has very short cipher texts and we therefore use it to construct an asymptotically efficient LWE-based single-server private information retrieval (PIR) protocol. The communication complexity of our protocol (in the public-key model) is k _ polylog(k) + log jDBj bits per single-bit query (here, k is a security parameter).

*B. Disadvantages*

- Short vector problems
- Worst-case hardness of problems on ideal lattices.
- Data flow Diagram.

## 3. Data flow diagram

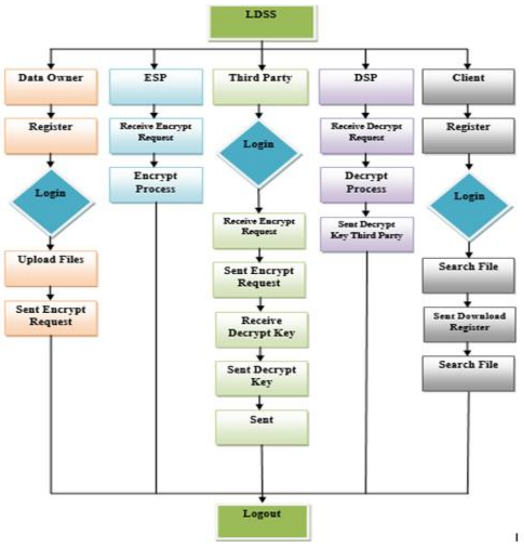

Fig. 1. Data flow diagram

## 4. System modules

*1) Module Description*

- Certification of files
- Privacy protection
- Request generation
- Forward security
- Access the files

*2) Certification of files*

Cloud storage is based on highly virtualized infrastructure and is like broader cloud computing in terms of accessible interfaces, near-instant elasticity and scalability, multi-tenancy, and metered resources. Certification of files refers to uploading the files in the cloud. Data Owner can upload the files into the cloud. Data owner upload the files in an Encrypted format in the cloud for providing more security to the particular data.

*3) Privacy protection*

Data owner uploads the data or a file only in an encrypted format for providing security. Using X-OR key encryption algorithm, a key will be randomly generated for uploading the data. (key is also in an encrypted format). When the data owner uploads the file into the cloud, data owner can hide some data or a file (data which the owner doesn't want to a public data) among all the files in the cloud/server. Hence user cannot view the hidden files in the server. Whenever data owner wants to display the data, he changes the hidden file into the unmasked data.

*4) Request generation*

User can view all the files or except the hidden files or data in the server. If any user wants to access the particular file or data in the server, then he sends a request to the particular data owner. User cannot access the data or a file in the cloud, without the permission of the data owner. Hence data owner can view all the user requests, and verify it. Then the user request will be forwarded to the Trusted Third Party Authenticator and the TTP will send the authentication to the user.

*5) Forward security*

The Authority people is able to view the list of uploader, users in this case he has the another option if he need to add the uploader he need to add otherwise delete and also he is able to give the keys for the requests from the user and uploader.

*6) Access the files*

The user can able to view the files if he wants to download the file he need to send the request to Authority after receiving the key he need to download.

## 5. Conclusion

In recent years, many studies on access control in cloud are based on attribute-based encryption algorithm (ABE). However, traditional ABE is not suitable for mobile cloud because it is computationally intensive and mobile devices only have limited resources. In this paper, we propose LDSS to address this issue. It introduces a novel LDSS-CP-ABE algorithm to migrate major computation overhead from mobile devices onto proxy servers, thus it can solve the secure data sharing problem in mobile cloud. The experimental results show that LDSS can ensure data privacy in mobile cloud and reduce the overhead on users' side in mobile cloud.

## References

[1] Gentry C, Halevi S. Implementing gentry's fully-homomorphic encryption scheme. in: Advances in Cryptology–EUROCRYPT 2011. Berlin, Heidelberg: Springer press, pp. 129-148, 2011.
[2] Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE. in: Proceeding of IEEE Symposium on Foundations of Computer Science. California, USA: IEEE press, pp. 97-106, Oct. 2011.
[3] Qihua Wang, HongxiaJin. "Data leakage mitigation for discertionary access control in collaboration clouds". the 16th ACM Symposium on Access Control Models and Technologies (SACMAT), pp.103-122, Jun. 2011.
[4] Adam Skillen and Mohammad Mannan. On Implementing Deniable Storage Encryption for Mobile Devices. the 20th Annual Network and Distributed System Security Symposium (NDSS), Feb. 2013.
[5] Wang W, Li Z, Owens R, et al. Secure and efficient access to outsourced data. in: Proceedings of the 2009 ACM workshop on Cloud computing security. Chicago, USA: ACM pp. 55-66, 2009.
[6] Maheshwari U, Vingralek R, Shapiro W. How to build a trusted database system on untrusted storage. in: Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4. USENIX Association, pp. 10-12, 2000.
[7] Kan Yang, XiaohuaJia, Kui Ren: Attribute-based fine-grained access control with efficient revocation in cloud storage systems. ASIACCS 2013, pp. 523-528, 2013.
[8] Crampton J, Martin K, Wild P. On key assignment for hierarchical access control. in: Computer Security Foundations Workshop. IEEE press, pp. 14-111, 2006.
[9] Shi E, Bethencourt J, Chan T H H, et al. Multi-dimensional range query over encrypted data. in: Proceedings of Symposium on Security and Privacy (SP), IEEE press, 2007. 350-364.