Indian Journal of Science and Technology, Vol 8(35), DOI: 10.17485/ijst/2015/v8i35/87834, December 2015
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

# Proxy Re-Encryption for Secure Data Storage in Clouds

**M. Martinaa* and V. Vaithiyanadhan**

School of Computing, SASTRA University, Thanjavur - 613401, Tamil Nadu, India;
nirmal_nancy@it.sastra.edu, vvn@it.sastra.edu

## Abstract

The evolving need for advanced care of patients at their homes along with the enormous growth in usage of mobile devices has resulted in the development of numerous mobile applications in enabling E-Medicine care. Cloud operations along with the developing mobile technologies is helping doctors in proper care of patients by enabling easy monitoring and assessing patient's health within their comfort limits. As such this leads to increase in frequency of sharing health information between doctors and nurses to provide better and safer care of patients. However in sharing information the hinders, privacy and safety issues may conflict with HIPAA standards. Through this paper we try to correct the privacy and security issues with E-Medicine care and cloud computing. We via cloud computing demonstrate a telemedicine application that allows doctors to monitor their patients remotely, and using this function as a base we tend to develop our model that tends to allow patients to share their health information with other doctors and medical professionals in a privilege and secure manner. The other indigenous feature includes the ability to handle big data and efficient counter mending.

**Keywords:** Cloud Computing, Customer Revocation, Elgamal Encyption, Large Data, Proxy Collusion, Proxy Re-Encryption, Secure Data Sharing

## 1. Introduction

The scope of the project is to establish a secure channel for storing Health Records of individuals in clouds, so that it can be accessed from anywhere irrespective of the geographical location.

Proxy Re-Encryption is based on the concept of a semi-trusted proxy that uses a re-encryption key to translate a cipher-text under the data owner's public key into another cipher text that can be decrypted by another user's private key. The data is never decrypted before it is re-encrypted hence the proxy will never be able to reveal the plaintext at any time. However the problem with this technique is that it does not handle the case where a revoked user and the proxy collude, which can then reveal all other user's private keys in the group. Also another major problem with this is that since it uses Elgamal public key cryptography, it will not allow the encryption or decryption of very large data, which is consequently a feature of medical data.

Remote caring of patients from home has become an increasing need as the world is developing day by day. Our project describes in Figure 1, that is, how health data is deployed in the cloud so that it can be accessed from anywhere irrespective of the Geographical Location provided there is an internet connection in the receiving end. In our project we record the ECG data and upload it to cloud. The Physician can be anywhere but he can check the ECG of the corresponding patient which will be stored in a database. The data will be stored in such a way that no third person gets access of the health data. For secure transmission of data we use Elgamal Encryption and Proxy Re-Encryption so that data will not be open to anyone other than the data owner and the data consumer. The key for encryption and decryption will be stored in a separate storage area called the Key Storage. Hence the database will be more secure and free of intruders.

In order to securely share data on cloud, several techniques can be used. Among them proxy re-encryption

*Author for correspondence

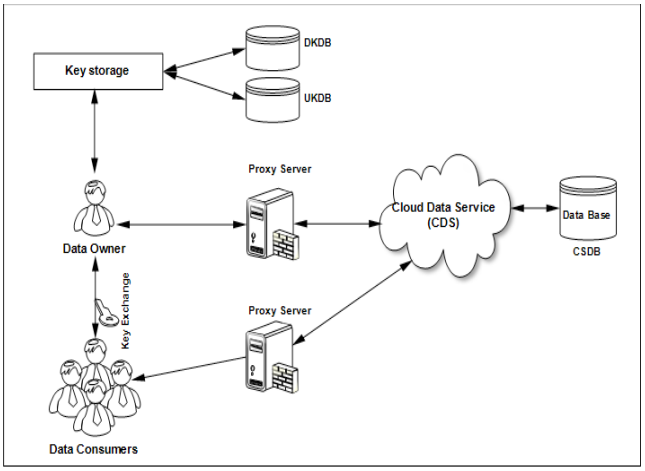


**Figure 1.** Architecture diagram for Secure data sharing

along with Elgamal Encryption is the most efficient technique for secure data sharing on cloud. The other techniques include the following:

Tran[5] used the Proxy Re-encryption technique using a single proxy. In this technique the key piece will be divided into two parts where one part will be with the data owner and other .When the data is to be uploaded the data will be first encrypted with the data owner's key piece and then it will be sent to the proxy server where it will be encrypted again with the remaining key piece and will be stored finally. When a data consumer wants access to the data, the same technique happens here. The key will be divided into two parts one with the proxy and other with the proxy. The proxy first decrypts the data with the key piece which it has and sends it to the data consumer where they can decrypt the data with the remaining key piece and get the entire plain text. Since there is a use of single proxy if the revoked user and proxy collude there is a chance of losing the privacy.

Nguyen Thanh Hung[6] provided solution for this case by using multiple proxies which eradicates the collusion process between the revoked user and the proxy.

Tu and Niu[9] used Cipher text Policy- Attribute Based Encryption (CP-ABE) for eradicating the user revocation technique by re-encrypting the data after the user is revoked rendering the revoked user's key useless. But this technique is not efficient for holding large data so it is not used.

In our work we get rid of certain limitations with the other works and provide a more efficient model for secure data sharing on cloud.The following table contains brief definitions of abbreviations used in our protocol.

| | |
|---|---|
| DSS | Data Sharing Service |
| CDS | Cloud Data Service |
| DO | Data Owner |
| DC | Data Consumer |
| KS | Key Storage |
| UKDB | User Key Data Base |
| DKDB | Data Key Data Base |

## 2. Storage of Health Data

### 2.1 Heart Beat Sensing

The heart beat sensing module to predict patient heart process, the sensor connects the patient to their body and starts the sensor monitor. The patient then runs an application. In Figure 2, illustrates Data Sharing, the application establishes a connection with the sensor. Once a connection is made, the sensor streams real-time health data to the application. The patient then inputs his/her email and password into the application, chooses a time interval and presses the Upload button. The app will send these credentials, details and the health data to the web service.

### 2.2 Data Storage

In Figure 3, data storage model for the database stored in the CSP our data storage model for the health-monitoring system. Note that the data storage model is generic medical data service for the proof of concept and is not focused on one particular medical service such as heart intensive care service.

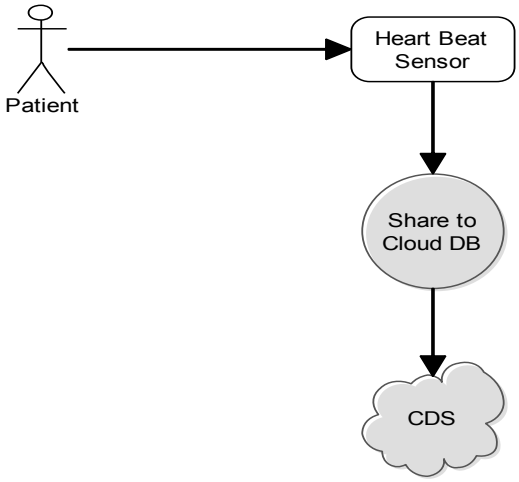


**Figure 2.** Data Share to Cloud DB.

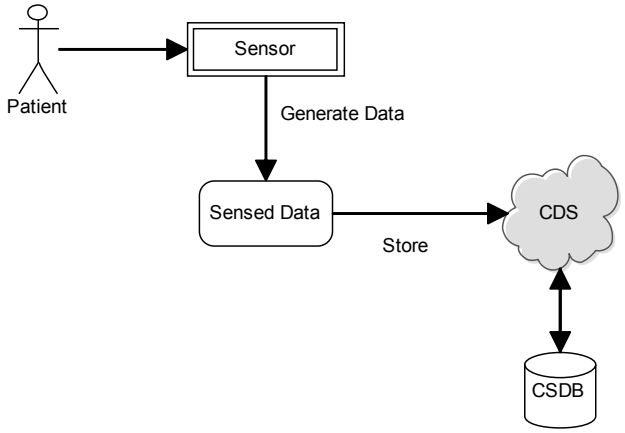


**Figure 3.** Data Storage Model.

The User table contains all the users of our system, including patients and doctors. The email and password are used for authentication and the role determines whether the user is a doctor or a patient. The Device table contains information about a health monitoring device connected to this system, such as the name and type of the device and its unique MAC address. The Service table creates a new Service record every time a user runs the application. It contains information such as the doctor that is authorized to view the data associated with the service, the device used, the patient it is monitoring, and the start and end times of the service. The Data table contains health data records that are generated every time steps 8 and 9 are called from the System.

## 2.3 Secure Data Sharing Protocol Initialization

The DO first sends a request to the DSS to upload data to the Cloud[1]. The DSS then generates a random private key x and its corresponding public key {p, b, c} using Elgamal encryption. In Figure 4, the DSS then partitions x into n+1 pieces and stores each piece in each of the n proxy servers. The DSS also generates a new user id for the data owner. The DSS then sends the user id, the remaining partitioned key piece, and the public key to the DO. The DO then generates a random symmetric key k and encrypts his data. The symmetric key is then encrypted itself by the DO using the public key {p, b, c} generated by the DSS. The DO then sends his user id, the encrypted data and the encrypted key to the DSS. The DSS generates a data id for the data. The DSS then sends the data id and the encrypted data to the CDS for storage. The DSS finally sends the data id and the encrypted key to the KS

## 2.4 Consumer Authorization

In Figure 5, DC wishes to access the DO's data, he sends an access request to the DO along with the data id of the data he wishes to gain access. Assuming the DO approves, he sends a request to the DSS and sends the request along with his user id, the data id and key piece. The DSS then verifies whether the data id and data owner id exist with a call to the CDS. If the CDS returns false, then the DSS notifies DO that the data does not exist and exits the protocol. If the CDS returns true, the DSS then retrieves the DO's key pieces from the proxy and computes the secret key x by adding all the key pieces together.

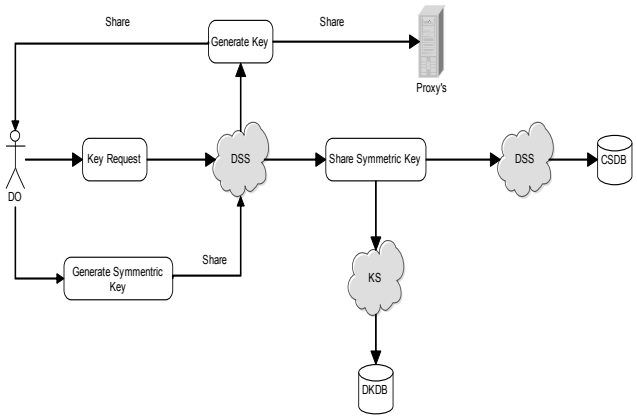


**Figure 4.** Key Management of Secure Data Sharing.

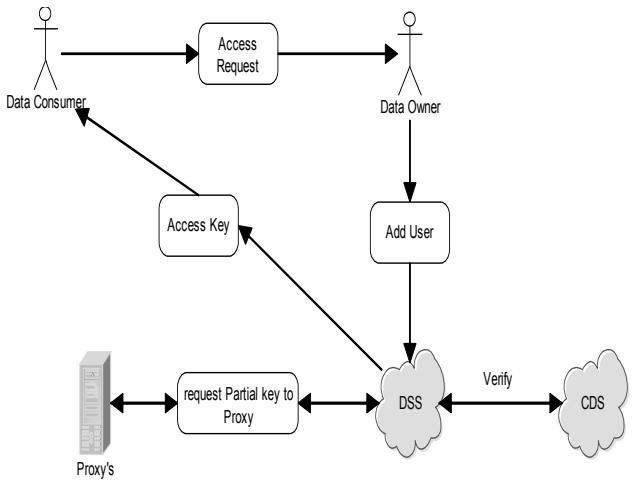


**Figure 5.** Customer Authorization.

The DSS will then generate new key pieces for the new DC, that when combined together is equivalent to the secret key x. The DSS will also generate a random user id as well as public/private key pair using Elgamal encryption for the DC. The DSS will then send the DC's user id, the public key and identifiers such as DO user id and data id to the KS. The KS will then store this in the UKDB. The newly generated key pieces corresponding to the DC are then stored in each of the proxy servers and the remaining piece is sent to the DO along with the private key of the DC. The DO finally sends this to the DC in a secure manner.

### 2.5 Data Access

Figure 6 illustrates the data accessing from CSDB, When a DC wishes to access data, he sends his key piece to the DSS along with identifiers to the data. The DSS gets the encrypted key from the DKDB via a call to the KS. The DSS then calls each proxy server to get the corresponding key piece of the DC and decrypts the encrypted key using each key piece. The DSS then uses the DC's key piece from step and decrypts the remaining encrypted key to reveal the full key. The DSS then gets the encrypted data from the CSDB via calls to the CDS. The encrypted data is then decrypted with the full key to reveal the full plaintext. The DSS then generates another arbitrary symmetric key and encrypts the data with this key. The DSS gets the corresponding DC's public key in the UKDB and encrypts the symmetric key using the public key. The encrypted data and the encrypted key are sent to the DC. The DC can then decrypt the key using his earlier distributed private key.

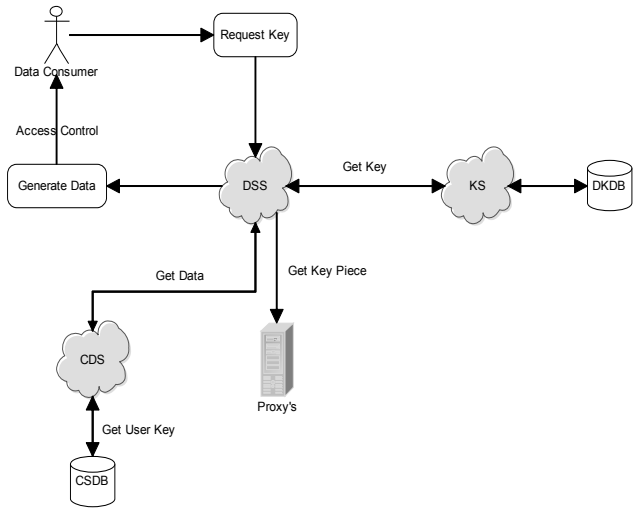


**Figure 6.** Data Access from CSDB.

### 2.6 Consumer Revocation

In Figure 7, When they DO decides to revoke a user access rights to data, he simply calls the DSS to request the user to be revoked rights to the specified data. The DSS will then remove the corresponding key pieces of the user in each of the proxy databases. Note that the data does not need to be re-encrypted and none of the other user's will be affected since only the key pieces corresponding to the user are removed. All other key pieces corresponding to other users still remain in the proxy database. Since the data does not need to be re-encrypted nor does their need to be any key re-distribution, the model is efficient and has a runtime of O(n) where n is the number of proxies.

## 3. Experimental Results

The data owner will send the secret key to the mail of the desired physician which the physician has mentioned while registering in the patient monitoring system. Once the secret key is received by the physician, they can download the ECG data that was uploaded in the cloud. In Figure 8, they can monitor the patient's ECG.

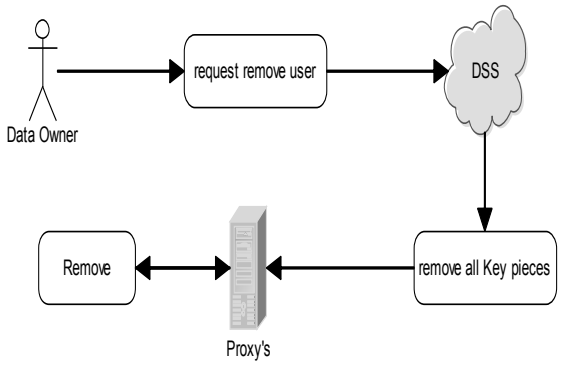


**Figure 7.** Consumer Revocation Diagram.

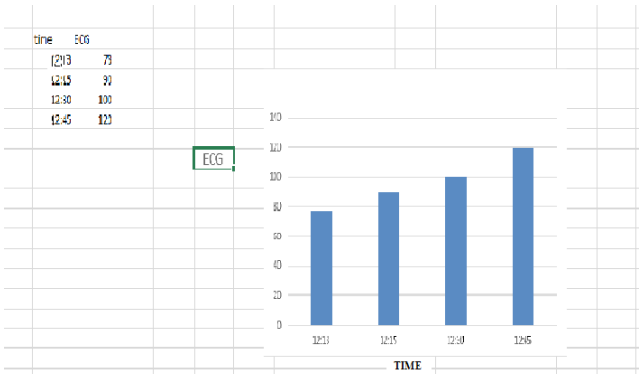


**Figure 8.** Patient ECG Monitoring.

## 4. Conclusion

In our work, we demonstrate how a secure channel can be established for storing health records on cloud. Being stored on cloud the physician can get access to check the ECG record of the corresponding patients and prescribe drugs or respective treatment based on the ECG data. This ECG data will be secure and cannot be accessed by any intruder. This way the patient need not meet the physician in person, instead of the can directly upload the data on the cloud and get treatment based on that.

## 5. References


1. Numera, nuumera acquires blue libris and expands offerings into tele care, network weekly news. 2012; 625.
2. Rocha F, Abreu S, Correia M. The final frontier: confidentiality and privacy in the Cloud. 2011; 44–50.
3. Guidelines on security and privacy in public cloud computing. National Institute of Standards and Technology (NIST). U.S. Department of Commerce. Special Publication. 800–144.
4. Wu R. Secure sharing of electronic medical records in cloud computing. Arizona State University, ProQuest Dissertations and Theses. 2012; 77.
5. Tran DH, Hai-Long N, Wei Z, Keong NW. Towards security in sharing data on cloud-based social networks. 8th International Conference on Information, Communications and Signal Processing (ICICS). 2011. p. 1–5.
6. Hung N T, Giang DH, Keong NW, Zhu H. Cloud enabled data sharing model. IEEE International Conference on Intelligence and Security Informatics (ISI). 2012. p. 1–6.
7. Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. 13th ACM Conference on Computer and Communications Security (CCS 06). 2006. p. 89–98.
8. Li J, Zhao G, Chen X, Xie D, Rong C, Li W, Tang L, Tang Y. Fine-grained data access control systems with user accountability in cloud computing. IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom). 2010. p. 89–96.
9. Tu S-S, Niu S-Z, Li H, Yun X-M, Li M-J. Fine grained access control and revocation for sharing data on clouds. Parallel and Distributed Processing Symposium Workshops and Ph.D. Forum (IPDPSW), 2012, IEEE 26th International. 2012 May 21–25; 2146–55.