A Platform for Secure Monitoring and Sharing of Generic Health Data in the Cloud

Danan Thilakanathan[a], Shiping Chen[b], Surya Nepal[b], Rafael Calvo[a], Leila Alem[b]

[a]*The University of Sydney, Australia*
Firstname.Lastname@sydney.edu.au
[b]*CSIRO ICT Centre*
Marsfield, Sydney, Australia
Firstname.Lastname@csiro.au

Abstract

The growing need for the remote caring of patients at home combined with the ever-increasing popularity of mobile devices due to their ubiquitous nature has resulted in many apps being developed to enable mobile telecare. The Cloud, in combination with mobile technologies has enabled doctors to conveniently monitor and assess a patient's health while the patient is at the comfort of their own home. This demands sharing of health information between healthcare teams such as doctors and nurses in order to provide better and safer care of patients. However, the sharing of health information introduces privacy and security issues which may conflict with HIPAA standards. In this paper, we attempt to address the issues of privacy and security in the domain of mobile telecare and Cloud computing. We first demonstrate a telecare application that will allow doctors to remotely monitor patients via the Cloud. We then use this system as a basis to showcase our model that will allow patients to share their health information with other doctors, nurses or medical professional in a secure and confidential manner. The key features of our model includes the ability to handle large data sizes and efficient user revocation.

Keywords: secure data sharing, cloud computing, proxy re-encryption, health monitoring system, cryptography

Acknowledgement

We would like to acknowledge Dr. Suraj Pandey for providing the documentation, code, sensors and app which helped us greatly in our research as well as allowed us to quickly deploy our prototype system.

1. Introduction

Advancements in mobile technology have allowed mobile devices, such as smartphones and tablets, to be used in a variety of different applications. With the added capability of Bluetooth [1] and Internet, and the fact that mobile phones are now a way of life amongst people of all ages due to its ubiquitous nature, it is now becoming more feasible than ever to use mobile technology for medical applications. A user can simply connect a health monitor to a mobile phone via Bluetooth to develop his or her own personal health monitor and management system.

There is currently a strong need to advance the field of health informatics [2]. As the world population ages due to increased life expectancy, this places pressure on the government to fund spending associated with the ageing population, especially in terms of health spending [3]. Consequently, the demand for cutting the cost of healthcare has increased and there is now a growing need for the remote caring of patients at home, in particular, for the elderly and the physically disabled. By leveraging the capability of mobile technology as well as Cloud computing, one can then develop a health monitoring system where the patient can be assessed by doctors in a remote location from the comfort of their own home. There are an abundant number of mobile apps available today for mobile telecare [4],[5],[6].

In recent times, there has also been a growing need for the sharing of health data between healthcare teams that include doctors, nurses and family members. Some benefits of sharing health information include safer and better health outcomes of the patient as the health professional gets a more complete picture of medical history. This is mainly due to not having to repeat medical history every time a health professional is consulted and also no more unnecessary tests. Sharing health information is also key to lowering healthcare costs [7]. However, the main issue with sharing health information is the privacy and security risks associated with it. Currently, there are not many mobile apps that handle this situation and we will attempt to address this problem in our work.

Building on advances in Cloud Computing, we seek to go beyond the mobile health applications, to enable secure sharing of telecare data in the Cloud. The Cloud, as an enabler for mobile telecare, can help provide effective treatment and care of patients due to its benefits such as on-demand access anywhere anytime, low costs and high elasticity. However, the Cloud is susceptible to privacy and security attacks, many of which occur from within the Cloud providers themselves [8] as they have direct access to stored data.

There is considerable work on protecting data from privacy and security attacks. NIST has developed guidelines to help consumers protect their data in the Cloud [9]. Encrypting data before storing to the Cloud is an effective way to prevent unauthorised users from accessing sensitive data. However, plain encryption techniques are not enough especially when considering the scenario of sharing data among a large group of users.

A trivial solution of data sharing is handing out encryption keys to all members of the group. In that way, only the members of the group will have data access. However, when considering the problem of user revocation, the data owner must re-encrypt the data again with a new key and distribute the new key to all remaining users. This is computationally inefficient and places a burden on the data owner when considering very large data sizes as well as large group sizes where users continually join and leave.

Our main contribution in this paper is to demonstrate our security protocol that will allow private and secure sharing of data in the Cloud within the context of mobile health applications. Moreover, we attempt to address the problems of achieving efficient user revocation, especially when considering large data sizes. First, we define a secure data sharing model and protocol. Second, we demonstrate the feasibility of the protocol through our developed prototype which combines smartphone, Bluetooth and Cloud computing technologies. Our protocol is generic and is not limited to only mobile health applications. Our protocol will also be efficient and handles the problem of user revocation and large data sizes. We also run a few experiments on our prototype and evaluate for feasibility of use in the real world. To the best of our knowledge, no other work has focused on private and secure sharing of health data via the Cloud. Many other works in mobile e-Health focus on securing communication over the Internet and do not consider data sharing [10], [11].

The remainder of this paper is organised as follows: We begin with a discussion of related work in Section 2. Next, we demonstrate our health-monitoring system in Section 3. We then discuss our secure data sharing algorithm in Section 4. In Section 5, we will discuss the limitations and constraints of our system. We then show how we implemented the system and give a brief evaluation in Section 6. Finally, in Section 7, we conclude our paper and discuss our future work.

2. Related Work

In this section, we review related work on data sharing in the Cloud and focus on literature used to handle the user revocation problem. We also briefly review previous related works on the integration of mobile technology and the Cloud for health-monitoring.

2.1. Data Sharing in the Cloud

Data sharing in the Cloud is fast becoming vital for organisations and social users alike. In a survey by InformationWeek [12], nearly all organisations shared their data in one way or another with 74% sharing data with customers and 64% sharing data with suppliers. The benefits include higher productivity and better time management for example by using collaborative tools such as Google Docs [13]. With social users, the benefits of data sharing are clear, for example with Facebook [14], where the ability to share photos and videos as well as share day-to-day information creates a sense of enjoyment in one's life and can also enrich some people as they are amazed at how many people are interested in the events in their lives. Healthcare providers are now rapidly turning to the Cloud and the benefits of sharing are also clear. Healthcare providers are willing to store and share electronic medical records via the Cloud and hence remove the geographical dependence between healthcare provider and patient [15]. Saradhy R. and Muralidhar K. [16] review the impact of the Internet on data sharing across many different organisations such as government agencies and businesses. Butler [17] describes the issues of data sharing on the internet where sharing information can allow users to infer details about users. Feldman and Patel et al. [18] discuss the important benefit of data sharing in terms of public health, in particular for education and professional development.

Tran et al. [19] utilises the idea of a proxy re-encryption scheme where the data owner's private key is divided into two parts where one is stored in the data owner's machine and the other on the proxy. The same occurs for all members of the group. The data owner encrypts data using his key piece while the proxy encrypts the entire data using the remaining key piece. An authorised member of the group can then retrieve data as the proxy will transform the ciphertext held by the data owner to another ciphertext which can be decrypted by the authorised member's key. The data will first be decrypted using the member's key piece in the proxy and then fully decrypted when the member decrypts it using the remaining key piece held by the authorised member. User revocation will simply involve removing the revoked user's key piece in the proxy and hence is efficient. However the problem with this technique is that it doesn't 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. We will extend upon this in our secure data sharing framework to handle the collusion and data size problem.

Nguyen Thanh Hung et al. [20] better handled the case of a revoked user colluding with a proxy by introducing a number of proxies. The more proxies there were, the less likely the

revoked user would be able to collude with all proxies to reveal all the keys. When a user is revoked access rights, the key manager simply instructs all proxies to remove the key pair corresponding to the revoked user. We take advantage of this in our data sharing model in order to enhance security.

Attribute-Based Encryption (ABE), originally proposed by Sahai and Waters et al. [21], is a technique used to enable fine-grained access control of data. Access Control Lists(ACLs) were initially used but were phased out as it provided only coarse-grained access to data and hence wasn't scalable, a primary feature of the Cloud [22]. There are two types of ABE [22]: Key-Policy ABE (KP-ABE) where the access control policy is stored with user's private key while a number of attributes are stored with the ciphertext. A user can decrypt data if, and only if, the attributes satisfy the access control policy. Ciphertext-Policy ABE(CP-ABE) is essentially the opposite of KP-ABE, where a number of attributes are stored with the user's private key and the access control policy is stored with the ciphertext.

Tu and Niu [23] makes use of CP-ABE in the context of enterprise applications and developed a revocation mechanism that simultaneously allows high adaptability, fine-grained access control and revocation. When a user is revoked access rights, the data is re-encrypted in the Cloud rendering the revoked user's key useless. Even though, the re-encryption process is delegated to the Cloud, this is not efficient when considering very large data sizes. Li and Niu et al. [24] also leverages ABE in the context of the sharing of personal health records (PHR) in the Cloud and is based on role-based fine-grained access control policies.

Yang et al. [25] proposed a combination of proxy re-encryption and attribute-based encryption to enable secure data sharing in the Cloud. It involved storing two keys; the ABE key and proxy re-encryption key as well as the encrypted data to the Cloud and uses an authorisation list. Data access involves using both keys to decrypt the data. The ABE key determines the user's access rights and the proxy key generates the cipher that can be transformed to enable decryption by the user's private key. User revocation simply involves removing the user from the authorisation list and hence is computationally efficient. However, the scheme does not handle the scenario where a revoked user rejoins the group with different access privileges.

Nguyen et al. [26] used the concept of bilinear maps to create a secure data sharing model in relation to tables. It involves splitting a key and giving a portion of the key to authorised users while the others are sent to the proxy. The user can then decrypt data with the help of all the proxies. Our method is different in that it is not limited to only table data and can handle large data sizes.

In our work, we attempt to address some of the shortcomings described to provide a better method of data sharing in the Cloud. We extend upon the work of Tran et al. [19] by attempting to solve the collusion problem and take advantage of the work by Nguyen Thanh Hung et al. [20] in terms of increasing the number of proxies, to provide a more secure and confidential method of data sharing in the Cloud.

2.2. Integration of Mobile and Cloud technologies for Health Monitoring

Recently, there have been works that have focused on the integration of mobile and cloud technologies for health monitoring. Fortino and Pathan et al. [27] introduced the BodyCloud architecture which enables the management and monitoring of body sensor data via the Cloud. It provides functionality receive and manage sensor data in a seamless way from a body sensor network (BSN). BodyCloud also comprises of a scalable framework that allows support for multiple data streams required for running concurrent applications.

Bellifemine et al. [28] and Fortino et al. [29] presented the SPINE framework. The open source framework allows developers to rapidly prototype and manage BSN applications. There are two main components of the SPINE framework namely the coordinator side which is implemented on a PC or smartphone and the BSN node side. On the coordinator side, SPINE provides application developers an intuitive interface to the BSN while on the node side, SPINE provides developers abstractions of hardware resources such as sensors, and an architecture to customise and extend the framework to support new physical platforms and services.

The scheme by Pandey et al. [30] integrates mobile and Cloud technologies with electrocardiogram (ECG) sensors to enable remote monitoring of patients with heart-related problems such as cardiac arrhythmias. The patient connects the sensors to their body and then run an app on a mobile device. The app connects to the sensors via Bluetooth. The app will then periodically upload data to the Cloud. The user can then download graphs from the Cloud which represent the user's health states. The scheme also implements middleware in the Cloud. There are web services for users to analyse their ECG, draw graphs, etc. The system is effective as it allows the user to adopt the Pay-As-You-Go methodology every time they require services to analyse their health data. The limitation with the scheme, however, is that it can only monitor ECG data and does not take into account monitoring other kinds of health problems such as pulse and temperature. Also, the current scheme does not handle security issues, in particular data sharing aspects. Our scheme builds on top of this scheme to enable a secure health-monitoring application that enables the user to share health data in a secure manner with other doctors and nurses and will be able to efficiently revoke users without putting too much burden on them. Our system is also generic in that it is not limited only to ECG monitoring and can also monitor other health data such as pulse and temperature.

2.3. Cloud Computing

Cloud Computing is the latest trend in the IT industry today and nearly all internet users use the Cloud in some form or another such as viewing emails via Hotmail or Gmail, writing documents via Google Docs, developing applications via Google App Engine and storing files via Amazon S3 or Windows SkyDrive. Cloud computing provides many benefits such as low costs due to the Pay-As-You-Go model, the ability to provide services and resources on-demand anywhere anytime, high availability as data is usually replicated among a number of servers, lowering the chance of data loss and provides elasticity in which more computing resources can be used when required [31]. Cloud computing is also benefiting healthcare [32]. Research from IDC Cloud [33] also indicates Cloud services will exceed $72 billion in 2015. Hence, Cloud technology can be a very attractive solution to develop medical applications and also allow patients to share their medical data with other doctors and nurses to allow for better care of patient's health.

However, when considering standards such as HIPAA (Health Insurance Portability and Accountability Act) [34], it is crucial for health-related data to be kept confidential from anyone unless authorised by the patient or some emergency regulations. The Cloud however is susceptible to many privacy and security attacks. As a result, many hospitals and health organisations are reluctant to adopt Cloud technology as a privacy breach in regards to its patient information can be devastating, especially in terms of cost [35]. As highlighted in the work of SeongHan et al. [36], the biggest obstacle hindering the progress and the wide adoption of the Cloud is privacy and security issues associated with it. This is further evidenced from a survey carried out by IDC Enterprise Panel [37], where most users regarded security as the top challenge.

In fact, many privacy and security attacks occur from within the Cloud providers themselves [8] as they usually have direct access to the stored data and some of them even sell the data to the third parties in order to gain profit. In fact malicious insiders represent one of the major issues affecting the Cloud as highlighted in Duncan et al. [38]. There are many examples of this as described in Huang et al. [39] such as the leaking of shared user documents, a company going out of business after losing 45% of user data due to administrator error, another company leaking its customer list and also a website forced to be shutdown as it allowed many illegal files to be shared such as pirated films as full television shows [45]. Several studies [46] have discussed privacy and security issues associated with the Cloud and current solutions used to achieve privacy and security protection in the Cloud.

2.4. Similar remote health-monitoring mobile platforms

AliveCor is a remote app-based ECG monitoring system [41]. The system is similar to our system in that it allows a patient to monitor their ECG on their iPhone and also share their ECG data to whomever the patient wants. It provides a bundle of useful features such as recording, displaying, transferring and storing high quality ECG data. However, the system was not developed with security in mind, hence it is possible an intruder will be able to steal health data with a certain amount of effort.

CardioComm Solutions also demonstrated their remote patient ECG monitoring service called HeartCheck Smart Monitoring [42]. The system allows rapid access and allows physicians to better review the ECG data in order to assess how the patient should be treated. However, it doesn't specifically focus on privacy and security aspects such as confidentiality of data as it is being transmitted to the Cloud or as it is stored within the Cloud.

Gradl et al. [43] also developed a Android-based application that allows real-time monitoring of ECG data similar to our prototype as well as automated arrhythmia detection. However, the application also does not focus on privacy and security aspects.

Xia et al. [44] addresses the usefulness of ECG data collected from patients themselves using mobile devices and the issues it presents. It does not focus on the security aspects associated with sending data to the Cloud.

Solutions are now tailored towards monitoring health using mobile devices, yet not many solutions focus on the security aspect of this. Our solutions leverages the mobile health idea and additionally incorporates privacy and security mechanisms to guarantee patient confidentiality; a crucial requirement in the health industry today.

3. Preliminaries

3.1. ElGamal Encryption

ElGamal encryption, invented by T. ElGamal [47] is a public-key cryptography system. We take advantage of ElGamal Encryption in our work since it does not rely on a user's public key infrastructure (PKI) and the algorithm is both simple and efficient. There are three main steps of the ElGamal encryption algorithm:

- Initialisation: Given a prime p, a primitive root c of p, compute $b = c^x mod p$. Also randomly select a secret key x. The public key is thus $\{p, b, c\}$ and private key is x.

- Encryption: Generate random value r and encrypt data m as follows:

$$\begin{aligned} E(m) &= m.b^r \ mod p \\ &= m.c^{rx} \ mod p \end{aligned} \tag{1}$$

  Also note: $g = c^r mod p$

- Decryption: This decrypts m with secret key x as follows:

$$\begin{aligned} D_x(E(m)) &= g^{-x}.E(m) \ mod p \\ &= (c^r)^{-x}.m.c^{rx} \ mod p \\ &= c^{-rx}.m.c^{rx} \ mod p \\ &= m \ mod p \end{aligned} \tag{2}$$

3.2. Proxy Re-encryption

Proxy Re-Encryption [19],[48] is based on the concept of a semi-trusted proxy that uses a re-encryption key to translate a ciphertext under the data owner's public key into another ciphertext 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. Many recent works have realised proxy re-encryption as a technique to enable data sharing in the Cloud. In our work, although we don't use proxy re-encryption explicitly, our system mimics a proxy re-encryption algorithm scheme from the point of view of the data owner and consumer.

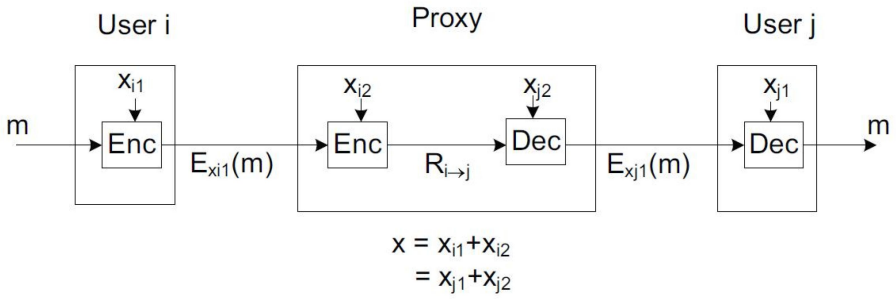


Figure 1: The Proxy Re-encryption scheme

Figure 1 depicts an ElGamal-based proxy re-encryption scheme used in the work of Tran et al. [19]. We make use of this idea in our secure sharing model and protocol. A user, say Alice who wants to share data with Bob encrypts her data using her public key and sends it to the proxy. The proxy converts the encrypted data under Alices public key to an encrypted data under Bobs public key without ever revealing the plaintext.

4. The Health Monitoring System

Figure 2 depicts our developed health monitoring system. It consists of the Cloud Storage Provider (CSP), the users of the system such as the patient and/or doctor, a health sensor, and a mobile device. Our system makes use of both proxy re-encryption and ElGamal encryption. We also discuss another similar work on health monitoring using mobile and Cloud technologies.

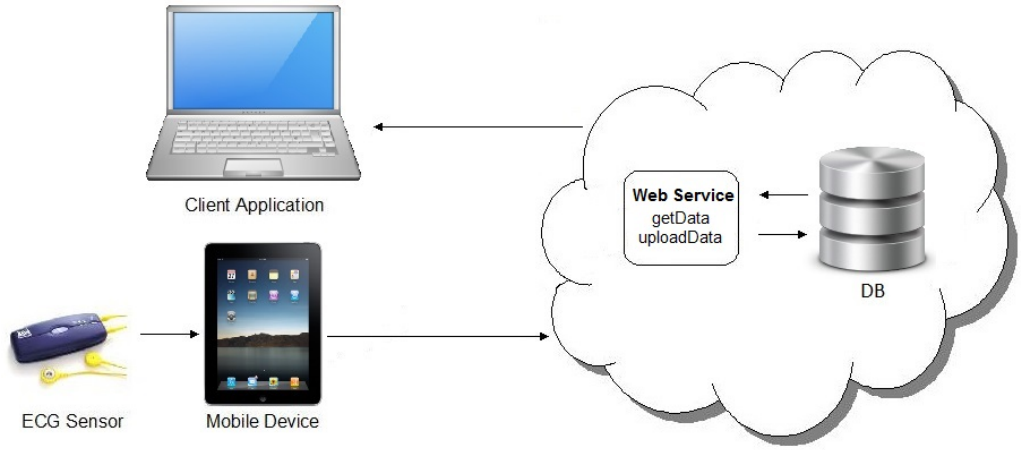


Figure 2: The Health-Monitoring System

4.1. Scenario

Consider the scenario of an elderly patient who suffers from occasional heart problems. He requires care on a regular basis. But due to his age and distant geographical location, he struggles to make a trip to his doctor regularly. Currently, the only option he has is to have a doctor or nurse visit him to monitor his heart to check for cardiac arrhythmias. This is very costly in both time and budget for both the doctor and the elderly sufferer. On some days, the doctors trip becomes unnecessary if the patient is coping well and on other days the doctor be required more often but may not always be there. The elderly sufferer would benefit from a system that will not only allow in-home monitoring but also the monitoring by a doctor without having to leave his room or without the doctor having to leave his surgery.

4.2. System Requirements

Considering the scenario, we derive a number of requirements for the health-monitoring system. Firstly, the system should allow a patient to monitor their health anytime anywhere. The system should also not depend on the patient or doctor's geographical location. The system should be scalable to handly many patients and health care teams such as doctors and nurses as well as different health devices and data formats. More importantly, the system should be generic in the sense that it should be able to monitor different health scenarios. Finally, the system should be user-friendly and simple enough for the elderly.

4.3. System Functionality

The functionality of the system is described as follows. The patient connects the sensors to their body and starts the sensor monitor. The patient then runs an app on a mobile device. The mobile app establishes a connection with the sensor via Bluetooth. Once a connection is made, the sensor streams real-time health data to the app. The patient then inputs his/her email and password into the app, chooses a time interval and presses the Upload button. The app will send

these credentials, details and the health data to the web service. The web service will then check the credentials in the database held by the CSP. If the user exists, the web service will store the health data corresponding to the user in the CSP. The app will periodically send the credentials and health data to the web service and steps 7-8 will be repeated until the user stops the app.

When a doctor wants to view the patients health data, they simply run an application on their computer. The application makes calls to the web service to retrieve the authorised patients health data. The doctor can then view, interpret and analyse the patients health and recommend any further actions to take. The geographical location of the doctor is not important; as long as they have a computer, an internet connection and the simple application, they will be able to view the data nearly anywhere anytime.

4.4. Data Model

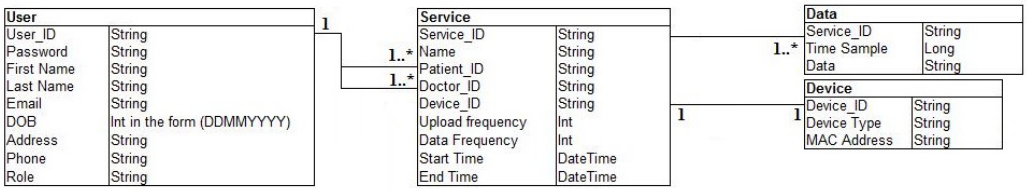


Figure 3: Data Model

Since the focus of this paper will be on enabling secure data sharing in the Cloud, the data model for the database stored in the CSP will be described. Fig. 3 illustrates our data model for the health-monitoring system. Note that the data 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.

The User table contains all the users of our system, including patients and doctors. The email and password is 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 app on the mobile device. It contains information such as the doctor that is authorised 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 Functionality. A data record can only be a part of a Service.

4.5. Privacy Issues for using CSP

Since the Cloud is at the forefront of many privacy and security attacks and the fact that many privacy attacks come from within the CSP itself as they usually have direct access to data and may steal data to sell to the third parties in order to gain profit, the entire database in the CSP needs to be encrypted. This means the data needs to be encrypted before sending the data over-the-wire. This will prevent any malicious outsider as well as the CSP itself to gain any useful data without the decryption key.

Since our focus is on data sharing with doctors and nurses, simple encryption techniques are not enough. As discussed, if we have many doctors and nurses authorised to view the patients data and the patient decides to revoke a specific doctor access rights to their data, the patient has to re-encrypt their data using another key and send the new key to all the other doctors and nurses. This is computationally inefficient and places a burden on the patient to re-encrypt and distribute new keys each time when revoking doctors/nurses access rights. It also places a burden on the remaining members of the group, as they constantly have to update their key set in order to keep up-to-date with all their patients data. The main reason why the patient would need to re-encrypt the data with a new key is that the revoked doctor still holds the key and can still theoretically access the data even if he is not allowed to. It cannot be assumed that the doctor or nurse will never view the patients data or that they will always keep the key a secret. This will then conflict with HIPAA standards that any entity should not access a patients health information without the patients consent.

5. Secure Data Sharing Model and Protocol

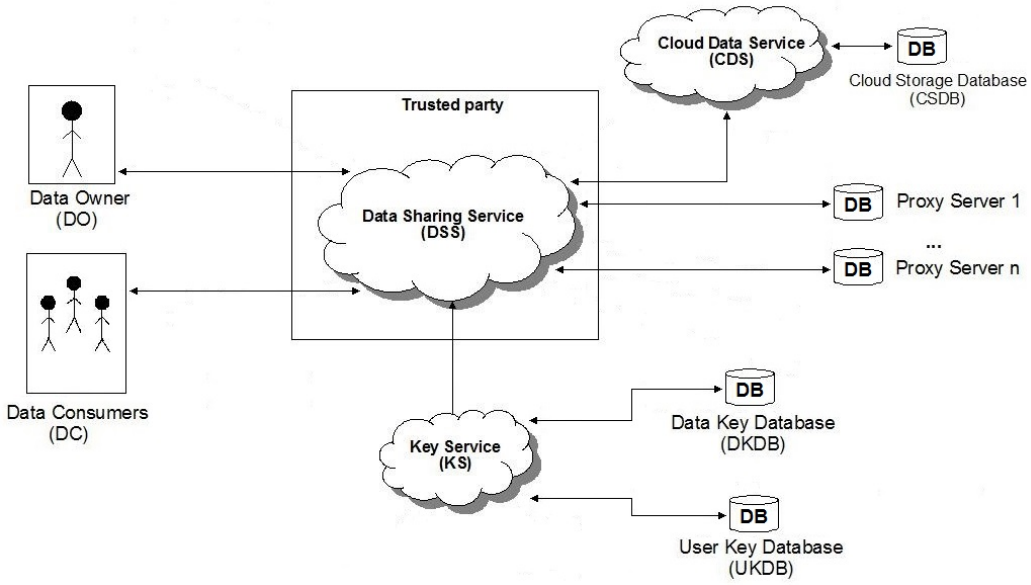


Figure 4: Secure Data Sharing Architecture

We now introduce our secure model and protocol that will enable data sharing amongst a group of users specified by the data owner. Our techniques is not limited in any way to medical applications and can be can be applied to other Cloud applications. Our protocol builds on the work of Tran et al. [19] as described in Section II and handles the collusion problem.

5.1. Secure Data Sharing Model

Figure 4 enhances the eHealth monitoring system illustrated in Figure 2 by adding a security layer that enables efficient secure data sharing. The original web service now represents the

Cloud Data Service (CDS) and we add another web service called the Data Sharing Service (DSS) that handles the data sharing aspects of the system. We assume throughout the rest of this paper that the DSS is fully trusted. However, this makes it particularly vulnerable to attackers and hence the DSS itself needs to be protected. In order to achieve this, the DSS can be modelled as a trusted private Cloud provider protected using traditional mechanisms such as internet firewalls. There are also a number of proxy services to store key pieces for members of the group and a Key Service (KS) to store the encrypted keys of the health data and keys of the data consumers (DC).

To briefly summarise how the model works, we assume that each user in the group, including the data owner (DO), has a key that can decrypt the appropriate keys in the Data Key Database (DKDB). However, their keys are partitioned into $n + 1$ parts where n parts are stored in each proxy and the user keeps the extra part. In this way, none of the users know the full key required to decrypt the keys in the Data Key database. When the user requires data access, they call the DSS. The DSS then decrypts the key in the DKDB using all the key pieces in the proxy database corresponding to the calling user. The key is then used to decrypt the data in the Cloud. When the data owner requests a user to be revoked, their key pieces in the proxies are simply removed and the original data need not be re-encrypted nor does their need to be any re-distribution of keys to remaining users. None of the other data consumers will be affected by the revocation since their corresponding key pieces still remain intact in the proxies and also with themselves. We will explain this in detail in the next section.

5.2. Secure Data Sharing Protocol

We now discuss our data sharing protocol in detail. The protocol has 4 steps namely; *initialisation*, *consumer authorisation*, *authorised data access*, and *consumer revocation*. Also of important note, we assume the DSS to be fully trusted in that it will always honestly follow the protocol. We make use of ElGamal encryption in our work and build upon the work of Tran et al. [19] to provide a more secure platform for data sharing. The following table contains brief definitions of abbreviations used in our protocol.

DO	Data Owner	The owner of the data and decides who has access permission to the data
DC	Data Consumer	Any user who has permission to access data given by the DO
DSS	Data Sharing Service	The trusted service that carries out most of the data sharing functionality in the protocol (see model)
CDS	Cloud Data Service	The service that allows calls to be made to Cloud storage (see model)
KS	Key Service	The service that allows calls to be made to the Cloud key service to obtain and store encryption keys (see model)
CSDB	Cloud Storage Database	The database containing encrypted data (see model)
DKDB	Data Key Database	The database which stores encryption keys which are encrypted themselves (see model)
UKDB	User Key Database	The database which stores all users including DC and DO private keys (see model)

5.2.1. Initialisation

1	DO → DSS	{key_request}
2	DSS	Generate key x
		$b = c^x mod\ p$
3	DSS	Generate $x_1 + x_2 + x_3 + ... + x_n + x_{n+1} = x$
		Generate u_{DO}
4	for (all proxy i)	
	DSS → proxy i	$\{u_{DO}, x_i\}$
5	DSS → DO	$\{u_{DO}, x_{n+1}, \{p, b, c\}\}$
6	DO	Generate symmetric key k
		$E_k(m)$
		Generate r, $\gamma = c^r mod\ p$
		$E_{\{p,b,c\}}(k) = (c^r, c^{rx}.k\ mod\ p)$
7	DO → DSS	$\{u_{DO}, E_k(m), E_{\{p,b,c\}}(k)\}$
8	DSS	Generate d_m
9	DSS → CDS → CSDB	$\{u_{DO}, d_m, E_k(m)\}$
10	DSS → KS → DKDB	$\{u_{DO}, d_m, E_{\{p,b,c\}}(k)\}$

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 (2). The DSS then partitions x into $n + 1$ pieces (3) and stores each piece in each of the n proxy servers (4). The DSS also generates a new user id for the data owner (3). The DSS then sends the user id, the remaining partitioned key piece, and the public key to the DO (5). The DO then generates a random symmetric key k and encrypts his data with it (6). The symmetric key is then encrypted itself by the DO using the public key $\{p, b, c\}$ generated by the DSS (6). The DO then sends his user id, the encrypted data and the encrypted key to the DSS (7). The DSS generates a data id for the data (8). The DSS then sends the data id and the encrypted data to the CDS (9) for storage. The DSS finally sends the data id and the encrypted key to the KS (10).

5.2.2. Consumer Authorisation

1	DC → DO	{access_request, d_m}
2	DO → DSS	addUser(u_{DO}, d_m, x_{n+1})
3	DSS ↔ CDS	Verify u_{DO}, d_m exists. If not, exit here.
4	for (all proxy i)	
	DSS → proxy i	$\{u_{DO}$, key_piece_request$\}$
	proxy i → DSS	x_i
5	DSS	Compute $x_1 + x_2 + x_3 + ... + x_n + x_{n+1} = x$
		Generate $x_{u1} + x_{u2} + x_{u3} + ... + x_{un} + x_{u(n+1)} = x$
		Generate u_{DC}
		Generate $\{p_{DC}, b_{DC}, c_{DC}\}$, x_{DC}
6	DSS → KS → UKDB	$\{u_{DC}, u_{DO}, d_m, \{p_{DC}, b_{DC}, c_{DC}\}\}$
7	for (all proxy i)	
	DSS → proxy i	$\{u_{DC}, u_{DO}, d_m, x_{ui}\}$
8	DSS → DO → DC	$\{u_{DC}, x_{u(n+1)}, x_{DC}\}$

When a DC wishes to access the DO's data m, he sends an access request to the DO along with the data id of the data he wishes to gain access to (1). 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

(2). The DSS then verifies whether the data id and data owner id exists with a call to the CDS (3). If the CDS returns false, then the DSS notifies DO that the data does not exist and exits the protocol (3). If the CDS returns true, the DSS then retrieves the DO's key pieces from the proxy (4) and computes the secret key x by adding all the key pieces together (5). The DSS will then generate new key pieces for the new DC, that when combined together is equivalent to the secret key x (5). The DSS will also generate a random user id as well as public/private key pair using ElGamal encryption for the DC (5). 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 (6). The KS will then store this in the UKDB (6). The newly generated key pieces corresponding to the DC are then stored in each of the proxy servers (7) and the remaining piece is sent to the DO along with the private key of the DC (8). The DO finally sends this to the DC in a secure manner (8).

5.2.3. Authorised Data Access

1	DC → DSS	$\{u_{\mathrm{DC}}, u_{\mathrm{DO}}, d_m, x_{\mathrm{u(n+1)}}\}$
2	DSS → KS → DKDB	getKey(u_{DO}, d_m)
3	DKDB → KS → DSS	$E_{\{\mathrm{p,b,c}\}}(k) = (c^r, c^{\mathrm{rx}}.k \ mod \ p)$
4	DSS → proxy 1	getKeyPiece(u_{DC})
5	proxy 1 → DSS	x_{iu}
6	DSS	$D_{x_{\mathrm{iu}}}(E_{\{\mathrm{p,b,c}\}}(k)) = (c^r, (c^r))^{-x_{\mathrm{iu}}}.c^{\mathrm{rx}}.k \ mod \ p)$
		$= (c^r, (c^r))^{x-x_{\mathrm{iu}}}.k \ mod \ p)$
7	Repeat 4-6 for proxies 2...n	Remaining cipher: $(c^r, (c^r)^{x-x_{1\mathrm{u}}-x_{2\mathrm{u}}-\ldots-x_{\mathrm{nu}}}.k \ mod \ p)$
8	DSS	$D_{x_{\mathrm{u(n+1)}}}((c^r, (c^r)^{x-x_{1\mathrm{u}}-x_{2\mathrm{u}}-\ldots-x_{\mathrm{nu}}}.k \ mod \ p))$
		$\rightarrow (c^r, (c^r)^{-x_{\mathrm{u(n+1)}}}.(c^r)^{x-x_{1\mathrm{u}}-x_{2\mathrm{u}}-\ldots-x_{\mathrm{nu}}}.k \ mod \ p)$
		$\rightarrow (c^r, (c^r)^{x-x_{1\mathrm{u}}-x_{2\mathrm{u}}-\ldots-x_{\mathrm{nu}}-x_{\mathrm{u(n+1)}}}.k \ mod \ p)$
		$\rightarrow (c^r, k \ mod \ p)$ since $x = x_1 + x_2 + x_3 + ... + x_n + x_{\mathrm{n+1}}$
9	DSS → CDS → CSDB	getData(u_{DO}, d_m)
10	CSDB → CDS → DSS	$E_k(m)$
11	DSS	$D_k(E_k(m)) = m$
		Generate $k1$
		$E_{\mathrm{k1}}(m)$
12	DSS → KS → UKDB	getUserKey(u_{DC})
13	UKDB → KS → DSS	$\{p_{\mathrm{DC}}, b_{\mathrm{DC}}, c_{\mathrm{DC}}\}$
14	DSS	Generate r_{DC}, $\gamma_{\mathrm{DC}} = {c_{\mathrm{DC}}}^{r_{\mathrm{DC}}} mod \ p_{\mathrm{DC}}$
		$E_{\{p_{\mathrm{DC}},b_{\mathrm{DC}},c_{\mathrm{DC}}\}}(k1) = ({c_{\mathrm{DC}}}^{r_{\mathrm{DC}}}, {c_{\mathrm{DC}}}^{r_{\mathrm{DC}}x_{\mathrm{DC}}}.k1 \ mod \ p)$
16	DSS → DC	$\{E_{\{p_{\mathrm{DC}},b_{\mathrm{DC}},c_{\mathrm{DC}}\}}(k1), E_{\mathrm{k1}}(m)\}$
17	DC	$D_{x_{\mathrm{DC}}}(E_{\{p_{\mathrm{DC}},b_{\mathrm{DC}},c_{\mathrm{DC}}\}}(k1))$
		$= ({c_{\mathrm{DC}}}^{r_{\mathrm{DC}}}, ({c_{\mathrm{DC}}}^{r_{\mathrm{DC}}})^{-x_{\mathrm{DC}}} {c_{\mathrm{DC}}}^{r_{\mathrm{DC}}x_{\mathrm{DC}}}.k1 \ mod \ p)$
		$= ({c_{\mathrm{DC}}}^{r_{\mathrm{DC}}}, k1 \ mod \ p)$
		$D_{\mathrm{k1}}(E_{\mathrm{k1}}(m)) = m$

When a DC wishes to access data, he sends his key piece to the DSS along with identifiers to the data (1). The DSS gets the encrypted key from the DKDB via a call to the KS (2, 3). The DSS then calls each proxy server to get the corresponding key piece of the DC (4, 5) and decrypts the encrypted key using each key piece (6, 7). The DSS then uses the DC's key piece from step (1) and decrypts the remaining encrypted key to reveal the full key (8). The DSS then gets the encrypted data from the CSDB via calls to the CDS (9, 10). The encrypted data is then decrypted with the full key to reveal the full plaintext (11). The DSS then generates another arbitrary

symmetric key and encrypts the data with this key (11). The DSS gets the corresponding DC's public key in the UKDB (12, 13) and encrypts the symmetric key using the public key (14). The encrypted data and the encrypted key are sent to the DC (16). The DC can then decrypt the key using his earlier distributed private key (17). Once the key is decrypted, the DC can then decrypt the data itself to reveal the full plaintext (17).

5.2.4. Consumer Revocation

1	DO → DSS	removeUser(u_{DO}, u_{DC}, d_m)
2	for (all proxy i)	
	DSS → proxy i	removeKeyPiece(u_{DO}, u_{DC}, d_m)
	proxy i	Remove $x_{u_{\mathrm{DC}}i}$

When the 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 (1). The DSS will then remove the corresponding key pieces of the user in each of the proxy databases (2). 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 is 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.

5.3. Security Analysis

From the description of our data sharing model and protocol, we now provide the model and protocol from a security perspective.

1. *Collusion between user and proxy* – If a non-revoked user colludes with all the proxies, in theory he can retrieve the secret key x by combining his x_u with his key pieces in the proxies. This will enable him to decrypt $E_{\{\mathrm{p,b,c}\}}(k)$ to reveal the key k and then decrypt the data itself to reveal m. If only one proxy was used such as with the work of Tran et al. [19], the likelihood of a user compromising a proxy would be high. However, the main distinguishing characteristic of our security model is that it supports multiple proxies and the actual number of proxies used in a system depends on its security requirements. Therefore the chance of a user colluding is much lower if more proxies are used, because each proxy database can be modeled as different CSPs in different locations around the world.
2. *Privacy of data against the Cloud* – There is no stage in our data model where the Cloud stores the plaintext of the data m or the key k. The Cloud would need the secret key x to be able to retrieve key k and then later retrieve m. Even if the Cloud is able to retrieve all the key pieces of all users from every proxy, it still cannot reveal the secret key x without colluding with a user. However, this is also very difficult as there is a low chance that the Cloud will be able to compromise all proxies. In principle, the more proxy databases there are, the more secure the system will be from privacy attacks. However, too many proxies can make the reliability of the whole system low. So there is a tradeoff between data privacy and reliability of the data service, which is beyond of scope of this paper.
3. *Consumer Revocation* – When a DC is revoked access rights to the data, his corresponding key pieces are simply removed from all the proxies and hence is efficient when compared to having to re-encrypt data and re-distributing keys to the remaining users. The revocation scheme has an efficiency of $O(n)$ where n is the number of proxy databases. At no

point does the DC know the data key k nor the full secret key x. Also, even if a revoked user colludes with all proxies, he will not be able to retrieve all keys since his pieces are removed. Even if he steals another users key pieces he still cannot recover x and hence the system is secure against revoked users colluding with the proxies.

4. *Large Data Sizes* – The work of Tran et al. [19] does not handle large data sizes effectively as the ElGamal cryptography algorithm discussed can only provide cryptography operations with data upto a certain size. Since our focus is within the context of health applications and data involved within the health domain tends to be very large, the protocol discussed in [19] is not suitable. Our protocol extends this and handles large data sizes effectively. The DO generates the symmetric key k that will be able to handle the encryption of the large data. This key can be any type of symmetric key such as RSA, AES, etc. Large data sizes are handled effectively and efficiently since ElGamal cryptography is used to encrypt/decrypt the symmetric key itself since the symmetric key is not likely to be too large.

6. Implementation and Evaluation

In this section, we will describe our implementation of the system, followed by a security evaluation of our system. We finally carry out a number of performance tests and provide an evaluation on the performance of the developed system.

6.1. Implementation

We implemented this system using Java. The Java Android SDK was used to develop the mobile app and was deployed on an ASUS Eee Pad Transformer Prime TF201 Tablet [49]. The tablet is capable of reading Bluetooth data from a variety of sources. The app, called HeartBeatSense was developed in such a way that it can be deployed and run on any Android device regardless of the type and/or size of the mobile device. This provides more convenience for everyday users and provides a greater reach for more users to gain benefits from using our system. Figure 5 illustrates our app carrying out the monitoring of a patient's ECG.

The web services was created using Java and Axis2. We deployed our web services using Apache Tomcat 7. We used MySQL 5.5 to represent the Cloud Storage database backend. For the client side application, which is used by either patient or doctor, we created a simple Java application that simply makes calls to get and receive data as well as authenticate the users.

For the sensors, we used the AliveECG Heart Activity Monitor [50] with Ambu sensors [51]. The sensors connect to the Heart Activity Monitor to measure the persons ECG activity. ECG stands for electrocardiogram and can be used to measure the electrical activity of the heart [52]. The Heart Activity Monitor can then send the ECG data out to an SD card or other devices via Bluetooth technology.

6.2. System Security

We now evaluate the security of our mobile-based prototype.

1. *Privacy violation* – In our prototype, the email and password are both stored with an additional random salt and consequently hashed. The hash values are then stored in the Cloud database. They are not stored in the clear and hence administrators themselves would not know the full credentials of the user. These credentials are required to use the system.

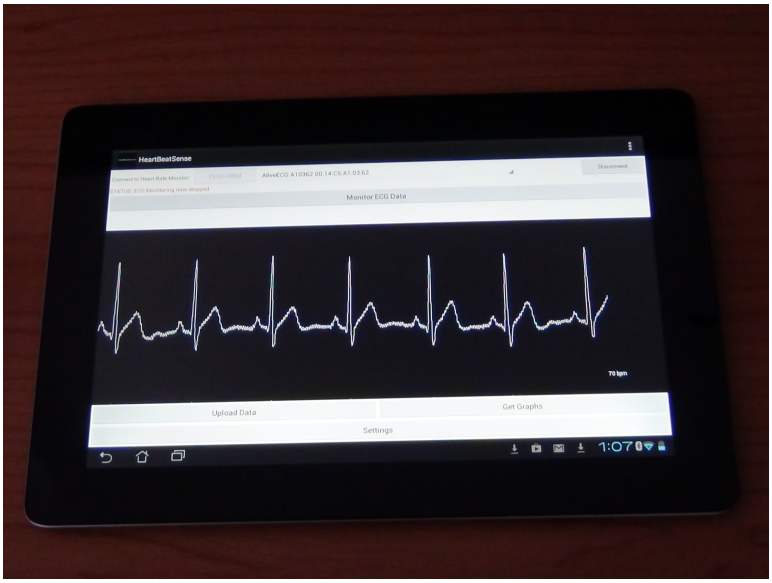


Figure 5: HearbeatSense app

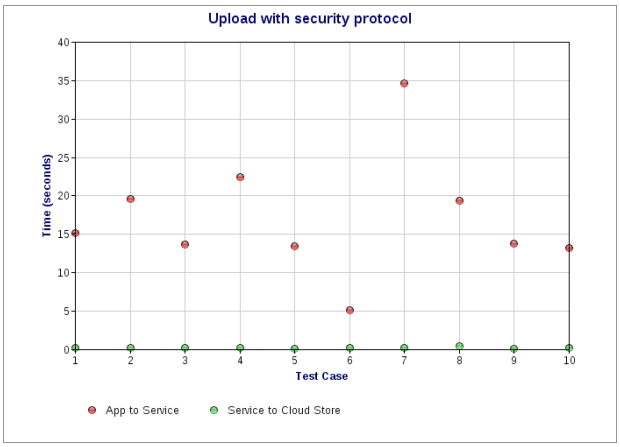
2. *Mobile stealing* – Even if an attacker steals a mobile phone in the hopes of finding any valuable confidential data, they will not find anything of value since nothing will be stored on the mobile phone. Once data is received from the health device, it is sent straight to the Cloud. Hence an attacker will not be able to find any trace of health data on the mobile device even when stolen.
3. *Sniffing attacks* – It is possible for an attacker to retrieve data as it is being sent from the mobile client to Cloud storage. However, our system first encrypts data using a symmetric key and then the symmetric key itself is encapsulated via the user's public key. The attacker would have to know the user's secret key (unbeknown even to the user) to decrypt the symmetric key and consequently the data making our system attractive to use in such scenarios.
4. *On-board tampering* – If a mobile device is suspected of on-board tampering, the user can simply use another mobile device. Any mobile device able to run our app would suffice as long as the user keeps his/her email and password a secret. Also, our app does not store any valuable information on a user's mobile phone. It just connects with a heart monitor via Bluetooth and forwards the data to the Cloud. As long as the user inputs his credentials in any Android mobile device containing the app, the data will be stored in the correct location identifiable to any authorised doctors, hence preventing data loss.

6.3. Performance Tests

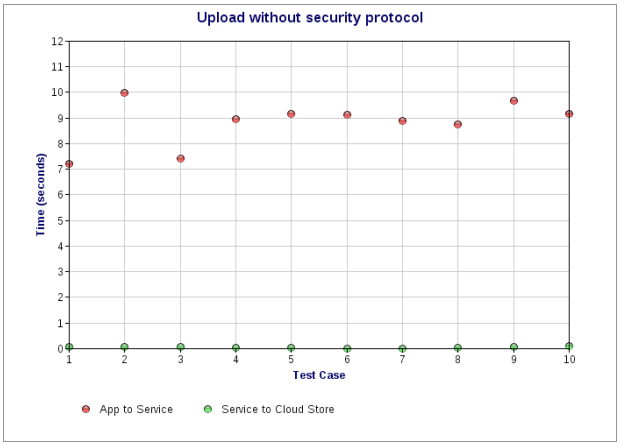
We carried out a number of performance tests of our system, primarily the uploading and downloading of ECG data. The purpose of the performance test was to test whether such a system will be feasible to be used by everyday people. Each of the performance tests were carried out on 10 seconds of ECG data or 3000 samples of ECG data points. For the tests, we

used an ASUS Eee Pad Transformer Prime TF201 to run the app, and a dual-core ASUS laptop with 2GB memory, 350GB storage and Windows Vista operating system.

For the uploading tests, we measured how long it would take for a patient to upload their ECG data to the Cloud. We measured the time it took from the moment the patient presses the upload button on their app to the storage of data in the database. We carried out 10 test cases and for each test case, we measured the time it took for the patient request to reach the Cloud service and then from service to Cloud storage. Also, we carried out the test cases using the secure data sharing protocol and then again without the secure data sharing protocol. Our results are shown in Figure 6.



(a) Uploading times with security protocol



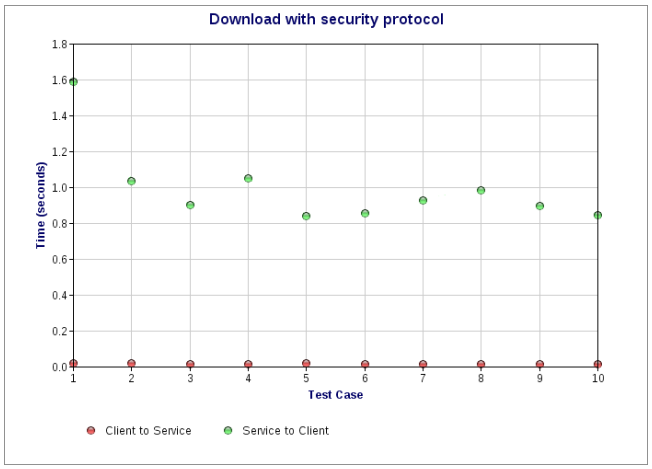
(b) Uploading times without security protocol

Figure 6: Uploading times

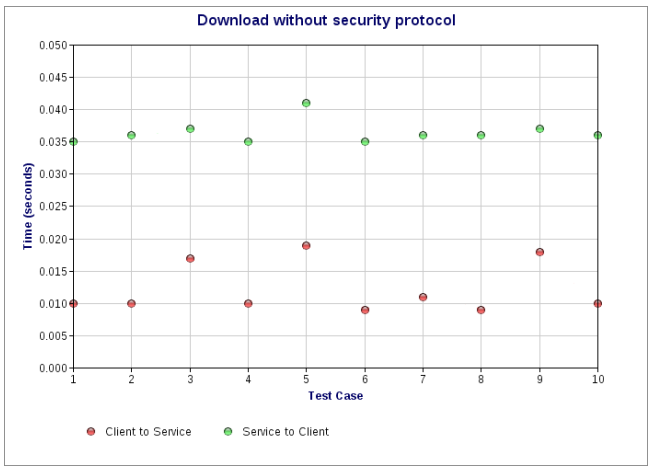
We found that in both cases using the security protocol and without, that it took significantly longer for the app to send the data to the Cloud service compared to the service storing the data to Cloud storage. This is expected since the mobile device first carries out operations on the data and then needs to transmit and connect wirelessly with the Cloud service. With the security

protocol in place, the delay is even longer and this is due to the keys being generated and data being encrypted before it is sent to the Cloud.

For the downloading tests, we measured how long it would take a patient to retrieve their data. We also carried out 10 test cases and for each test case, we measured the time it took for the patient request to reach the Cloud service and then the time it took to retrieve the data and return it to the patient. Similar to our uploading tests, we carried out the test cases using the secure data sharing protocol and again without the secure data sharing protocol. Figure 7 illustrates our test case results.



(a) Downloading times with security protocol

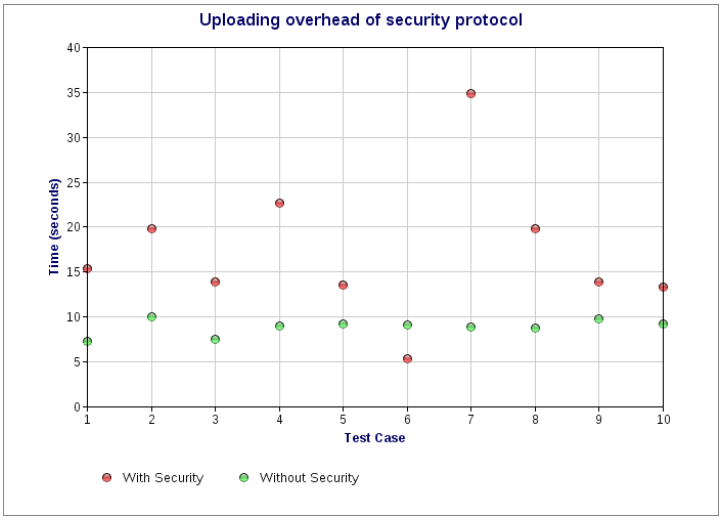


(b) Downloading times without security protocol

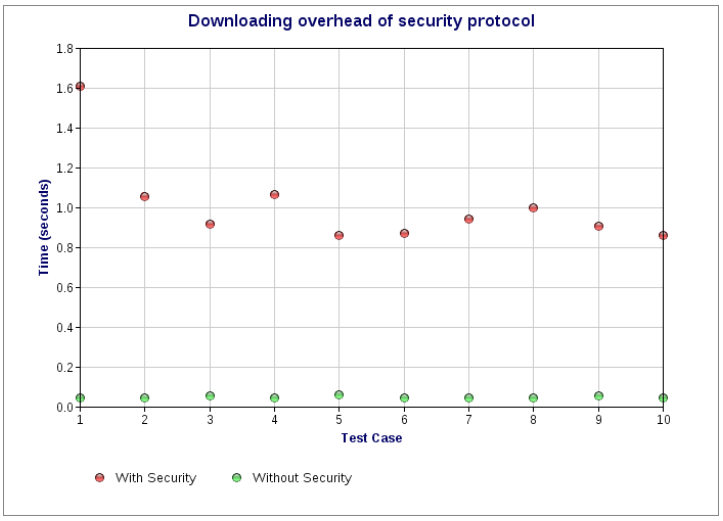
Figure 7: Downloading times

From our results, we found that this time it was much quicker for the patient request to reach the Cloud service. Although, the time taken to retrieve data from Cloud storage and return to user was slightly longer, it still returned the results in less than 1 second.

The total uploading and downloading times are highlighted in Figure 8. We measure the overhead introduced with the security protocol in place compared to a system with no security mechanims whatsoever.



(a) Uploading overhead



(b) Downloading overhead

Figure 8: Uploading and Downloading Overhead

From our results, the security protocol introduced significantly more overhead compared to without it. We also found that uploading times in general took a lot longer compared to download times when the security protocol was in place. Without the security protocol, the difference was negligible. The average time taken to upload data to the Cloud was 17.27 seconds with a standard deviation of 7.40 seconds with the security protocol in place. Without the security protocol in

place the average time was 8.89 seconds with a standard deviation of 8.92 seconds. With the security protocol in place, retrieval times took around 1-2 seconds. With the security protocol in place, the average time of download was 1.00 second with a standard deviation of 0.21 seconds and without the security protocol, the average was 0.05 seconds with a standard deviation of 0.01 seconds.

6.4. Evaluation

From the performance tests, uploading took much more time on average than downloading times. The main reason uploading times took longer was due to the fact that the a mobile device was used to send large packets of data. The patient retrieves data on a desktop PC and since the PC has much powerful processing capabilities than a mobile device, processing times were much quicker. During our testing, the downloading client and the web service were running on one machine which also explains our quicker running times when downloading. Also, the data is packaged in an XML document and is sent to the web service via SOAP which may have contributed to slow uploading times. Although uploading times could be improved, it is not crucial to the feasibility of the system since a patient does not usually care how long the data takes to store in the database. However, patients, and especially doctors, may be concerned regarding the time it takes to download data from the database as they may not want to wait long periods of time to retrieve ECG data for analysis. From the performance tests, the downloading times were very small and hence efficient making our system feasible to be deployed in the real world.

Also, when comparing the overhead of our security protocol, only the uploading times had a significant overhead. Downloading times were similar and almost negligible. The uploading times took longer due to the symmetric key generation and the encrypting of the data with it, as well as the encryption of the key itself with the ElGamal public key. However, patients will not usually care about uploading times, as they will mainly be in a resting position as their ECG is being monitored. When downloading data, it is more important that data arrives quickly such that doctors can provide more rapid treatment options. Hence, this makes our solution feasible to be used in the real world.

To further improve our performance tests, we could run test cases for different sizes of ECG data, e.g 30 second intervals, 1 minute intervals, 5 minute intervals and so on. Since we are still prototyping the system and not yet using a Cloud database, we rely on MySQL as our database, which does not handle very large data string sizes.

7. Conclusion

Data sharing applications in the Cloud is an exciting area and is fast becoming feasible in the near future. As more and more applications enter the health domain, there is now increasing demand to use the Cloud for health-related purposes such as for the sharing of health information. This places strong demands for data privacy and security in the Cloud. In this paper, we have developed a secure data sharing model and protocol that will allow private and secure sharing of data in the Cloud. Our secure data sharing protocol addresses the user revocation problem and the handling of large data sizes. The feasibility of the protocol has been demonstrated through our developed prototype that allows remote health-monitoring via the Cloud. We then carried out an analysis on the security aspects of our protocol. We also carried out performance tests on our developed prototype to test for feasibility. The overhead introduced by our security protocol

in our prototype was a little longer in uploading due to the cryptography algorithms and key management operations, however we found this to be efficient as upload times were mainly insignificant to the user. Download times were comparatively low and hence makes our solution particularly attractive to use and share data.

Currently, in our secure data sharing model and protocol, we assume the DSS to be a fully trusted party. As part of our future work, we aim to remove this assumption and treat the DSS as an untrusted party. We also plan to continue to improve our developed app. Also as part of our future work, we plan to include and perform usability tests. In the future, we plan to include the monitoring of other health services such as glucose levels, pulse and temperature to name a few.

References

[1] Bluetooth Technology Web Site http://www.bluetooth.com/Pages/Bluetooth-Home.aspx

[2] National Academy of Engineering http://www.engineeringchallenges.org/cms/8996/8938.aspx

[3] Wayne Swan MP (January 2010): Australia to 2050: future challenges.Intergeneration Report. Source: http://archive.treasury.gov.au/igr/igr2010/report/pdf/IGR_2010.pdf

[4] O2 health takes telecare mobile. (2012, Mar 08). M2 Presswire.

[5] Bosch selects doro to enrich TeleCare offering with mobile solutions. (2012, Nov 15). M2 Presswire.

[6] Numera; numera acquires BlueLibris and expands offerings into telecare. (2012). Network Weekly News, , 625.

[7] HealthTech Wire Interview (05/04/2012). Data Sharing is the key to lower healthcare costs. EMC Corporation. Source: http://www.healthtechwire.com/emc-corporation/data-sharing-is-the-key-to-lower-healthcare-costs-3132/

[8] Rocha F., Abreu S., Correia M.: The Final Frontier: Confidentiality and Privacy in the Cloud: 44-50. Publication Year: 2011

[9] 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 http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf

[10] Aramudhan, M.; Mohan, K.: "New Secure Communication Protocols For Mobile E-health System." International Journal of Computer Applications 8(4):1015, October 2010. Published By Foundation of Computer Science.

[11] Marti, R.; Delgado, J.; Perramon, X., "Security specification and implementation for mobile e-health services," e-Technology, e-Commerce and e-Service, 2004. EEE '04. 2004 IEEE International Conference on , vol., no., pp.241,248, 28-31 March 2004

[12] Michael Healey, Information Week. (2010, Jun 19) Why IT Needs To Push Data Sharing Efforts. Source: http://www.informationweek.com/services/integration/why-it-needs-to-push-data-sharing-effort/225700544

[13] Riley, D. A. (2010). Using google wave and docs for group collaboration. Library Hi Tech News, 27(4), 12-14. doi: http://dx.doi.org/10.1108/07419051011083181

[14] Gellin, A. (2012, Feb 23). Facebook's benefits make it worthwhile. Buffalo News.

[15] Wu, R. (2012). Secure sharing of electronic medical records in cloud computing. Arizona State University). ProQuest Dissertations and Theses, , 77.

[16] Sarathy R., Muralidhar K.: Secure and useful data sharing (2006) Decision Support Systems, 42 (1), pp. 204 220.

[17] Butler, Declan: Data sharing threatens privacy. Nature Publishing Group. Issue 7163, Volume 449. Pages 644 645. DOI 10.1038/449644a

[18] Feldman, Lindsay; Patel, Deesha; Ortmann, Leonard; Robinson, Kara; Popovic, Tanja: Educating for the future: another important benefit of data sharing. The Lancet, Volume 379, Issue 9829, 19-25 May 2012, Pages 1877 1878

[19] Tran D.H, Hai-Long Nguyen, Wei Zha, Wee Keong Ng: Towards Security in Sharing Data on Cloud-Based Social Networks. Information, Communications and Signal Processing(ICICS) 2011 8th International Conference: 1 - 5

[20] Nguyen Thanh Hung, Do Hoang Giang, Ng Wee Keong, Huafei Zhu: Cloud-Enabled Data Sharing Model. Intelligence and Security Informatics(ISI), 2012 IEEE International Conference: 1 - 6

[21] Vipul Goyal, Omkant Pandey, Amit Sahai, Brent Waters: Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. 13th ACM Conference on Computer and Communications Security(CCS 06) 2006: 89 - 98

[22] Jin Li, Gansen Zhao, Xiaofeng Chen, Dongqing Xie, Chunming Rong, Wenjun Li, Lianzhang Tang, Yong Tang: Fine-grained Data Access Control Systems with User Accountability in Cloud Computing. Cloud Computing Technology and Science(CloudCom), 2010 IEEE Second International Conference: 89 -96

[23] Shan-shan Tu; Shao-zhang Niu; Hui Li; Yun Xiao-ming; Meng-jiao Li; , "Fine-grained Access Control and Revocation for Sharing Data on Clouds," Parallel and Distributed Processing Symposium Workshops

& PhD Forum (IPDPSW), 2012 IEEE 26th International , vol., no., pp.2146-2155, 21-25 May 2012 doi: 10.1109/IPDPSW.2012.265

[24] Li, Ming; Yu, Shucheng; Zheng, Yao; Ren, Kui; Lou, Wenjing; , "Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption," Parallel and Distributed Systems, IEEE Transactions on , vol.24, no.1, pp.131-143, Jan. 2013

[25] Yanjiang Yang, Youcheng Zhang: A Generic Scheme for Secure Data Sharing in Cloud. Parallel Processing Workshops(ICPPW) 2011 40th International Conference: 145 - 153

[26] Nguyen Thanh Hung, Do Hoang Giang, Ng Wee Keong, Huafei Zhu: Cloud-Enabled Data Sharing Model. Intelligence and Security Informatics(ISI), 2012 IEEE International Conference: 1 6

[27] G. Fortino, M. Pathan, G. Di Fatta: BodyCloud: Integration of Cloud Computing and Body Sensor Networks. In Proc. of IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom 2012), Taipei, Taiwan, Dec 3-6, 2012.

[28] F. Bellifemine, G. Fortino, R. Giannantonio, R. Gravina, A. Guerrieri, M. Sgroi: SPINE: A domain-specific framework for rapid prototyping of WBSN applications. Software Practice and Experience, Wiley, 41(3), 2011, pp. 237-265, DOI:10.1002/spe.

[29] G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski, R. Jafari: Enabling Effective Programming and Flexible Management of Efficient Body Sensor Network Applications. IEEE Transactions on Human-Machine Systems, vol. 43, no. 1, pp. 115–133, Jan. 2013.

[30] Suraj Pandey, William Voorsluys, Sheng Niu, Ahsan Khandoker, Rajkumar Buyya: An Autonomic Cloud Environment for Hosting ECG Data Analysis Services. Future Generation Computer Systems v.28 n.1: 147 - 151. January 2012

[31] NIST Definition of Cloud Computing csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[32] Giniat, E. J. (2011). Cloud computing: Innovating the business of health care. Healthcare Financial Management, 65(5), 130-1.

[33] IDC Cloud Research http://www.idc.com/prodserv/idc_cloud.jsp

[34] HIPAA. U.S. Department of Health and Human Services http://www.hhs.gov/ocr/privacy/index.html

[35] Saarinen, Juha (Aug 7, 2012). UK health trust fined for privacy breach. Itnews Technology News. Source: http://www.itnews.com.au/News/311079,uk-health-trust-fined-for-privacy-breach.aspx

[36] SeongHan, Shin, Kobara, K., Imai, H.: A Secure Public Cloud Storage System. Internet Technology and Secured Transactions(ICITST), 2011 International Conference: 103 -109

[37] Minqi Zhou, Rong Zhang, Wei Xie, Weining Qian, Aoying Zhou: Security and Privacy in Cloud Computing: A Survey. Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference: 105-112

[38] Duncan, A.J.; Creese, S.; Goldsmith, M.; , "Insider Attacks in Cloud Computing," Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on , vol., no., pp.857-862, 25-27 June 2012 doi: 10.1109/TrustCom.2012.188

[39] RuWei Huang, XiaoLin Gui, Si Yu, Wei Zhuang: Research on Privacy-Preserving Cloud Storage Framework Supporting Ciphertext Retrieval. 2011 International Conference on Network Computing and Information Security: 93-97

[40] AliveCor website. Source: http://www.alivecor.com

[41] AliveCor(TM) mobile ECG device for heart rhythm monitoring now available by prescription. (2013, Mar 08). Business Wire.

[42] CardioComm solutions, inc. reveals a new remote mobile ECG monitoring solution at medica 2011. (2011, Nov 22). Business Wire.

[43] Gradl, S.; Kugler, P.; Lohmuller, C.; Eskofier, B., "Real-time ECG monitoring and arrhythmia detection using Android-based mobile devices," Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE , vol., no., pp.2452,2455, Aug. 28 2012-Sept. 1 2012

[44] Henian Xia, Irfan Asif, Xiaopeng Zhao, Cloud-ECG for real time ECG monitoring and analysis, Computer Methods and Programs in Biomedicine, Volume 110, Issue 3, June 2013, Pages 253-259, ISSN 0169-2607, 10.1016/j.cmpb.2012.11.008.

[45] Nick Galvin: File-sharing service users in cloud over access to data. The Age. 23rd January 2012.

[46] Deyan Chen; Hong Zhao; , "Data Security and Privacy Protection Issues in Cloud Computing," Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on , vol.1, no., pp.647-651, 23-25 March 2012 doi: 10.1109/ICCSEE.2012.193

[47] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In Proceedings of CRYPTO 84 on Advances cryptology, 1985.

[48] Xu An Wang, Weidong Zhong: A New Identity Based Proxy Re-encryption Scheme. Biomedial Engineering and Computer Science(ICBECS) 2010 International Conference: 1 - 4

[49] ASUS Eee Pad Transformer Prime TF201 http://www.asus.com/Eee/Eee_Pad/Eee_Pad_Transformer_Prime_TF201/

[50] Alive Technologies http://www.alivetec.com/index.htm

[51] Ambu Blue Sensor ECG Electrodes http://www.ambu.com/corp/products/clinical_studies/ambu_blue_sensor_ecg_electrodes.aspx
[52] Electrocardiography. WikiPedia definition. http://en.wikipedia.org/wiki/Electrocardiography