University of Sydney

Doctoral Thesis

Secure Data Sharing and Collaboration in the Cloud

Author:
Danan Thilakanathan

Supervisors:
Professor Rafael A. Calvo, Dr.
Shiping Chen

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Faculty of Engineering and Information Technologies
School of Electrical and Information Engineering

June 2016

UNIVERSITY OF SYDNEY

Abstract

Faculty of Engineering and Information Technologies
School of Electrical and Information Engineering

Doctor of Philosophy

Secure Data Sharing and Collaboration in the Cloud

by Danan Thilakanathan

Cloud technology can be leveraged to enable data-sharing capabilities, which can benefit the user through greater productivity and efficiency. However, the Cloud is susceptible to many privacy and security vulnerabilities, which hinders the progress and widescale adoption of data sharing for the purposes of collaboration. Thus, there is a strong demand for data owners to not only ensure that their data is kept private and secure in the Cloud, but to also have a degree of control over their own data contents once they are shared with data consumers.

Specifically, the main issues for data sharing in the Cloud include key management, security attacks, and data-owner access control. In terms of key management, it is vital that data must first be encrypted before storage in the Cloud, to prevent privacy and security breaches. However, the management of encryption keys is a great challenge. The sharing of keys with data consumers has proven to be ineffective, especially when considering data-consumer revocation. Security attacks may also prevent the widescale usage of the Cloud for data-sharing purposes. Common security attacks include insider attacks, collusion attacks, and man-in-the-middle attacks. In terms of access control, authorised data consumers could do anything they wish with an owner's data, including sending it to their peers and colleagues without the data owner's knowledge.

Throughout this thesis, we investigate ways in which to address these issues. We first propose a key partitioning technique that aims to address the key management problem. We deploy this technique in a number of scenarios, such as remote healthcare management. We also develop secure data-sharing protocols that aim to mitigate and prevent security attacks on the Cloud. Finally, we focus on giving the data owner greater control, by developing a self-controlled software object called SafeProtect.

Publications

Book Chapter

- **Danan Thilakanathan**, Shiping Chen, Surya Nepal, Rafael A. Calvo: Secure data sharing in the Cloud. Book chapter in Security, Privacy, and Trust in Cloud Systems, by Springer (2013): 45 - 72. ISBN: 978-3-642-38585-8. Source: http://dx.doi.org/10.1007/978-3-642-38586-5_2

Journals

- **Danan Thilakanathan**, Shiping Chen, Surya Nepal, Rafael A. Calvo, Leila Alem (2012): A Platform for Secure Monitoring and Sharing of Generic Health Data in the Cloud. Special Issue on Integration of Cloud Computing and Body Sensor Networks, Future Generation Computer Systems (FGCS): 102 - 113. **(Impact Factor = 2.639)**
- **Danan Thilakanathan**, Shiping Chen, Surya Nepal, Rafael A. Calvo (2015): SafeProtect: Controlled Data Sharing with User-Defined Policies in Cloud-based Collaborative Environment. IEEE Transactions on Emerging Topics in Computing: 1 - 1.
- **Danan Thilakanathan**, Rafael A. Calvo, Shiping Chen, Surya Nepal, Nick Glozier (2016): Facilitating Secure Sharing of Personal Health Data in the Cloud (JMIR Medical Informatics). **(Impact Factor = 3.428) (In-Press)**

Conferences

- **Danan Thilakanathan**, Shiping Chen, Surya Nepal, Rafael A. Calvo (2013): Secure and Controlled Sharing of Data in Distributed Computing. 2nd IEEE International Conference on Big Data Science and Engineering (BDSE 2013): 825 - 832.
- **Danan Thilakanathan**, Yu Zhao, Shiping Chen, Surya Nepal, Rafael A. Calvo, Abelardo Pardo (2014): Protecting and Analysing Health Care Data on Cloud. Second International Conference on Advanced Cloud and Big Data (CBD 2014): 143 - 149.

- **Danan Thilakanathan**, Shiping Chen, Surya Nepal, Dongxi Liu, Rafael Calvo, John Zic (2014): Secure Multiparty Data Sharing in the Cloud using Hardware-based TPM Devices. IEEE International Conference on Cloud Computing (IEEE Cloud 2014): 224 - 231. **(Acceptance rate: 18%)**

- Shiping Chen, **Danan Thilakanathan**, Donna Xu, Surya Nepal, Rafael A. Calvo (2015): Self Protecting Data Sharing using Generic Policies. 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2015): 1197 - 1200. **(ERA Ranking: A) (Best Poster Award)**

- Surya Nepal, Shiping Chen, Jinhui Yao, **Danan Thilakanathan** (2011): DIaaS: Data Integrity as a Service in the Cloud. IEEE CLOUD 2011: 308-315.

- Weidong Huang, Leila Alem, Surya Nepal, **Danan Thilakanathan**: Supporting tele-assistance and tele-monitoring in safety-critical environments. Australian Conference on Human-Computer Interaction (OZCHI 2013): 539 - 542.

Acknowledgements

I would like to acknowledge my supervisors, Dr Shiping Chen, Professor Rafael A. Calvo, and Dr Surya Nepal, for their tremendous support and encouragement throughout my degree. Their words of advice helped me to learn a great deal and have been very valuable in helping me to progress.

I would also like to thank the School of Electrical and Information Engineering, The University of Sydney, and the Digital Productivity Flagship, CSIRO, for giving me the opportunity to complete my PhD degree and for providing me with support through the APA scholarship and the CSIRO Top-Up Scholarship.

Finally, I would like to thank my mum, dad, sister and friends, who I am grateful to have in my life and who have constantly kept me motivated with their encouraging words of support.

Contents

List of Figures

List of Tables

Abbreviations

DO	**D**ata **O**wner
DC	**D**ata **C**onsumer
DSS	**D**ata **S**haring **S**ervice
KS	**K**ey **S**ervice
CDS	**C**loud **D**ata **S**ervice
DKDB	**D**ata **K**ey **D**ata**B**ase
UKDB	**U**ser **K**ey **D**ata**B**ase
CSP	**C**loud **S**torage **P**rovider
SSA	**S**afe**S**hare **A**pplication
ACP	**A**ccess **C**ontrol **P**olicy
TED	**T**rusted **E**xtension **D**evice
TTI	**T**rusted **T**ED **I**ssuer
PCA	**P**rivate **C**ertifying **A**uthority
SPO	**S**elf **P**rotecting **O**bject

This thesis is dedicated to my parents, my sister Cynthuja and my grandfather.

Chapter 1

Introduction

Cloud computing is a rapidly growing technology trend in the world today, and nearly all Internet users and enterprises alike use the Cloud, in some form or another [1]. Ever since Cloud computing began to evolve from the early days of grid and utility computing [2], its popularity has grown immensely. One of the main reasons for the Cloud's popularity is that it allows for the ability to access data anywhere, and at any time. The most popular usage of the Cloud includes: viewing emails via Hotmail or Gmail; writing, sharing and collaborating on documents via Google Docs; developing applications via the Google App Engine; and storing files via Amazon S3, Google Drive or Windows SkyDrive. Amazon EC2/S3 was one of the first widely available Cloud-computing services [2]. 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, and at any time; high availability, as data is usually replicated among a number of servers; lowering the chance of data loss; and providing elasticity, whereby more computing resources can be used when required [3]. Cloud computing also provides benefits to many fields, such as healthcare [4]. There is also a strong push by

IT organisations to share their data with one another, which is achieved via the Cloud. Thus, Cloud technology is very popular amongst both end users and organisations.

The International Organization for Standardization (ISO) [5] defines Cloud computing as a "paradigm for enabling network access to a scalable and elastic pool of shareable physical or virtual resources with on-demand self-service provisioning and administration" To better understand the problem, it is important to have an understanding of Cloud computing. Thus, we provide the essential characteristics, types of services and deployment methods, below.

ISO also defines five essential characteristics of the Cloud:

- **Broad network access** - A feature where the physical and virtual resources are available over a network and accessed through standard mechanisms that promote use by heterogeneous client platforms. The focus of this key characteristic is that cloud computing offers an increased level of convenience in that users can access physical and virtual resources from wherever they need to work, as long as it is network accessible, using a wide variety of clients including devices such as mobile phones, tablets, laptops, and workstations.

- **Measured Service** - A feature where the metered delivery of cloud services is such that usage can be monitored, controlled, reported, and billed. This is an important feature needed to optimize and validate the delivered cloud service. The focus of this key characteristic is that the customer may only pay for the resources that they use. From the customers' perspective, cloud computing offers the users value by enabling a switch from a low efficiency and asset utilization business model to a high efficiency one.

- **Multi-tenancy** - A feature where physical or virtual resources are allocated in such a way that multiple tenants and their computations and data are isolated from and inaccessible to one another. Typically, and within the context of multi-tenancy, the group of cloud service users that form a tenant will all belong to the same cloud service customer organization. There might be cases where the group of cloud service users involves users from multiple different customers, particularly in the case of public cloud and community cloud deployments. However, a given cloud service customer organization might have many different tenancies with a single cloud service provider representing different groups within the organization

- **On-demand self-service** - A feature where a cloud service customer can provision computing capabilities, as needed, automatically or with minimal interaction with the cloud service provider. The focus of this key characteristic is that cloud computing offers users a relative reduction in costs, time, and effort needed to take an action, since it grants the user the ability to do what they need, when they need it, without requiring additional human user interactions or overhead.

- **Rapid elasticity and scalability** - A feature where physical or virtual resources can be rapidly and elastically provisioned, in some cases automatically, to quickly increase or decrease resources. For the cloud service customer, the physical or virtual resources available for provisioning often appear to be unlimited and can be purchased in any quantity at any time, subject to constraints of service agreements. Therefore, the focus of this key characteristic is that cloud computing means that the customers no longer need to worry about limited resources and might not need to worry about capacity planning. From the customers' perspective, if new resources are needed, they are available automatically, immediately, and can appear to be infinite, subject to constraints of service agreements.

- **Resource pooling** - A feature where a cloud service provider's physical or virtual resources can be aggregated in order to serve one or more cloud service customers. The focus of this key characteristic is that cloud service providers can support multi-tenancy while at the same time using abstraction to mask the complexity of the process from the customer. From the customer's perspective, all they know is that the service works, while they generally have no control or knowledge over how the resources are being provided or where the resources are located. This offloads some of the customer's original workload, such as maintenance requirements, to the provider. Even with this level of abstraction, it should be pointed out that users might still be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter).

The Cloud provides a number of different types of services, and we list the three most common ones as follows:

- **Software as a Service (SaaS)** - The consumer can choose to run the application on demand but has no control of the underlying application. The Cloud provider has full control over the application.
- **Platform as a Service (PaaS)** - This allows the consumer to deploy consumer-created and/or acquired applications created using programming languages, libraries, services and tools supported by the Cloud provider [3]. The Cloud provider has control over the platform and its applications. The Cloud provider has full control of the hardware, while the client has control over the application.
- **Infrastructure as a Service (IaaS)** - Basic computing infrastructure such as servers, storage, software, processing and network equipment is provided as a service to Cloud consumers.

There are four different deployment models for the Cloud [6]. These include:

- **Private cloud** - The Cloud infrastructure is owned, operated and managed within an organisation. Only entities within the organisation are provisioned for using the services provided by the Cloud. The servers may exist on or outside the premises of the organisation.
- **Community cloud** - The Cloud infrastructure is provisioned for use by a community of users from various organisations who share common goals and concerns.
- **Public cloud** - The Cloud infrastructure is provisioned for use by the public. The servers generally exist on the Cloud provider's premises.
- **Hybrid cloud** - A composition of two or more of the above deployment models.

1.1 Motivation

Data sharing is becoming increasingly important for end users, enterprises and even health industries. There is currently a push for IT organisations to increase their data-sharing efforts. According to a survey by Information Week [1], nearly all organisations shared their data in some way, with 74% sharing their data with customers and 64% sharing it with suppliers. A fourth of the surveyed organisations consider data sharing to be a top priority. The benefits that organisations can gain from data sharing are higher productivity and revenue. People in general love to share information with one another. Whether it is with friends or family, or even strangers around the world, many people benefit greatly through sharing data, as they achieve higher levels of productivity and a better education. The National Institute of Health (NIH) states that data sharing "reinforces open scientific inquiry, encourages diversity of analysis and opinion, promotes

new research, makes possible the testing of new or alternative hypotheses and methods of analysis, supports studies on data collection methods and measurement, facilitates the education of new researchers, enables the exploration of topics not envisioned by the initial investigators, and permits the creation of new datasets when data from multiple sources are combined" [7]. Data sharing can also help businesses understand more about their customers and provide better services for them [8]. Thus, data sharing is very crucial among users from all walks of life, regardless of their age, gender, race, or whether they work in industry.

The Cloud [3] can be used to enable data-sharing capabilities, which can provide an abundant number of benefits to the user. With multiple users from different organisations contributing to data in the Cloud, less time and money will be spent, compared with having to manually exchange data that creates a clutter of redundant and possibly out-of-date documents. Thus, the Cloud makes data sharing with anyone in the world both more convenient and easier than any other method of sharing.

Some benefits of data sharing in the Cloud include more data reliability and availability, the ability to work from anywhere, not being constrained to one machine, no server maintenance, and saved costs to name a few [9].

There are many examples of Cloud usage in today's world, which brings with it a huge number of benefits. There are social networking services such as Facebook and Twitter. The benefits of sharing data through social networks are numerous [10], such as the ability to share photos, videos, information and events, which creates a sense of enhanced enjoyment in one's life and can enrich the lives of some people who are amazed at how many people are interested in their life and well-being. The sharing of research data has been shown to benefit the general scientific community [11]. Group collaborative

tools have been of major importance to students and group-related projects [12]. Google Docs provides data-sharing capabilities, whereby groups of students or teams working on a project can share documents and effectively collaborate with each other. This allows for higher productivity than the previous method of continually sending updated versions of a document to members of the group via email attachments. Also, in modern healthcare environments, healthcare providers are willing to store and share electronic medical records via the Cloud and hence remove the geographical dependence between the healthcare provider and the patient [13]. The sharing of medical data allows for the remote monitoring and diagnosis of patients, without the patient having to leave their home.

The Cloud, however, is susceptible to many privacy and security attacks [14]. As highlighted in [15], the biggest obstacles hindering the progress and wide adoption of the Cloud are the privacy and security issues associated with it. According to a survey carried out by IDC Enterprise Panel [16] in August 2008, Cloud users regard security as the top challenge, with 75% of surveyed users worried about their critical business and IT systems being vulnerable to attack. Once the data owner stores their data on the Cloud, they effectively lose full control over their data. The Cloud Service Provider is able to do whatever they wish with the data, without the data owner's knowledge. Evidently, many privacy and security attacks occur from within the Cloud providers themselves [17], as providers usually have direct access to stored data and steal data to sell to third parties, in order to gain profits. There are many examples of this happening in the real world, as highlighted in [18]. In today's world, there is a strong need to share information to groups of people around the world. Since the Cloud is riddled with so many privacy issues, many users are still apprehensive about sharing their most critical data with other users. Such threats also affect the trust of the data owner.

We now list some of the key requirements of secure data sharing in the Cloud, which we derived by reflecting and reviewing the literature in Chapter 2.

- Firstly, the data owner should be able to specify a group of users that are allowed to view his or her data.
- Any member within the group should be able to gain access to the data at any time, and anywhere, without the data owner's intervention.
- The data owner should have some degree of control over their own data, no matter where the data is stored.
- No-one other than the data owner and the members of the group should be able to gain access to the data, including the Cloud Service Provider.
- The data owner should be able to add new users to the group.
- The data owner should also be able to revoke the access rights of any member of the group to his or her shared data.
- No member of the group should be allowed to revoke rights or join new users to the group.

1.2 Key Research Problems

Our research aim is to achieve an environment where the data owner is able to store and share data in the Cloud while maintaining both privacy and security. A few of the major privacy and security issues related to data sharing in the Cloud include:

- **Key Management:** One of the main issues related to data sharing in the Cloud is key management. One trivial solution to achieving secure data sharing in the

Cloud is for the data owner to encrypt his data before storing it on the Cloud, and hence the data remains secure against the Cloud provider and other malicious users. When the data owner wants to share his data to a group, he sends the key used for data encryption to each member of the group. Any member of the group can then obtain the encrypted data from the Cloud and decrypt the data using the key, thereby not requiring the intervention of the data owner. However, the problem with this technique is that it is computationally inefficient and places too great a burden on the data owner, when considering factors such as user revocation. When the data owner revokes access rights to a member of the group, that member should not be able to gain access to the data. Since the member still has the data access key, the data owner has to re-encrypt the data with a new key, rendering the revoked member's key useless. When the data is re-encrypted, he must distribute the new key to the remaining users in the group, which is computationally inefficient and places too great a burden on the data owner, when considering large group sizes. Therefore, this solution is impractical for deployment in the real world, with regards to highly critical data such as business-, government- and/or medical-related data. Thus, there needs to be a solution whereby encryption keys are managed on/off the Cloud, which does not place a significant burden on the data owner/consumer.

Note that the data consumer still has access to the data already decrypted before he was revoked access. Although he can still view the old data, he can no longer view any new data shared within the group or data that had yet to be accessed before he was revoked.

- **Security Attacks when Sharing Data:** There are many different types of security attacks when sharing data in the Cloud. We describe a few of the most

common types of attacks, below:

- *Insider Attacks:* This is the most prevalent type of attack in relation to data sharing in the Cloud [17]. Cloud providers have full and direct access to the data owner's data and are thus able to steal critical data without the data owner's permission. Insider attacks are common and the stolen data are generally sold to third parties in order to gain profit [19].
- *Sniffing Attacks:* A malicious user intercepts the public communication network and attempts to retrieve sensitive information such as a username, password, and critical data. Such attacks are likely to be successful when there are no forms of encryption, when data is sent over public communication channels. Thus, it is crucial to ensure that data is always encrypted before sending it over the network. Man-in-the-middle (or Man-in-the-Cloud) based attacks can steal data stored in Cloud services such as Google Drive, Dropbox and OneDrive, to name a few, by stealing the user's synchronisation token [20].
- *Collusion Attacks:* These types of attacks occur when two or more parties agree to reveal some secret information, illegally. For instance, a Cloud provider and a dishonest data-sharing consumer can combine their secret keys to reveal some critical information about the data owner's data. This can affect the entire privacy and security of all of the data stored in the Cloud, thus severely affecting the trust of the data owner with regards to using the Cloud for data sharing and collaboration purposes. Collusion attacks are mainly prevalent in eCommerce services [21].

Consequently, security protocol(s) need to be in place to protect against these types of attacks.

- **Dishonest Data Consumers:** Since the data owner has chosen to entrust his specified data consumers with his data, the data owner assumes that the data consumer will use the data as expected and will not inadvertently or accidentally leak the data to outsiders. For instance, a dishonest data consumer might find the data owner's data interesting enough to decide to share it with his friends. He might copy the data to a USB device and send it to friends, or send the data via email attachments. This can be easily done without the knowledge of the data owner, which affects the trust of the data owner when their own consumers are the culprits of the data leakage. Thus, there needs to be some form of mechanism that provides the data owner with greater access control over his data. The data owner should be able to specify complex policies that data consumers must adhere to when using the data.

1.3 Key Contributions

To address the above specific issues, we make the following contributions in this thesis.

1.3.1 Key Partitioning Algorithm

As discussed, the main problem with encryption-key management related to data sharing in the Cloud is that it is computationally inefficient and places too heavy a burden on the data owner to manage an authorised group of users. This problem is especially true when considering factors such as user revocation, or managing large groups of users in excess of thousands to hundreds of thousands.

A few studies have attempted to address this problem, with the most popular techniques being Attribute-Based Encryption (ABE) [22] and Proxy Re-encryption [23]. Tu and Niu et al. [24] have used a technique called CP-ABE (discussed in detail in the next chapter), which bundles unique attributes in a secret key and sends the key to authorised users. In terms of user revocation, the data is re-encrypted in the Cloud, which reduces the burden on the data owner. Tran et al. [25] have proposed a proxy re-encryption scheme whereby a key is divided into two parts, with one partition kept by the consumer and the other by the Cloud proxy. This allows for a more efficient way to handle user revocation, as the data owner does not need to re-encrypt the data and re-distribute the keys.

The main problems with current solutions, however, are that they either do not handle the sharing of data with large numbers of Cloud users, or they lack a more efficient solution. For instance, the solution provided by Tran et al. [25] relies on the proxy to carry out encryption/decryption operations and might therefore be too much for the proxy to handle. Also, the solution does not handle the scenario where a revoked user and the proxy are colluding, which will ultimately reveal the full plaintext key and thus the data, as well. The solution of Tu and Niu et al. [24] is inelegant, since there is still a heavy computational overhead as the data needs to be re-encrypted when a user is revoked. The solution also lacks transparency in terms of sharing data with a variety of users around the world, and is more focused on sharing data with users who have defined attributes that are more suited to enterprises.

In our approach, we have extended the work of Tran et al. [25] to allow for more efficient key sharing amongst a large number of users. We propose a key partitioning cryptographic algorithm that allows the data owner to share data with many users and revoke them on-the-fly, without the need to undertake the re-encryption or re-distribution of

keys. Our solution is also secure against collusion-based attacks, and places no burden on the data owner or the Cloud to manage keys. To demonstrate our algorithm, we developed prototypes in a number of healthcare scenarios. We evaluated the prototypes in terms of performance, usability and scalability. This helped to demonstrate the feasibility of implementing a similar system in the real world.

1.3.2 Secure Data Sharing Protocol

One of the biggest issues preventing the widescale adoption of the Cloud for the sharing and collaboration of data, is privacy and security [14],[15]. Attacks can occur in transit, on Cloud storage and on backup media. In fact, the worst culprits are malicious Cloud providers themselves, as they have full, direct access to the data [17]. In order to keep data secure, it is thus crucial that data must be kept encrypted at all times, even when stored on the Cloud. However, encryption alone is not enough, as hackers and Cloud providers continue to find new ways to access data. This can occur through collusion-based attacks where the Cloud provider colludes with an authorised consumer to reveal some vital information about the data. As mentioned earlier, other forms of attack include sniffing attacks, accidental or intentional leaking of data by authorised consumers, or even over-the-shoulder attacks.

There is currently on-going research on how to protect the confidentiality and security of data stored in the Cloud. Jayalatchumy et al. [26] have proposed implementing security as a Cloud service using a discretion algorithm, and have implemented an intrusion detection system for the Cloud. Cavoukian [27] has argued the need for flexible and user-centric identity management so that in the future, a user would not have to re-enter credentials for a website and could rely on an identity service to manage website access. There has also been research on using access control as a form of security. Access

control provides restrictions on who can view the data and who cannot. Access Control Lists (ACLs) were originally used [28] for data protection. XACML [29] is a generic, open-source access control policy language built on XML allowing the data owner to specify complex policies governing how their data should be used. The ABC4Trust EU Project proposed the use of Privacy Attribute-Based Credentials (Privacy ABCs) [30] as a way to enhance the privacy of individual authentication without ever requiring them to reveal their full personal details and thus allowing their identity to be kept anonymous. The AU2EU project aims to implement an integrated eAuthentication and eAuthorisation framework for trusted collaborations and delivery of services by utilising identity/attribute providers and policy enforcement mechanisms to name a few [31].

However, effective protocols need to be in place to simultaneously prevent privacy and security breaches and reduce the burden on data owners to manage their groups of authorised consumers. For instance, although ACLs provide a degree of control over who can access the data, this is not effective, as it is too coarse-grained and unscalable, which is one of the primary features of the Cloud.

In this thesis, we develop secure data-sharing protocols that allow the data owner to efficiently and securely share and manage their data with many users. Through our protocols, we aim to improve the trust of the data owner in using the Cloud to share and collaborate data with other users. Our protocols prevent insider attacks, collusion-based attacks, sniffing/man-in-the- middle attacks, and a number of other common security attacks. We demonstrate our protocols through our software-based self-protecting data object called "SafeShare." We also carry out performance tests on on SafeShare. We later improve upon this work and propose a hardware-based self-protecting data object, using a physical device, the Trusted Extension Device (TED). We discuss the performance of this approach and ways to improve performance overall.

1.3.3 Self-Controlling Data Object

One of the growing problems related to data sharing in the Cloud is the data owner access-control problem. Once the data leaves the trusted premises of the data owner's machine, he no longer has any control over it. The data owner has no knowledge of where the data is stored, who has access to the data, or how many copies of the data are being kept. On top of the privacy and security issues discussed above, an authorised data consumer might inadvertently leak the data owner's data, either accidentally or intentionally. This can happen when an authorised data consumer uses the fully decrypted plaintext and sends it to their peers via email attachments and/or USB transfer.

Currently, the data owner has no knowledge of who is leaking the data, which makes auditing and accountability difficult. The data owner also has no control over preventing the data owner from leaking the data or carrying out an illegal operation on the data. There have not been many research projects that have focused on dishonest data consumers who might leak data either accidentally or intentionally. Squicciarini et al. [32] have proposed the idea of a Self-Controlling Object (SCO), which bundles data and access control policy in a JAR file. The JAR file includes an executable that logs every operation of the data consumer and periodically flushes the log file to the Cloud, in order for the data owner to access and subsequently hold the offending data consumers accountable. The main issue with this solution is that management of the SCOs is tedious and still does not prevent a dishonest consumer from leaking data; it only reports the leak to the Cloud. Chen et al. [33] have proposed DataSafe, which similarly bundles the data and policy in a file, with data access occurring on DataSafe machines. The issue with this solution, however, is that it requires specific hardware and thus would

not work with the Cloud. This is due to the Cloud feature that data can be accessed from anywhere, at any time.

We introduce the idea of self-protecting data objects. Self-protecting data objects are a fairly new contribution to research, and aim to provide the data owner with greater access control over their data. With self-protecting data objects, the data owner now has some level of control over their data, even if they store their data on the Cloud. This can help to increase the data owner's level of trust when sharing data. We demonstrate this idea through our proposed policy-based self-protecting data object called "SafeProtect Object." We develop the object as well as a plugin to Microsoft Word which can be used to access the object to demonstrate the feasibility of our idea. This work was well recognised and achieved the Best Poster Award in CCGrid 2015.

1.4 Thesis Organisation

In this thesis, we focus on enabling private and secure data sharing in the Cloud, while providing the data owner greater access control over their data. By enabling greater access control, the data owner has greater confidence and trust in storing his data in the Cloud, without worrying about sensitive data leakage by Cloud insiders and/or authorised data consumers. Throughout this thesis, we tailor solutions that specifically address the problems of key management in terms of user revocation and the need for greater data-owner access control.

- In Chapter 2, we review the existing literature on the methods for achieving private, secure and efficient data sharing in the Cloud.

- In Chapter 3, we provide a formal description of the problem statement and describe our trust model.

- In Chapter 4, we describe and detail our solution to the problem of key management, in terms of user revocation in the Cloud. This solution will enable data owners to share data with very large groups of data consumers, whilst providing the data owner with the low-burden ability to revoke data consumers. Our solution uses a key partitioning technique, which splits the encryption key and distributes partitions between consumers and the Cloud. We also demonstrate our solution in the context of healthcare.

- In Chapter 5, we present solutions that prevent security attacks carried out by malicious insiders and outsiders, related to data sharing in the Cloud. Our solutions focus on preventing collusion attacks between attacker, revoked user and/or CSP, insider attacks, sniffing attacks, and man-in-the-middle attacks, to name a few.

- In Chapter 6, we present our solution to the data owner access control problem. In our solution, we do not primarily focus on the Cloud but instead, focus on ensuring that the data is used by data consumers in the way that the data owner expects. We present a solution ensuring that data access by data consumers abides by policies set out by the data owner. For example, if a data owner states in the policy that his data should not be copied, a mechanism will prevent the data from being copied by the data consumers via USB transfer or email attachments, etc.

- In Chapter 7, we lay out the directions for future work and conclude the thesis.

Chapter 2

Literature Review

In this chapter, we present a review of the research literature related to secure data sharing in the Cloud. We first present a summary of reviews on the privacy and security of data in the Cloud. We then detail the privacy issues in the Cloud, provide some examples, and list a few requirements for secure data sharing in the Cloud. We then describe, compare and contrast work done on key management in the Cloud and data owner access control. We conclude the review by providing future directions for private and secure data sharing in the Cloud.

2.1 Reviews on Privacy and Security in the Cloud

There have been a number of reviews on security and privacy in the Cloud. Xiao et al. [34] have identified the five concerns of Cloud computing – confidentiality, integrity, availability, accountability and privacy – and have thoroughly reviewed the threats to each of the concerns, aswell as defence strategies. Chen and Zhao [35] outline the requirements for achieving privacy and security in the Cloud and also briefly outline

the requirements for secure data sharing in the Cloud. Zhao [36] provides a survey on privacy and security in the Cloud, focusing on how privacy laws should also take Cloud computing into consideration, and what work can be done to prevent privacy and security breaches of one's personal data in the Cloud. Wang et al. [37] have explored the factors that affect the management of information security in Cloud computing. They explain that the necessary security need for enterprises is to understand the dynamics of information security in the Cloud. Saravanakumar et al. [38] surveys related works on Cloud interoperability, security, privacy and trust. Hosseinzadeh et al. [39] surveyed works that proposed techniques to enhance privacy and security via obfuscation and diversification techniques. Raja et al. [40] analyses works and emphasises the need for privacy preserving identity management in a public Cloud environment. Kumari et al. [41] reviews works on security issues and concerns in the Cloud as well as some countermeasures. Oza et al. [42] have carried out a survey on a number of users to determine the user experience of Cloud computing, and have found that the main issue for all users is trust and how to make a choice between different Cloud Service Providers. Wang [43] carried out a study on the privacy and security compliance of Software as a Service (SaaS) among enterprises.

Table 2.1 shows a summary of the reviews of privacy and security in the Cloud. The table categorises the related work in two aspects; Cloud security and Data sharing. The table depicts whether the related work addresses the threats, provides defense strategies and/or provides a number of requirements in order to achieve Cloud security or data sharing privacy. The table also depicts whether the related work addresses how much of an impact the Cloud and/or data sharing has had on society in general.

	Cloud Security	Data Sharing	Threats	Defense Strategies	Require-ments	Impact on society
Xiao Z. et al. [34]	Y	N	Y	Y	N	Y
Chen and Zhao [35]	Y	Y	Y	N	Y	Y
Zhao [36]	Y	N	Y	Y	Y	Y
Wang et al. [37]	Y	N	Y	Y	Y	Y
Saravanakumar et al. [38]	Y	N	Y	N	N	N
Hosseinzadeh et al. [39]	Y	N	N	Y	N	N
Raja et al. [40]	N	N	Y	Y	N	Y
Jen-Sheng Wang et al. [37]	Y	N	N	Y	Y	Y
Yu-Hui Wang [43]	Y	N	N	Y	Y	N
Oza et al. [42]	Y	N	Y	N	Y	Y
	Y = Yes N = No					

TABLE 2.1: Summary of reviews

2.2 Privacy and Security Issues in the Cloud

Privacy has many definitions in the literature. Some examples of the different definitions of privacy are: "being left alone," "the control we have over information about ourselves" and "the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others" [44], to

name a few. The Organization for Economic Cooperation and Development (OECD) [35] defines privacy as "any information relating to an identified or identifiable individual (data subject)." The American Institute of Certified Public Accountants (AICPA) and the Canadian Institute of Chartered Accountants (CICA) in the Generally Accepted Privacy Principles (GAPP) standard [35], define privacy as, "The rights and obligations of individuals and organizations with respect to the collection, use, retention, and disclosure of personal information." From these definitions, it is clear that a person has some level of control over what they want to disclose about themselves and what they want to keep secret about the rest of their information. Privacy should not be assumed to have the same meaning as confidentiality. Confidentiality involves allowing only authorised users to gain access to that information, and no one else. Confidentiality can be regarded as a subset relation of privacy.

Security, on the other hand, is defined by NIST [45] as "A condition that results from the establishment and maintenance of protective measures that enable an enterprise to perform its mission or critical functions despite risks posed by threats to its use of information systems. Protective measures may involve a combination of deterrence, avoidance, prevention, detection, recovery, and correction that should form part of the enterprise's risk management approach."

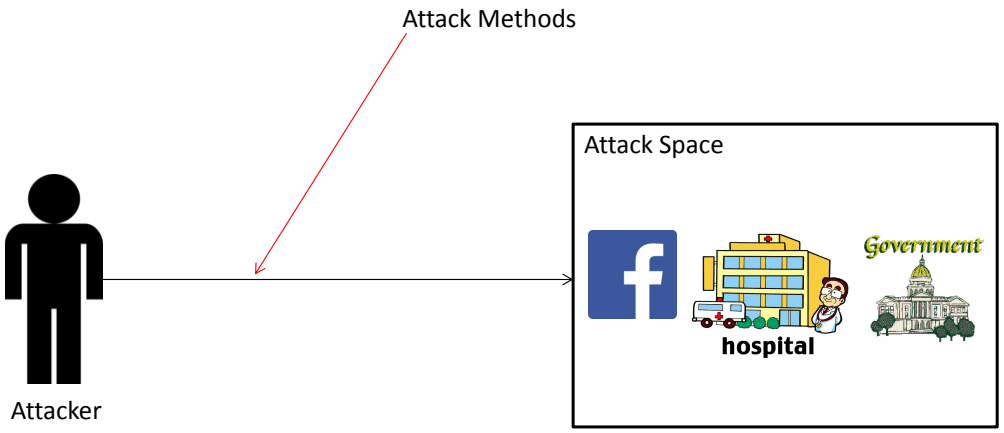


FIGURE 2.1: Attack Model

Figure 2.1 highlights a basic attack model. An attacker will use attack methods to carry out a privacy and security attack on an attack space. We now describe each of these components in detail, as well as provide guidelines and discuss the current approaches used to mitigate and/or prevent these attacks.

2.2.1 Attackers

There are many different types of attackers, with different reasons to attack users [46], [47]. The following are examples of the motives of attackers.

- *To steal valuable data* - Hackers love to steal data, as there is sensitive data stored in the Cloud worth millions of dollars. With access to valuable data, they can then generate revenue; for example, WikiLeaks [48].

- *To cause controversy* - Some attackers purely love the thrill and excitement of causing chaos on the Internet, and the Cloud is one of the best mediums to target, mainly because of the popularity of the Internet and it being easier to steal data over the Internet in comparison to a personal computer system. The attack on Apple iCloud is one example, which revealed a number of private pictures of celebrities that were posted all over the Web [49].

- *To obtain revenge* - Former workers recently stripped of their position at an organisation might express their dissatisfaction by hacking the organisation's network. When an organisation makes use of the Cloud, this becomes all too easy for the former employee, and there have been many cases of this happening. For instance, the case of a former employee who managed to access the Cloud provider's server and delete an entire season of a children's TV show [17].

- *To help* - In contrast, a hacker might try to help an organisation by identifying the security flaws in their system. A hacker might be confident enough to bypass the existing security protocol and implant his or her own mechanisms in order to expose the protocol. For example, in 1988, a first-year graduate student created the devastating Morris worm to demonstrate the inadequacies of the computer network's security measures at the time [50].

- *To prove intellect and gain prestige* - Attackers may also want to show off their skills and gain prestige among their peers through hacking a large organisation with solid security mechanisms. Some hackers make a career out of hacking organisations. For example, the attack on the New York Times website exposed many sensitive records. The attack was partly motivated by the need to prove the hacker's intelligence, as the attacker included his own name in the expertise category [51].

- *Are just curious* - Some hackers are curious to learn something about a company and/or an organisation. These types of hackers do not usually have a malicious intention, as they may not be aware of breaking security rules; however, this does not mean that these hackers are less dangerous. For example, the Morris worm in 1988 was also partly motivated by curiosity [50].

2.2.2 Attack Methods

There are a number of types of privacy and security attacks that are possible in the Cloud. The following is a summary of the common types of attacks that could occur in the Cloud.

- *Insider Attacks* - As discussed in Chapter 1, a malicious insider has full and direct access to the data. They can do anything they wish with the data, including selling them for profit [19].

- *Sniffing Attacks* - Attackers intercept the public communication channels and retrieve sensitive information, such as passwords.

- *Collusion Attacks* - When two or more parties have secret information and agree to reveal this illegally.

- *XML Signature Wrapping Attacks* - Using different kinds of XML signature wrapping attacks, one can completely take over the administrative rights of the Cloud user and create, delete and modify images, as well as create new instances in the victim's Cloud [52].

- *Cross-site scripting attacks* - attackers can inject a piece of code into web applications to bypass access control mechanisms. Researchers found this possible with

Amazon Web Services [52], in November 2011. They were able to gain free access to all customer data, authentication data, tokens, and plaintext passwords.

- *Flooding Attack Problem* - Provided that a malicious user can send requests to the Cloud, he or she can then easily overload the server by creating bogus data requests to the Cloud [53]. The attempt is to increase the workload of the Cloud servers by needlessly consuming lots of resources.

- *Insecure Interface* - A vulnerability in the interface which users use to interact with the Cloud may divulge sensitive information [6].

- *Denial-of-Service Attacks* - Malicious code is injected into the browser to open many windows and as a result, deny legitimate users access to services.

- *Law Enforcement Requests* - When law enforcement demands access to the data of a Cloud Service Provider, the Cloud Service Provider is unlikely to deny such a request. Hence, there is an inherent threat to user privacy and data confidentiality.

- *Data Stealing Problem* - A term used to describe the stealing of a user account and password by any means [53], such as through brute-force attacks or over-the-shoulder techniques. The privacy and confidentiality of a user's data will be severely breached. A common mechanism to prevent such attacks is to include an extra value during the authentication process. This value can be distributed to the correct user via SMS, thus mitigating the likelihood of data confidentiality issues.

- *Over-The-Shoulder Attacks:* In these types of attacks, a user attempts to steal critical information via direct observation, such as looking over one's shoulder or taking a snapshot, or even taking video footage. It is the responsibility of the data consumer to keep the data secure and ensure that no secret information is leaked.

These types of attacks will not be covered within the thesis, as this is outside the scope of our work.

2.2.3 Attack Space

Whilst an attacker may have the motivation and methods to undertake an attack, they usually then seek to target specific contexts such as healthcare, social networks, etc., depending on how sensitive the data is or how easy it is to obtain the data. We look at a few contexts from which an attacker may seek to retrieve sensitive data, below.

Healthcare: In the context of healthcare, patients reveal their health-related information to healthcare professionals in order to diagnose and treat illnesses [54]. In the United States, the Health Insurance Portability and Accountability Act (HIPAA) [55] provides federal protection for an individual's personal health information and provides individuals with rights to their information. The HIPAA Privacy Rule provides protection for a patient's personal health information and guidelines on how external entities such as doctors and nurses can gain access to the patient's data, with the patient's consent. While the HIPAA guidelines are also followed by other countries around the world, it is not followed in Australia. The Commonwealth Privacy Act 1988 aims to protect all personal information including health data in Australia [56]. In NSW, all health information is covered by the Health Records and Information Privacy Act (HRIPA) [56]. As [54] argues, since the patient decides to share their data with one or more healthcare professionals, their data is no longer private, but is confidential. In 2014, Community Health Systems, which was one of the largest hospital operators in the US, notified 4.5 million patients that their personal information was stolen by cybercriminals [57]. Hackers were able to access patient names, addresses, dates of birth, and telephone

numbers. Similarly, another attack on a diagnostic radiology service provider exposed the billing information and medical data of over 300,000 patients [58].

Social Networking: Social networking has changed the lives of today's generation. There are many social networking sites with millions of users communicating with each other. Some examples are Facebook, Twitter, Flickr, YouTube, and the list goes on. Internet privacy has been determined as the "right to be left alone" [59]. The technology that is built to support social networking does not effectively support privacy and may even sell personal information about the individual to third parties, and it is mainly up to the individual to disclose information while maintaining privacy. The individual needs to make sure that they do not unknowingly disclose personal information about themselves. Simply disclosing their age, suburb and nationality is enough for malicious users to identify the person. Facebook has undergone scrutiny in the past for not strengthening its privacy measures on user profiles, as private photos could still be viewed by non-private viewers through a friend-of-a-friend, by simply having a friend comment on the image [60]. Facebook was also the target of phishing attacks in early 2012, which involved attempting to steal user accounts and learn financial information [61]. Once accounts were stolen, the user's profile would be locked out and the profile picture would change. In fact, Facebook has been the target of a number of phishing attacks, such as Ramnit [62], which affected up to 45,000 users. Google revealed in June 2011 that hackers from China stole passwords and attempted to break into email accounts to steal information [63]. More than 100 people were affected, including senior government officials. People began to discuss whether this, and the Sony incident, was the start of the downfall of Cloud computing [64]. Hotmail and Yahoo Mail users were also targeted in phishing attacks [65],[66]. The attacks involved a user either clicking a malicious link in the email or viewing the email itself, which would then run a malicious code and attempt to

compromise the user's account. In April 2011, Sony was involved in a massive security blunder that potentially gave away 100 million credit card numbers. Hackers claimed to have stolen millions of credit card numbers from Sony's PlayStation Network [67].

Government: Nearly all governments collect information about their citizens and residents, such as education, finance, gender, loans, earnings, medical costs, criminal offences, and so on [68]. Governments also release data to the public, for their citizens to view. This might not guarantee the privacy of citizens, as some users might be able to infer information about a particular user through government data. In Australia, the Commonwealth Privacy Act of 1988 aims to protect the privacy of any personal information [56]. This includes the collection, use, storage, disclosure and access to any personal information. The Act permits the handling of health information for health and research purposes. In the United States, the Privacy Act of 1974 aims to protect an individual's privacy [69]. According to the Act, individuals have the right to view the information that the government has about them, to modify or remove incorrect information, and to also sue the government for violations of the Act, including, but not limited to, unauthorised access to personal information. Governments also need to keep data private from other governments [70], as the results can be devastating if information is leaked, as with WikiLeaks [48].

Education: Schools usually collect personal health information on all students. This includes the name, telephone number, address, contact details, financial details, allergies/disabilities and family history, to name a few. It is usually strongly implied that schools keep this information confidential and private [71]. Privacy and security breaches may also affect student grades and even open up avenues for stealing and plagiarism. For example, Google Docs (used by many students for assignments and homework) contained a flaw that inadvertently shared user documents with unauthorised users [18].

Other users could access and edit documents without the permission of the Google Docs owner.

Corporations: Major businesses and organisations also require the privacy and confidentiality of their data. The leakage of sensitive information can result in the loss of revenue for a company, even to the extent of ruining the business. In 2007, Salesforce.com leaked customer contact lists after an employee revealed the list to a phisher, and in turn allowed scammers to target phishing attacks against Salesforce.com customers [19]. A Distributed Denial-of-Service (DDoS) attack on Amazon Web Services in 2009 forced many companies to shut down temporarily, such as Bitbucket [72].

Attacks contribute heavily to user suspicion, which affects their trust in storing sensitive data in the Cloud. In order to gain users' trust in using the Cloud to store critical data, mechanisms need to be implemented that guarantee data is kept both confidential and secure from unauthorised users.

2.2.4 Existing Solutions to Privacy and Security Issues

We now discuss the guidelines for a private and secure Cloud, as well as the current technologies in place to prevent privacy and security attacks on the Cloud.

2.2.4.1 Guidelines

According to [73], the above issues could have the following impacts on the Cloud:

Governance - Organisations usually have standards, practices, protocols, policies and procedures that employees must abide by, which can cover application development, testing, implementation, monitoring, and so on. When an organisation makes use of

Cloud services, there is always the possibility that employees bypass these rules, as there is a lack of organisational rules regarding the Cloud.

Compliance - This refers to an organisation's responsibility to operate in agreement with established laws and regulations. There are a number of privacy and security laws within different countries and states, and when using the Cloud, an organisation has to consider whether they are likely to breach any privacy or security laws, as data stored in the Cloud is usually stored in multiple locations around the world, and at times without the knowledge of the user.

Trust - It is a well-known fact that when a user or organisation chooses to outsource their data to the Cloud, they relinquish full control of their data and provide a high level of trust to the Cloud provider. As discussed in the introduction and the next section, most privacy and security attacks on data arise from insider attacks. The Cloud provider usually has direct access to data and is thus more likely to steal data for illegal purposes. In terms of trust, there is also the issue of data ownership, such as who owns the data, and contracts specifying whether the Cloud has some or no access to parts of its data.

Architecture - The architecture of the Cloud needs to be designed in a way that prevents privacy and security attacks. For instance, IaaS Cloud providers can provide Virtual Machine Images to consumers. An organisation that makes use of these images could store highly critical data. An attacker may examine the images to see whether they leak information. An attacker may also supply a corrupted virtual machine image to users and thus steal confidential data. It is important that the architecture of the Cloud is developed such that it ensures privacy and security, as attackers are always on the lookout for security holes within Cloud architecture.

Identity and Access Management - As data sensitivity and privacy are becoming an ever-increasing issue for organisations, the identity and authorisation framework present in the organisation may not extend to the Cloud, and malicious users may be able to gain unwarranted and unauthorised access to the data.

Software Isolation - With multi-tenant Cloud computing architectures, computations for different consumers are carried out in isolation, even if the software remains in a single software stack. Applications running in the Cloud are susceptible to attack and compromise; therefore, isolation is needed to prevent such attacks.

Data Protection - Data stored in a public cloud usually reside with other data from other organisations. When an organisation places their sensitive data in a public cloud, they must account for the possible privacy and security attacks by ensuring proper access control mechanisms, such as encryption. Since data is stored "in the open," there is a world of opportunity for malicious users to steal data. Similar concerns exist when data is in transit.

Availability - As defined in the NIST Security and Privacy Guidelines [73], availability is the extent to which an organisation's full set of computational resources is accessible and usable. Attacks such as Denial-of-Service attacks, server downtime and natural disasters, affect availability and stored data and, more importantly, cause downtime, which greatly affects an organisation. *Incident Response* - An incident response is an organised method of dealing with the consequences of a security attack. The Cloud contains many layers such as applications, the operating system, network, database and so on, and a log is generated of any event as part of its intrusion detection system. Such a complexity of layers means that it will take many hours to identify an attack in the Cloud.

2.2.4.2 Technologies

In this section, we discuss what is currently being done to protect and/or mitigate privacy and security attacks on the Cloud.

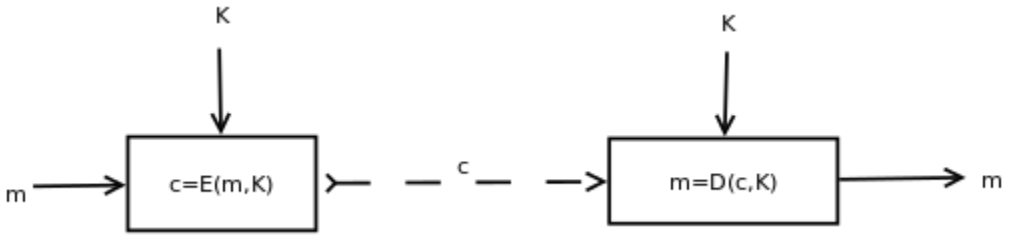
Currently, there is on-going research into how to protect the confidentiality and security of data stored in the Cloud. Jayalatchumy et al. [74] propose the implementation of security as a service in the Cloud and the implementation of an intrusion detection system for the Cloud. They develop a discretion algorithm which processes data to make it anonymous and sends this to various service providers in the Cloud for further processing. Although the data is anonymous, it still presents a considerable risk to store data in plaintext in the Cloud. Cavoukian [26] argues the need for flexible and user-centric identity management so that in the future, a user will not have to re-enter their credentials for a website and can rely on an identity service to manage website access. This of course has its problems, including that there is potentially a single point of failure in the identity service unless it is carefully designed.

The THEWS architecture proposed by Ruotsalainen et al. [75] has developed a privacy management architecture to help the data owner create and manage the network and maintain the privacy of ubiquitous health information. Routsalanien et al. point out that there is an asymmetric relationship between health information systems and their users because users rarely have the power "to force a system to put personal rules into effect." The architecture makes uses of policies and one of the main challenges with this work is how to translate narrative rules defined by the user into machine-readable policies.

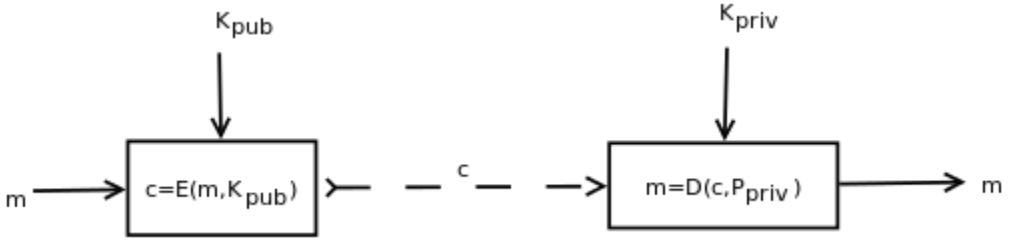
In order to protect a user's data confidentiality, some form of access control needs to be

implemented in the Cloud. Access control should allow a user to choose who can view his data and who cannot. As mentioned in Chapter 1, Access Control Lists (ACLs) [28] were originally used, however, this is ineffective as it is too coarse-grained and unscalable, which is one of the primary features of the Cloud.

An alternative and effective access control technique is encryption. Encrypting data ensures that data is protected from unauthorised users. There are two types of encryption: symmetric and asymmetric. In symmetric encryption, a key is used to encrypt the data to make it virtually unreadable. The same key is also used to convert the unreadable ciphertext into its original plaintext. This key must be kept confidential with the data owner. In asymmetric encryption, public and private keys are used to encrypt and decrypt data. A user encrypts the data using another person's public key. The other person then uses his private key to decrypt the data. The public key can be broadcast to the world but the private key must remain confidential with the user.



(A) Symmetric Encryption



(B) Asymmetric Encryption

FIGURE 2.2: Symmetric and Asymmetric Encryption

Encryption of data held in the cloud is crucial for preserving the security of the data. Much of the literature suggests the need for encrypting data in the Cloud in some form

or another. RuWei et al. [18] state that encryption must occur in transit, at rest, and on backup media. Gentry [76] proposes the use of homomorphic encryption to keep data secure and confidential. With homomorphic encryption, it is possible to perform operations such as querying and searching encrypted data without ever having to decrypt the data, thus exposing privacy. Yao et al. [77] propose a system called "TrustStore," which encrypts and partitions data on the client side and sends each partition to different Cloud storage providers. This greatly enhances the confidentiality of data, as the probability of compromising two or more storage providers is low. However, it does not handle the case of data sharing and collaboration, which is the focus of our thesis. Silva et al. [78] present a data encryption solution for mobile health apps and have conducted a performance evaluation comparing both symmetric and asymmetric encryption algorithms. Even when data is encrypted, it may still be possible for a malicious Cloud provider to deduce information from the encrypted data. Zhang et al. [79] propose a novel solution that adds noise obfuscation based on a time-series pattern, to client data stored in the Cloud. This can help to protect the privacy of the owner's data, since it prevents malicious service providers from deducing information from the encrypted data.

Paul et al. [80] proposes a "Good Enough" method for privacy-preserving Cloud data storage which aims to balance cost efficiency and security. The method utilises hashed data splitting techniques to anonymize and separate uniquely identifiable data and also preserves privacy by taking advantage of multiple Cloud providers. Rahaman et al. [81] proposes the Preserving cloud computing Privacy (PccP) model which aims to preserve the privacy of user information in the Cloud.

Table 2.2 presents works related to privacy and security protection in the Cloud. We categorise and distinguish between papers that focus on privacy, security or both.

Method	Privacy protection	Security protection
Jayalatchumy et al. [74]	N	Y
Cavoukian [26]	N	Y
Ruotsalainen et al. [75]	Y	N
RuWei et al. [18]	Y	Y
Gentry [76]	Y	Y
Yao et al. [77]	Y	Y
Silva et al. [78]	Y	N
Zhang et al. [79]	Y	N
Paul et a. [80]	Y	N
Rahaman et al. [81]	Y	N
	Y = Yes N = No	

TABLE 2.2: Related works on privacy and security protection in the Cloud

When considering data sharing and collaboration, simple encryption techniques do not suffice, especially when considering key management. To enable secure and confidential data sharing and collaboration in the Cloud, there first needs to be proper key management in the Cloud. This will be explained in detail, in the next section.

There are research (and implementation) opportunities covering areas such as optimal key management and distribution amongst a set of identified users. How is revocation implemented and if a key or other credential is revoked, can the user rejoin the group while preserving their rights in the system? We will provide a solution to this in Chapter 4.

2.3 Private and Secure Data Sharing in the Cloud

In this section, we discuss the traditional approach to sharing data via the Cloud and why the traditional approach is not effective. We also discuss the key management problem and review a number of works that address this issue. We then review recent works that aim to provide private and secure data sharing in the Cloud, and discuss the latest techniques used to achieve this.

2.3.1 Traditional Approach

A trivial solution to data sharing and collaboration in the Cloud involves a data owner distributing encryption keys to every user that he authorises. Each user that has authorised access can then obtain the encrypted data from the Cloud and decrypt the data using the supplied key. This ensures that no unauthorised user gains access to data even if the user manages to download the ciphertext from the Cloud, as the user does not possess the decryption key.

This solution, however, is both inefficient and ineffective. Once the data owner decides to revoke a user's access to their data, one trivial solution would be for the data owner to decrypt the data and then re-encrypt the data, this time with a new key, and distribute this new key to the remaining users in the group. This can become extremely costly and places an immense burden on the data owner, when considering group sizes between thousands and millions of users. Furthermore, as members of the group continually join and leave, continually re-encrypting data and sending re-encryption keys to a group of this size, becomes impractical for the data owner and infeasible to implement in the real world. Currently, there is on-going research on this problem.

2.3.1.1 The Need for Key Management in the Cloud

Key management covers the creation/deletion of keys, activation/deactivation of keys, transportation of keys, storage of keys, and so on [82]. It does not cover encryption and decryption. Most Cloud Service Providers provide basic key encryption schemes for protecting data, or they might leave it to the user to encrypt their own data.

Either way, there is a need to encrypt data involved in the Cloud. This raises a few questions, namely:

- How do we handle the keys that are used for encryption?
- Where should the keys be stored and who has access to those keys?
- How do we recover data if keys are lost?

The solution we propose in Chapter 4 will address these questions. Both encryption and key management are very important to help secure applications and data stored in the Cloud [83]. In recent times especially, there has been a strong need for Cloud providers to adopt a robust key management scheme for their services. However, there are still key management issues affecting Cloud computing, as described in [84]. We discuss the three requirements of effective key management, below.

- *Secure key stores:* The key stores themselves must be protected from malicious users. If a malicious user gains access to the keys, they will then be able to access any encrypted data that the key corresponds to. Therefore, the key stores themselves must be protected in storage, in transit, and on backup media.
- *Access to key stores:* Access to the key stores should be limited to the users that have the rights to access the data. The separation of roles should be used to help

control access. The entity that uses a given key should not be the entity that stores the key.

- *Key backup and recoverability:* Keys need secure backup and recovery solutions. The loss of keys, although effective for destroying access to data, can be highly devastating for a business, and Cloud providers need to ensure that keys are not lost, through backup and recovery mechanisms.

Tim Mather [85] states that key management in enterprises today is broken, and that key management in the Cloud is a failed model that is neither effective nor scalable. Fortunately, there are a number of standards for key management in the Cloud, which are briefly described, below.

- *OASIS Key Management Interoperability Protocol (KMIP)* - Used to define a single, comprehensive protocol for communication between encryption systems and enterprise key management systems [86]. KMIP is becoming a widely accepted standard within industries, who are looking to implement it within their frameworks.

- *NIST SP 800-57* - Provides general guidelines on key management, the recommended types of encryption schemes and protection requirements, as well as information on key recovery [87].

- *IEEE 1619.3 Key Management* - covers storage encryption and key management, mainly for IaaS storage [84]. The standard has been dissolved since December 2010.

- *ISO/IEC 11770-5:2011* - Specifies key establishment mechanisms for multiple entities to provide procedures for handling the cryptographic keys used in symmetric and asymmetric encryption algorithms [88].

- Other standards include ISO 11568-2:2012 [89] and IETF KeyProv.

Bruce Schneier [83] states that "Key management is the hardest part of cryptography and often the Achilles' heel of an otherwise secure system." [83] argues that since technology is so broad, as it spans various operating systems, storage, encryption and key management, virtualisation, VM mobility and Cloud, key management solutions in the Cloud need to be broader. Luther [65], on the other hand, states that key management is more difficult than cryptography; where cryptography boils down to mathematics, key management involves technology, people, and processes. He states that strong encryption is nearly impossible to beat compared to key management, which is not as robust.

2.3.1.2 Review of works on Key Management

Zhao et al. [90] proposed a progressive elliptic curve encryption scheme (PECE), whereby a piece of data is encrypted a number of times using multiple keys, and later decrypted using one key. Data sharing involves one user, say, Alice, encrypting her data by using her private key and storing the encrypted data in the Cloud. Another user, say, Bob, sends a request for data access permission by sending his public key to Alice. Alice sends a credential to the storage provider for the re-encryption of data, and sends a credential to Bob to decrypt the data. This is an effective technique, as it keeps data confidential because data is encrypted throughout all of the stages, thereby never allowing a malicious user to view the plaintext data. This technique also does not

allow the permission bearer (in our case, Bob) to share the file owned by the permission holder (Alice) with other users. The main problem with this technique, however, is that it requires the data owner to be online at all times, making it inefficient for everyday users. This technique also assumes that the private key of the Cloud provider is shared with the data owner. Realistically, no system administrator would want to share their keys with users, thus making it impractical to deploy this technique.

Lei et al. [91] have illustrated the need for proper key management in the Cloud environment. A Cloud Key Management Infrastructure (CKMI) is proposed, which contains a Cloud Key Management Client (CKMC) and Cloud Key Management Server (CKMS). The protocol includes: objects that contain keys and certificates, etc.; the operations upon them, such as creation, deletion, retrieval and the updating of keys and certificates; and attributes related to the object in question, such as the object identifier. The method is effective for proper key management; however, if the server is broken, all of the user's data is lost and there is no proper backup and recovery mechanism, which is a key requirement of key management, as described above.

Huangi et al. [18] have built on the Leakage Resilient Authenticated Key Exchange (LR-AKE) first proposed by Fathi et al. [92], and they have proposed the LR-AKE Cluster mode protocol for effective key management. The LR-AKE involves the user remembering a password whilst additionally storing a high-entropy secret on the client machine, to allow communication between different servers. In the LR-AKE Cluster mode, the client generates authentication secrets for each server and partial data keys. Each pair authenticates and communicates with each other to combine partial keys, in order to reveal full data keys when the user requests this. The main weakness with this protocol is that if any one of the client servers fail, the data is lost, as the keys used

to access the data will be unavailable. The LR-AKE Cluster+ mode builds on the LR-AKE Cluster mode, where, aside from the user's personal password, the client chooses a random password (256 bits long), and this random password and the authentication secrets are stored on another device (e.g., a USB drive), for added security and higher availability. Secrets are required from both communication parties and hence the data remains information – theoretically secure and confidential. One of the drawbacks to this approach is that it requires the maintenance of a number of servers and the client, which adds unwanted complexity when trying to attract large number of users to the Cloud.

Sanka et al. [93] have proposed capability lists for effective key management and data access, where the data owner does not have to be online at all times. The model involves using a capability list, where the data owner creates a list containing an entry for each user and the permissions for file access, and stores this list in the CSP. When a user requests access to a file, he makes this request directly to the CSP; therefore, the data owner does not have to be online at all times, and only needs to be online when registering new users or revoking users from the list. The model is confidential and secure against the Cloud and unauthorised users, since they never know the contents of the encrypted data because the key is a shared symmetric key between the data owner and the user. The main issue with the model, however, is that it assumes that the CSP will not alter the capability list. The CSP has access to the unencrypted capability list and can maliciously alter files or prevent users from accessing them.

Bennani et al. [94] propose a model that replicates the database in the cloud n times, where n represents the number of roles. When a role is revoked access rights, the corresponding database is removed. In the worst case, changing the access rights of a role leads to the creation from scratch of a new view, and re-keying the corresponding

database. One of the main problems with this model is that it is unfeasible to implement, since it introduces high redundancy and is thus inefficient.

Boldyreva et al. [95] have suggested Identity-Based Encryption (IBE), where a user's ID is used to generate keys in order to access data. Sahai and Seyalioglu [96] have introduced worry-free encryption, where a user can send files to others without worrying about whether they have the right to access data. The solution uses functional encryption with public keys. With all of these solutions, although the Cloud is assumed to be untrusted, the authorised user is always assumed to conform to operations allowed by the data owner. However, once the data is decrypted, the data owner loses control of their data and the decryptor can do whatever he wishes with it, without being caught. In these solutions, the data consumer is assumed to be fully trusted. Wang et al. [97] presents a key management scheme based on hypergraphs. The solution, however, uses rekeying which can be inefficient in terms of user revocation. Song et al. [98] proposes a hidden mapping hyper-combined public key management scheme. Buchade et al. [99] identifies, compares and applies state-of-the-art key management methods to various Cloud environments.

2.3.1.3 Discussion

Table 2.3 shows a summary of the existing literature based on key management in the Cloud. The works that were reviewed had a strong focus on preventing the need for the data owner to be online at all times. Many of the works that were reviewed also had a strong focus on preventing the Cloud from viewing any of the plaintext at all times. However, in terms of achieving proper key management in the Cloud, some form of redundancy had to be introduced in some of the works. Some of the works proposed

Method	Data/Key Redundancy	Data Owner online at all times	Confidentiality preserved from CSP	Single point of failure
Zhao et al. [90]	Y	Y	Y	Y
Lei et al. [91]	N	N	Y	Y
Huang et al. [18]	Y	N	Y	N
Sanka et al. [93]	N	N	N	N
Bennani et al. [94]	Y	N	Y	N
Boldyreva et al. [95]	N	N	Y	N
Sahai et al. [96]	N	N	Y	N
Wang et al. [97]	Y	N	Y	N
Song et al. [98]	N	N	Y	N
	Y = Yes N = No			

TABLE 2.3: Summary of related key management literature

solutions which had a single point of failure. In other words, if one component of their system failed, the entire system also failed.

Proper key management can lead to more secure and confidential sharing of data in the Cloud. A poor key management system can lead to the complete unreliability of the Cloud, and can also diminish the trust of its consumers. Thus, it is imperative that more research is undertaken on achieving more robust key management for the Cloud, not only to attract more consumers and build trust but also to provide a foundation for secure and private data sharing in the Cloud.

2.3.2 Recent Approaches

In this section, we provide a review of the current literature on enabling secure and confidential data sharing in the Cloud.

2.3.2.1 Attribute-Based Encryption

Attribute-Based Encryption (ABE) is an effective and promising technique that is used to provide fine-grained access control to data in the Cloud. Initially, access to data in the Cloud was provided through Access Control Lists (ACLs); however, this was unscalable and only provided coarse-grained access to data [28]. Attribute-Based Encryption, first proposed by Goyal et al. [22], provides a more scalable and fine-grained access control to data, in comparison to ACLs.

Attribute-Based Encryption is an access control mechanism, where a user or a piece of data has attributes that are associated with it. An access control policy is defined and if the attributes satisfy the access control policy, the user should be able to access the piece of data.

There are two kinds of ABE [28], which are described as follows.

Key-Policy ABE (KP-ABE): The access control policy is stored with the user's private key and the encrypted data additionally stores a number of attributes associated with the data. A user can only decrypt the data if the attributes of the data satisfy the access control policy in the user's key. The access control policy is usually defined as an access tree, with interior nodes representing threshold gates and leaf nodes representing attributes.

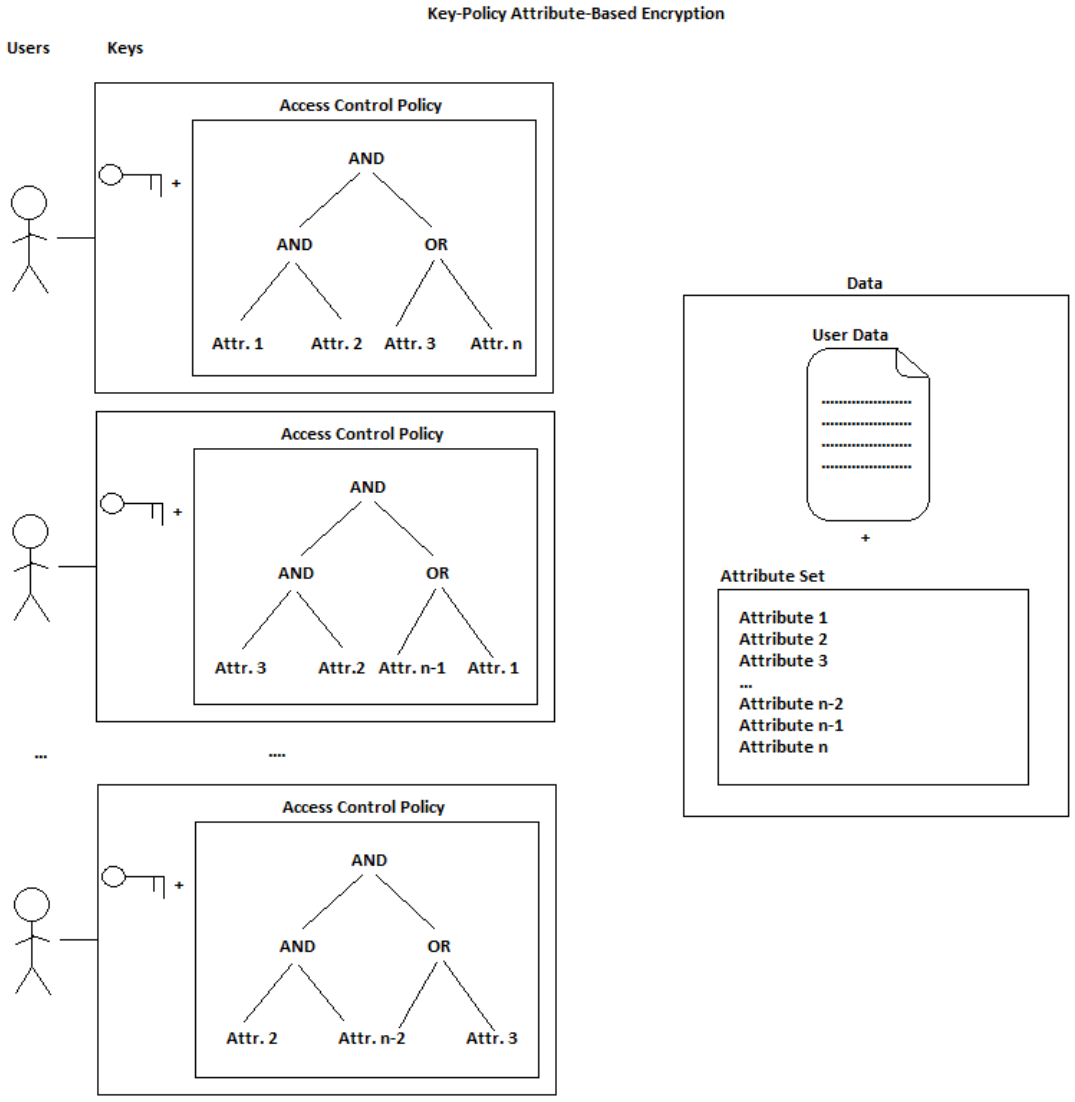


FIGURE 2.3: Key-Policy Attribute-Based Encryption

Ciphertext-Policy ABE (CP-ABE): Essentially the converse of KP-ABE. The access control policy is stored with the data and the attributes are stored in the user's key.

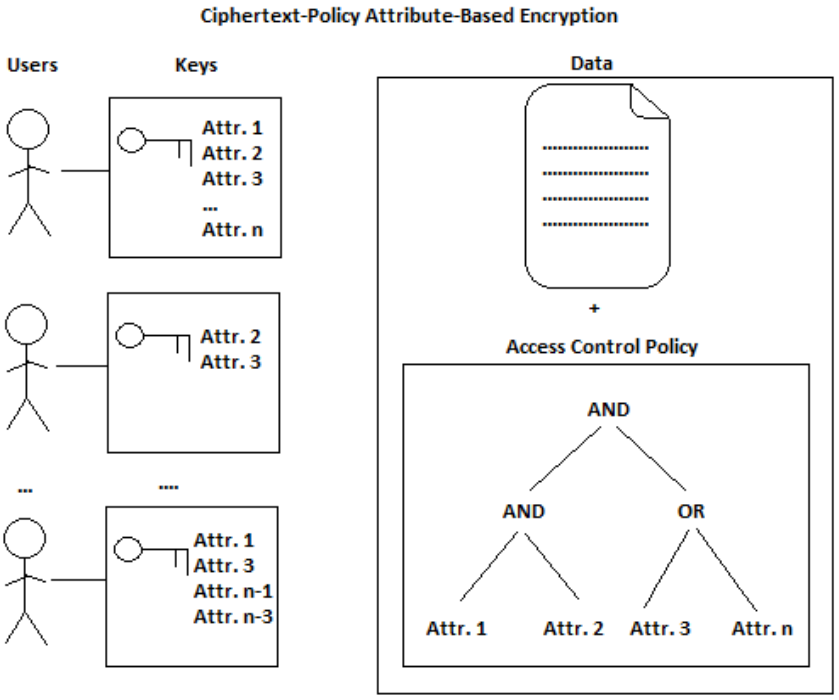


FIGURE 2.4: Ciphertext-Policy Attribute-Based Encryption

ABE is also used for data sharing and collaboration. Tu et al. [24] have made use of CP-ABE in the context of enterprise applications, and have also developed a revocation mechanism that simultaneously allows high adaptability, fine-grained access control, and revocation. The department assigns users a set of attributes within their secret key, and distributes the secret key to the respective users. Any user that satisfies the access control policy defined by the data collaborator, can access the data. When a user's access rights are revoked, the data is re- ncrypted in the Cloud, rendering the revoked user's key useless. The CP-ABE model is proven to be semantically secure against chosen ciphertext attacks. However, the scheme is inelegant in the case of user revocation, since the updating of ciphertexts following user revocation causes a heavy computation overhead, even if the burden is transferred to the Cloud.

Li et al. [100] leverage ABE in the context of sharing personal health records (PHR) in the Cloud. Their framework is comprised of a public domain consisting of users

who access professional records, such as doctors, nurses and medical researchers, and a personal domain consisting of users who are personally associated with the data owner, such as family and close friends. Role attributes that represent their professional role are assigned to the public domain users, and they retrieve their secret keys from an attribute authority. This is effective, as the data owner need not be online at all times. In terms of access control, data owners specify role-based fine-grained access control policies for their PHR files. Using role-based access policies greatly reduces the key management overhead for owners and users, as the owner does not have to manage keys for each user.

2.3.2.2 Proxy Re-encryption

Proxy Re-encryption is another technique that is fast becoming adopted for enabling secure and confidential data sharing and collaboration in the Cloud.

Proxy Re-encryption [23] allows a semi-trusted proxy with a re-encryption key, to translate a ciphertext under the data owner's public key into another ciphertext, which can be decrypted by another user's secret key. At no stage will the proxy be able to access the plaintext. Researchers have utilised Proxy Re-encryption in relation to the Cloud, particularly for secure and confidential data sharing and collaboration in the Cloud.

We demonstrate a basic Proxy Re-encryption scheme with the diagram below. A user, Alice, encrypts her data m, using her public key. When she wants to share the data with another user, Bob, she sends the encrypted data to a proxy. The proxy then converts the data encrypted under Alice's public key into data that is encrypted under Bob's public key, and sends this to Bob. Bob can now use his private key to decrypt the ciphertext and reveal the contents.

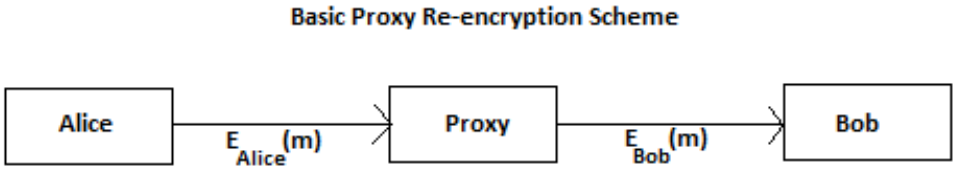


FIGURE 2.5: A Basic Proxy Re-encryption Scheme

A number of researchers have proposed Proxy Re-encryption for enabling secure and confidential data sharing and collaboration in the Cloud.

Tran et al. [25] use the idea of a Proxy Re-encryption scheme where the data owner's private key is divided into two parts. One half is stored on the data owner's machine while the other is stored in the Cloud proxy. The data owner encrypts the data with half of his private key, which is then encrypted again by the proxy using the other half of the key. Another user, who has been granted access rights, will then have the same key divided with different parts. One half will be kept on the granted user's machine, and the other half stored on the Cloud proxy. The user who has access rights can then retrieve the data, as the proxy will decrypt the ciphertext with half of the user's private key in the proxy, and then decrypt this again on the user's side to retrieve the full plaintext. When the data owner wishes to revoke a user's data access, he simply informs the Cloud proxy to remove the user's key piece. The main strength with this scheme is that it does not require re-encryption if a user's rights are revoked and hence saves on computation costs, especially when considering the large number of users in groups. As with the PECE scheme described above [90], this scheme does not allow outsiders to view the original plaintext at any point, as the data remains in an unreadable format in the Cloud. Only users with granted access rights can view the original plaintext. However, the main problem with this scheme is that of collusion attacks: if a revoked user and the proxy collude, that user then has access to the other group users' private

keys. Furthermore, the proxy might suffer from too many encryption and decryption operations. The model also assumes that the data owner has already given permission to a number of users to access the data.

2.3.2.3 Hybrid ABE and PRE

ABE and Proxy Re-encryption have also been used in combination with each other to provide extra security and privacy for data sharing and collaboration in the Cloud. A number of works in the literature are taking advantage of combining the power of the two schemes, to provide more robustness and guarantee further trust on the part of the data owner, with regards to the secure sharing of data in the Cloud.

The work of Yu et al. [101] was one of the first to combine ABE, Proxy Re-encryption and lazy encryption schemes for Cloud privacy and security. The scheme works through the data owner encrypting his data using a symmetric key, and then encrypting the symmetric key using a set of attributes according to the KP-ABE scheme. A new user joins the system when the data owner assigns an access structure and its corresponding secret key, and distributes this to the new user. To revoke a user, the data owner determines the minimum number of attributes, which will never satisfy the revoked user's access structure, and updates these as necessary. All of the remaining users' secret keys will also be updated. Due to the heavy burden on the data owner, which could require him to be online at all times in order to provide key updates, proxy re-encryption is introduced to allow the Cloud to carry out these tasks. Thus, most of the computational overhead is delegated to the Cloud. The data owner's data is kept secure and confidential at all times, as the Cloud is only exposed to the ciphertext and not to the original data contents.

Yanjiang et al. [102] have also proposed a combination of the ABE and Proxy Re-encryption schemes, to enable secure data sharing in the Cloud. The model involves a data owner, Alice, encrypting data d with a random key k. Alice then determines another random value k1 and using access control policy pol, encrypts k1 using ABE. Alice then computes k2 using operations on k and k1, i.e., k2 = k*k1, and encrypts with her public key using Proxy Re-encryption. The two keys (ABE and proxy) and the encrypted data are then stored in the Cloud. Using an authorisation list, if an authorised user exists, he can then obtain the proxy key, which is then re-encrypted with the user's key. Using this, he decrypts the ABE key, then calculates k, i.e., k1*k2, and finally obtains the decrypted file. This technique ensures that data is kept confidential and is protected against any unauthorised users in the Cloud. In the scenario that a user's access rights are revoked, the data owner simply informs the Cloud to remove that user's entry in the authorisation list, which is computationally efficient. This scheme, however, does not deal with the scenario in which a revoked user rejoins the group with different access privileges. The revoked user still has the decryption keys corresponding to ABE and can, in theory, regain unauthorised access to data.

Liu et al. [103] have proposed a clock-based proxy re-encryption scheme (C-PRE) and combined this with CP-ABE to achieve fine-grained access control and scalable user revocation. In C-PRE, the data owner and the Cloud share a secret key and this key is used to calculate the PRE keys, based on the Cloud's internal clock. The Cloud will re-encrypt the ciphertext with the PRE keys. Each user is associated with a set of attributes and an eligible time, which determines how long the user can access the data. The data itself is associated with an access control structure by CP-ABE and also has an access time. When a user requests file access, the Cloud determines the current time using its internal clock and then uses the shared key to calculate PRE keys

in time format, for all of the attributes in the access structure. The PRE keys are then used to re-encrypt the ciphertext. Only users whose attributes satisfy the access control structure and whose eligible time satisfies the access time, can decrypt the data. The main benefit of this technique is that the re-encryption of all of the data is delegated to the Cloud instead of the data owner and is thus efficient, from the data owner's perspective. The user revocation problem is also solved, since the data can only be accessed if the user's attribute satisfies the access control structure, and their eligible time satisfies the access time. One problem with this technique, however, is that data is re-encrypted each time a user makes an access request. Even though the re-encryption is delegated to the Cloud, it is still not a very efficient solution, especially when considering very large data sizes.

2.3.2.4 Self Management and Control Methods

Recent works have also focused on data that can manage and control keys and/or data access, by itself. Chen et al. [33] have proposed bundling data with an access policy, and sending this bundle to authorised users and untrusted applications. The proposed architecture, DataSafe, enables the conversion of policy into hardware tags with parts of data associated with them, such as parts of documents, electronic health records, etc., which helps to provide better access control. However, that data can only be accessed on DataSafe machines with special hardware, limiting the ability of users to gain access anywhere, at any time. Our system can be used on any hardware and on any operating system, requiring only the Java Runtime Environment.

Sundareswaran et al. [104] also bundle the data with an access policy. Additionally, a log file is also bundled with the data. Any operation that the user carries out will be appended to the log file, and this log file will be periodically sent to the Cloud. A data

owner can then access the log files to check whether data is being used appropriately. This prevents man-in-the-middle attacks, as well as attacks related to disassembling the bundled JAR file to read contents, as logs will make notifications of this. To prevent the tampering of log files, a hash function is used to verify integrity. The problem with this is that it does not deny the user the control to carry out illegal operations, such as redistributing copies without permission. Our solution incorporates both log files and data control against illegal operations.

Squicciarini et al. [32] use the idea of self-controlling objects (SCOs) to control how data is used. Data policies, user-created policies and jurisdiction-based policies are encoded in the SCOs, along with the data. The relevant permissions are also created with the SCO. The solution uses CP-ABE [105] for policies; therefore, breaking and reverse-engineering an SCO will still not retrieve plaintext data unless the user is authorised. However, the user can still redistribute to other unauthorised users. The work presented in this paper is based on SCOs; however, we go beyond the current solution and extend SCOs to provide better data access control.

Kayem [106] provides a solution that prevents authorised users from illegal data exchange. The solution uses an invisible digital watermark, which is a hash of the encrypted data and key. However, it does not provide the data owner full control over how data is to be viewed, or how many copies should be made. Burnap et al. [107] propose a solution where parts of the data remain encrypted throughout its lifetime and can only be decrypted if the user has access rights. Kirkpatrick et al. [108] describe a solution that enables data access only to known, trusted devices. A unique device is characterised by Physically Unclonable Functions (PUFs). However, many users never stick to one machine when working, and are likely to use a number of different machines.

One of the distinct features of the Cloud is the ability to access data anywhere, at any time, hence the need to provide more flexibility than this solution.

Zic et al. [109] and Nepal et al. [110] have proposed a technology called the Trust Extension Device (TED), a USB-sized Single Board Computer (SBC), running Linux. The SBC has the Trusted Platform Module (TPM) chip incorporated into it through the use of an additional daughter board. The TED plugs into any untrusted machine via a USB interface and enables secure transactions with an institution. One of the benefits of this solution is that cryptographic keys are encapsulated within the device and are never exposed. All cryptographic operations are done inside the TED via API calls. Since it is a hardware-based TPM, this makes it extremely difficult to retrieve the keys, compared to software implementations that can be bypassed.

2.3.2.5 Discussion

Table 2.4 shows a summary of the existing literature based on secure and confidential data sharing in the Cloud. Many of the works reviewed place a strong focus on preventing collusion attacks, as well as researching ways for the data owner to be online only when required. In terms of user revocation, some of the reviewed literature demonstrates fast methods of user revocation; for instance, where revocation involves simply removing a key. Other works require the data to be re-encrypted and the keys to be re-distributed in a secure method, and this mainly occurs with works that use ABE techniques.

Data sharing and collaboration in the Cloud remain a strong focus of research and in particular, many works are focusing on solving the user revocation problem, as well as ways to manage the sharing and collaboration of large data sizes.

Method	ABE	PRE	SMC	Likelihood of collusion attacks	User revocation
Tu et al. [24]	Y	N	N	N	S
Li et al. [100]	Y	N	N	N	F
Tran et al. [25]	N	Y	N	Y	F
Yu et al. [101]	Y	Y	N	N	S
Yanjiang et al. [102]	Y	Y	N	N	F
Liu et al. [103]	Y	Y	N	N	F
Chen et al. [33]	N	N	Y	N	F
Sundareswaran et al. [104]	N	N	Y	Y	F
Squicciarini et al. [32]	N	N	Y	Y	F
Kayem [106]	N	N	Y	N	F
Burnap et al. [107]	N	N	Y	N	S
Kirkpatrick et al. [108]	N	N	Y	N	F
Zic et al. [109]	N	N	Y	N	F
	Y = Yes N = No F = Fast S = Slow				

TABLE 2.4: Summary of related literature

2.4 Summary

In this chapter, we presented a literature review of the work on enabling secure and confidential data sharing and collaboration using Cloud computing technology. We examined definitions related to Cloud computing and privacy. We then looked at privacy and security issues affecting the Cloud, followed by what is being done to address these

issues.

We then discussed why data sharing in the Cloud is important and the traditional approach to data sharing in the Cloud. We discussed key management in the Cloud and how proper key management leads to more secure and confidential data, which can aid the secure and private sharing of data in the Cloud. We reviewed current, state-of-the-art literature related to key management in the Cloud. We explained the different techniques, namely ABE and PRE, currently used to enable secure data sharing in the Cloud. We also reviewed current, state-of-the-art literature in relation to secure and confidential data sharing in the Cloud and gave a brief overview of the future of data sharing in the Cloud, where the data owner could have more control over the usage of their data.

In the next chapter, we present a formal description of the problem and list the system assumptions and the trust model upon which we will build our solutions in Chapters 4, 5 and 6.

Chapter 3

The Data Sharing Problem and Preliminaries

In this chapter, we define the problem space in terms of sensitive data sharing in the Cloud, and the preliminary concepts needed for our contribution.

3.1 Example of Cloud Data Sharing

To demonstrate the problems associated with data sharing in the Cloud, we showcase a very simple system for private and secure data sharing in the Cloud. We take a very simple scenario: a data owner stores data contents (for example, a document) in Cloud storage (for example, Google Drive) and shares it with data consumers (e.g., workplace colleagues).

Figure 3.1 below illustrates the basic architecture for how data sharing occurs in the Cloud.

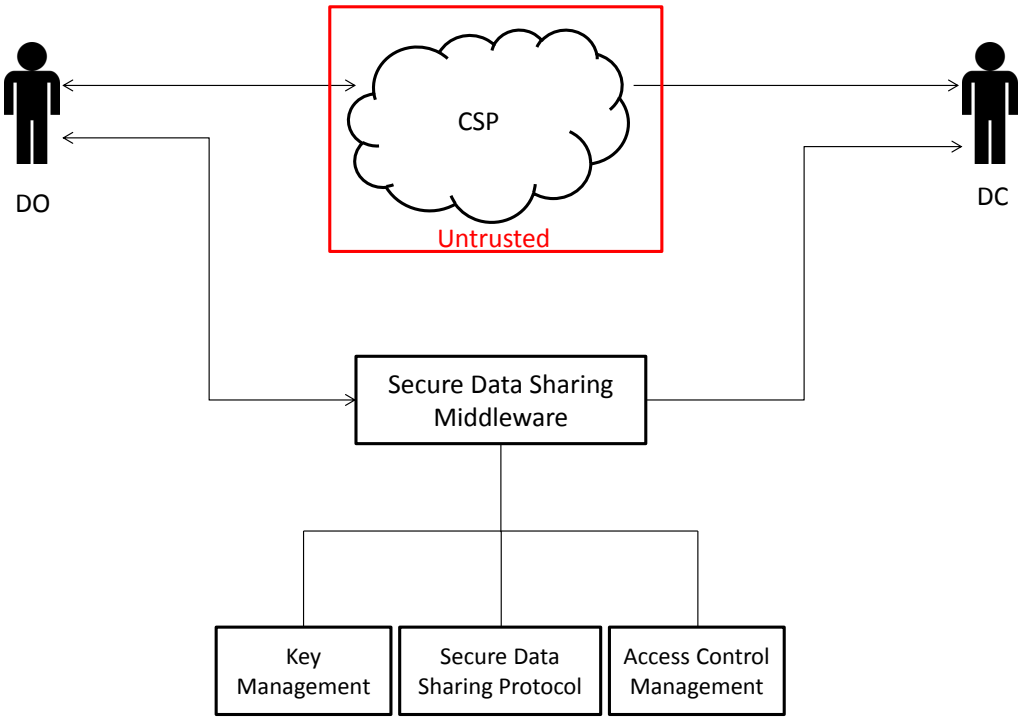


FIGURE 3.1: Basic Architecture for Data Sharing in the Cloud

Throughout our thesis, we develop solutions within the Key Management (Chapter 4), Secure Data Sharing (Chapter 5) and Access Control (Chapter 6) components.

3.1.1 Key Components

- **CSP:** An Untrusted Cloud Storage Provider that is used to store data contents on remote servers.
- **Data-Sharing Middleware:** The systems that are in place to ensure that the data is kept private and secure. The middleware includes three components: Key Management, Secure Data Sharing and Access Control.
- **Key Management:** Responsible for the management of encryption keys.

- **Secure Data Sharing:** Ensures that the data is kept secure at all times, using either software- or hardware-based mechanisms.
- **Access Control:** Ensures that the data is used as per the rules set by the data owner.
- **DO:** The Data Owner responsible for generating and sharing data contents. The DO stores encrypted data contents in the CSP and the encryption keys required to access the data in the Key Management platform. The DO can also decide who can access the data contents.
- **DC:** The authorised Data Consumer who wishes to access the DO's data. The DC downloads the data contents from the CSP and the corresponding encryption keys from Key Management, and decrypts them locally on their machine.

3.2 Key Management

Key management consists of five different operations:

- **Key Generation:** How keys are created and for whom. For example, keys may need to be generated for consumers, trusted key providers, and for various Cloud Service Providers.
- **Key Distribution:** How keys are distributed to the required entities. It is crucial that keys are distributed via secure and private channels, to prevent data leakage.
- **Key Storage:** The location where the keys are stored. Are the keys stored in the Cloud provider's storage, with a trusted key provider, or on the data owner's local machine? It is important that keys do not get lost, stolen or tampered with. It is

common practice to keep keys protected and duplicated across various platforms, to prevent data loss.

- **Key Revocation:** The process of removing a consumer's key or rendering the consumer's key useless, when the data owner wishes to revoke consumer access to their data.

- **Key Update:** How keys are refreshed and updated, to prevent leakage of the key. Updating the key could introduce data re-encryption, which is costly and places a burden on the data owner.

3.2.1 Broadcast Group Key Sharing Example

We present the algorithm of a simple Broadcast Group Key Management (BGKM) [111].

Setup(l): It initialises the BGKM scheme using a security parameter l. It also initialises the set of used secrets S, the secret space SS, and the key space KS.

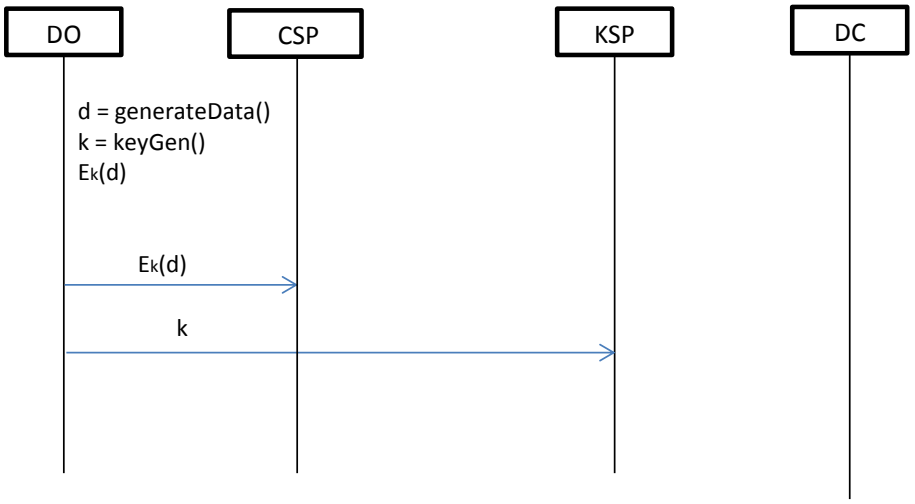
SecGen(): It picks a random bit string s $\in$ S uniformly at random from SS, adds s to S and outputs s.

KeyGen(S): It picks a group key k uniformly at random from KS and outputs the public information tuple PI computed from the secrets in S and the group key k.

KeyDer(s, PI): It takes the user's secret s and the public information PI to output the group key. The derived group key is equal to k if, and only if, s $\in$ S.**Update(S):** Whenever the set S changes, a new group key k' is generated. Depending on the construction, it either executes the KeyGen algorithm again, or incrementally updates the output of the last KeyGen algorithm.

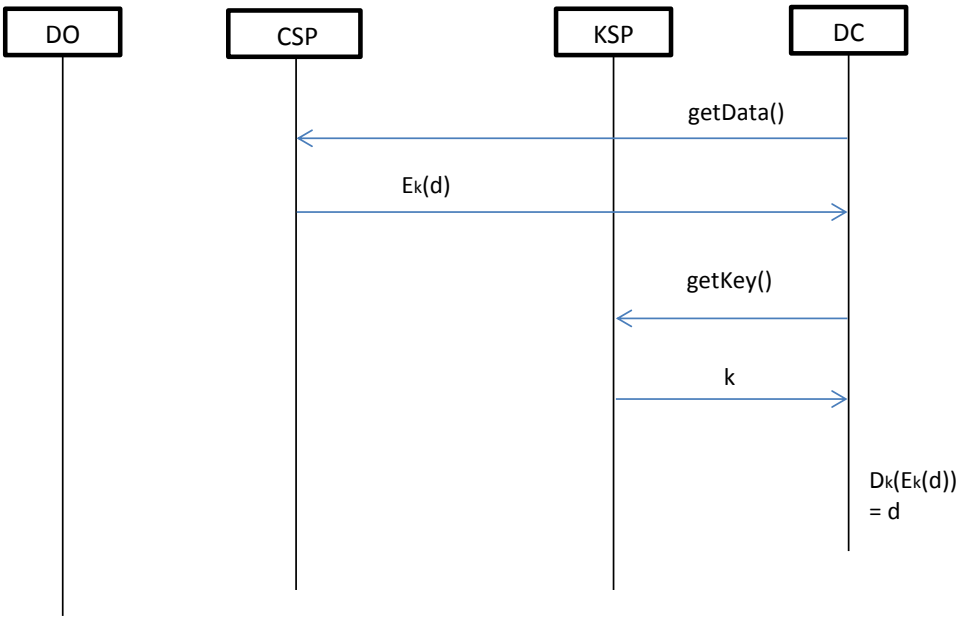
3.3 Secure Data Sharing Protocol

Secure Data Storage



The DO first generates data and encryption keys. To securely store data in the Cloud, the DO must first encrypt the data on their own machine. The encrypted data is sent to the CSP for storage, and the encryption keys are sent to a trusted key service provider. Note that the untrusted CSP has no knowledge of the encryption keys and thus the stored data contents remain illegible.

Consumer Data Access



Assuming that the DC is authorised to view the DO's data, the DC can then download the encrypted data from the Cloud. The DC can also securely retrieve the key from the trusted key provider, via private communication channels (e.g., over the phone, in person, etc.). The DC can then use this key to decrypt the data contents and view the full plaintext.

3.4 Data Owner Access Control

One of the main drawbacks with data sharing in the Cloud is that the data owner loses full control of his data when it leaves the confines of his personal machine. Current approaches [112] to regain some degree of access control include, but are not limited to:

- **Access Control Matrix:** Uses a two-dimensional matrix to represent which subject can access which data, and the operations that are allowed upon them.

	D1	**D2**	**D3**
Bob	R	W	RWD
Alice		RW	R
Dave			W

  From the table above, R denotes Read, W denotes Write, and D denotes Delete.

- **Access Control Lists(ACL):** In an ACL, data objects maintain a list of subjects and the operations they can perform on the data contents. ACLs are widely used as a form of access control.

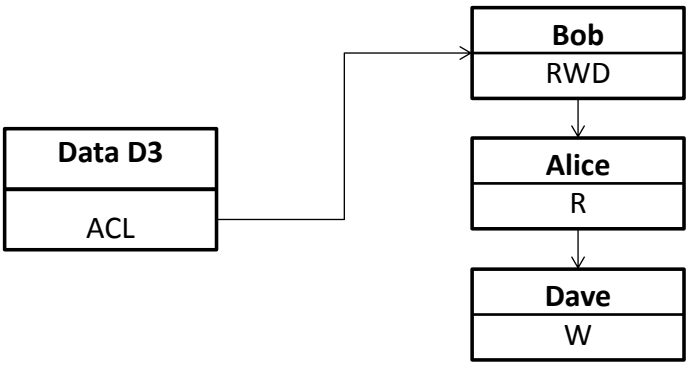


FIGURE 3.2: Access Control List

Figure 3.2 illustrates a simple ACL for data D3. Bob can read, write and delete, Alice can only read, and Dave can only write.

- **Access Control Capability Lists(ACCL):** The reverse of ACL, the subject maintains a list of data objects and the operations that they can perform on each of them.

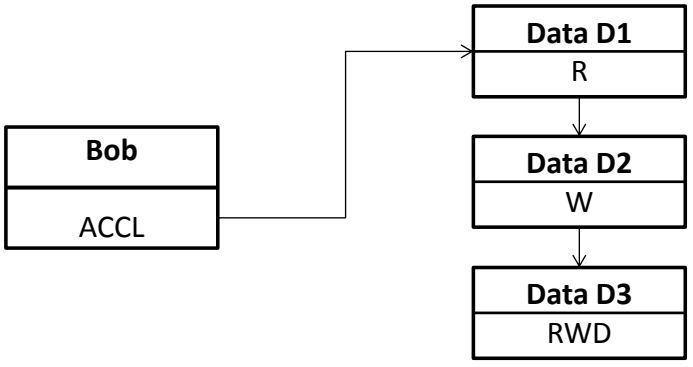


FIGURE 3.3: Access Control Capability List

Figure 3.3 is an example of an ACCL for the subject, Bob, and the operations that he can perform on each of the data contents D1, D2 and D3, respectively.

- **Role-Based Access Control(RBAC):** Similar to ACLs, roles are instead used to associate users with permissions. For example, a user with a "manager" role would have different access rights compared to a user with an "employee" role.

- **Attribute-Based Encryption(ABE):** As described in Chapter 2, ABE provides more fine-grained access control compared to the above. Attributes are stored in the data or private key of the user. An access control policy enforces data access, and uses attributes to determine whether data access should be permitted or denied.

Access Control Matrices, ACLs, and ACCLs are simple to implement and can be used to control how the data is to be used by subjects. However, if the number of subjects is very large, in excess of thousands, it can be cumbersome to manage them. Especially if permissions need to be changed often. RBAC makes this process easier since many subjects can be grouped in roles based on similar attributes such as department. Thus, permissions can be changed for a single role which will in turn control access to a group of subjects. However, some complexities may arise if a subject requires more permissions than what was assigned in their role(s). ABE is also similar to RBAC since it uses attributes, however, it is not as easy to change attributes once the attributes have been assigned to a user through the private key. Since ABE utilises encryption, it is most suitable for data that is sent through the Cloud.

3.5 Secure Data Sharing Challenges

3.5.1 Key Management Issues

As mentioned in Chapter 1, the trivial solution to key sharing places a burden on the data owner, as the data owner is responsible for the re-encryption and redistribution of encryption keys to every member of the group, every time a data consumer is revoked. In Chapter 2, we discussed current solutions that have aimed to mitigate these issues;

however, many solutions still place some burden on the data owner, or the encryption key is somehow revealed to the CSP with the assumption that it is trusted to keep the key secret. The solutions presented also tend to have performance issues, especially due to re-encryption.

- Poor management of encryption keys.
- Little to no focus on private sharing with a large number of users, in excess of thousands to tens of thousands.
- Key information tends to be stored on CSPs that are assumed to be trusted, which is a risky assumption.

In Chapter 4, we present an efficient method of managing keys for data sharing that places little burden on the data owner.

3.5.2 Issues for Secure Data Sharing Protocols

There are a number of different security threats in the Cloud that prevent its widescale adoption for data-sharing purposes. As discussed in Chapter 2, insider attacks continue to represent the biggest threat, as insiders have direct and unrestricted access to the data. Whilst the trivial solution (discussed in Chapter 1) helps to prevent insider attacks, it does not guard against collusion attacks, whereby a consumer might intentionally or unknowingly give away the data owner's sensitive information to the Cloud provider. Also, one of the main requirements of data sharing in the Cloud is that the data must remain encrypted at all times. Any leakage of sensitive data can be devastating. Hackers will always try to exploit any vulnerability that they can find. Thus, one of the biggest

challenges is ensuring that data is always protected, no matter what, and that it should only be accessed by those who have the necessary permissions.

The main issues include:

- Insider Attacks, as insiders have direct access to data owners' data.
- Collusion Attacks between data consumers and Cloud providers.
- Other common security attacks, such as man-in-the-middle attacks, sniffing attacks, etc.

In Chapter 5, we develop secure protocols and systems to allow for the more secure sharing of data in the Cloud.

3.5.3 Access Control Challenges

While access control matrices, ACLs, ACCLs, RBAC and ABE provide solid control over who can access the data and the types of operations that can be performed, they still only provide the data owner with limited access control. Especially with the rapid growth of data sharing in recent times, data owners have a stronger demand for greater access control over their data, when sharing via the Cloud. For instance, malicious insiders have full access to the data owner's data. Also, authorised data consumers who have read/write capability can then make another copy of the data and send it to their friends, via email attachments or USB transfer, with little to no knowledge of the data owner. Once the data contents reside on the consumer's personal machine, they can do anything they want with it. In current research, very little work has been done to fulfil the data owner's need to have stronger control over their own data, when sharing it in the Cloud.

Thus, the main issues are:

- Not fine-grained enough.
- Very little research on preventing unauthorised data sharing by authorised data consumers via email attachments, USB transfer, etc.
- The data consumer can do anything with the data once in possession of it, and it is difficult to hold them accountable.

In Chapter 6, we propose a solution that attempts to address these challenges.

3.6 Problem Statement

The problem statement that will be addressed within the thesis is as follows:

To prevent the leakage of the data owner's protected data to unauthorised entities, after access is given to one or more authorised data consumers (or recipients). The data owner should be able to provide access to a large amount of recipients while also being able to efficiently and effectively revoke recipients from data access, at any time. The protected data should also be accompanied by a policy that states who, what, where, when and how the data is to be accessed. The policy should be enforced during the lifetime of the protected data.

In other words, a data owner should be able to share his or her data with millions of users whilst ensuring that it is protected from unauthorised access and usage. The data should remain encrypted from any unauthorised user or Cloud insider. The data owner should also be able to revoke a user's access on-the-fly, without having to re-encrypt and re-distribute keys. Thus, key management needs to be efficient.

In addition, the data owner should be able to specify a policy that enforces how the data is to be used by the authorised data consumer. The data owner can use the policy to enforce a wide variety of complex conditions. For example, a policy could state that "the data can only be accessed by students for five days" or "the data can only be read, but cannot be copied, modified or printed." This gives the data owner greater access control over their data.

We now provide a formal description of the above problem statement.

We consider a data owner $do \in O$ where O represents the set of all possible users. The do creates data $d \in D$ where D represents all possible data. The data d can be distributed and accessed by data consumers $dc \in C$ where C is a subset of O chosen by the do. We now provide the following restrictions:

1. The data d cannot be accessed by any entity outside of C.
2. The data d cannot be accessed by dc, without the permission of data owner do.
3. The data owner do can give permission to one or more data consumers $dc \in C$ to access the data d.
4. The data owner do can revoke the access of data consumer dc to the data d.
5. If a policy $p \in P$ is associated with the data d and the data owner do gives permission to the data consumer dc, d can be accessed by dc as long as p is obeyed.

3.7 Assumptions and Threat Model

We now describe the system assumptions and the threat model, which will be used throughout Chapters 4, 5 and 6.

3.7.1 System Assumptions

Throughout this thesis, we make the following assumptions:

- We assume data owners and data consumers to have some form of computing device and an Internet connection in order to create, share and access data objects.
- We do not focus on shoulder-surfing types of attacks, such as stealing data via peaking over-the-shoulder, taking a snapshot or video footage, etc. Such types of attacks are outside the scope of this thesis.
- We assume that the data consumer is legitimate and does not hand over their credentials/authentication mechanism to an unauthorised data consumer or entity.
- We assume the Cloud Service Provider (CSP) to be honest-but-curious, in the sense that the CSP will carry out the steps of the protocol as expected, but is always curious and will look for ways to find out any information about the data consumer.
- We assume that our developed mobile apps in Chapter 4 will keep any sensitive information inserted by the user secure and private, and will not inadvertently send information to the CSP or any other adversary, without the prior knowledge of the mobile user.

- In Chapter 4, we assume the data consumer to be trusted. In Chapters 5 and 6, we assume the data consumer to be semi-trusted. In other words, the data consumer will access and use the data as expected, but may either intentionally or accidentally leak data to other entities. For example, a data consumer might send the data owner's data to a friend via email attachments or USB, or might be forced to hand over the data via law enforcement.

3.7.2 Trust/Threat Model

We now describe the trust model for our thesis. The main goal of an adversary is to leak the data owner's sensitive data. An authorised data consumer also poses a threat and may leak the data owner's sensitive data; therefore, the authorised data consumer is an adversary. Adversaries may take advantage of public communication channels to retrieve sensitive data. In this thesis, we do not consider the following threats:

- Vulnerabilities through the data consumer's operating system, which may leak sensitive data.
- Denial-of-Service attacks.
- Loss of data availability in the Cloud attacks.
- Malicious software, such as viruses.
- Hardware-based attacks.
- Memory-based attacks.

3.8 Preliminaries

We will be using the following, preliminary technology terminology in the rest of this thesis.

3.8.1 Symmetric Encryption vs Asymmetric Encryption

We use a symmetric encryption cryptography algorithm in our work, to protect the data owner's data from being accessed by untrusted Cloud servers. Note that in our work, we do not specify which symmetric algorithm is used. In theory, any symmetric encryption algorithm can be used, based on the level of sensitivity of the health data. Throughout this thesis, the encryption algorithms that we make use of include ElGamal encryption and CP-ABE (described in sections 3.7.2 and 3.7.3, respectively).

While asymmetric encryption may be more secure compared to symmetric encryption, due to the greater degree of difficulty in guessing the key, symmetric encryption is far better suited to data sharing in the Cloud. This is due to symmetric encryption using the one key to encrypt and decrypt data, whereas asymmetric encryption uses two keys. When sharing data with very large groups, sometimes in excess of thousands, key management is more efficient when there is only key to protect. In our work, we make use of both symmetric and asymmetric encryption. The symmetric key is used to protect the data and the ElGamal-based asymmetric key is used to protect the symmetric key, which also has the added benefit of bundling the user's identity with the data.

One of the drawbacks of using symmetric encryption for data sharing is that the data owner's identity is not bundled with the data. This makes it difficult for the data owner to claim ownership of the data. Since our solution also uses asymmetric encryption

via the ElGamal algorithm, the data owner's identity is also bundled along with the symmetric key.

3.8.2 ElGamal Encryption

We take advantage of ElGamal encryption in Chapters 4 and 5, since the algorithm is both simple and efficient and can provide simple consumer revocation with a low cost and overhead. ElGamal encryption, invented by Taher Elgamal [68], is a public-key cryptography system. One of the drawbacks of ElGamal encryption is that it is very computationally inefficient and time-consuming to decrypt fairly large data. Thus, the algorithm is best suited to the encryption and decryption of small data. In this thesis, we mainly use ElGamal encryption to add a further layer of protection, by encrypting/decrypting another encryption key instead of the data itself.

There are three main steps to the ElGamal encryption algorithm:

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

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

$$\begin{aligned} E(m) &= m \cdot b^r\ mod\ p \\ &= m \cdot c^{\mathrm{rx}}\ mod\ p \end{aligned} \tag{3.1}$$

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

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

$$\begin{aligned} D_x(E(m)) &= g^{-x} \cdot E(m) \ mod \ p \\ &= (c^r)^{-x} \cdot m \cdot c^{rx} \ mod \ p \\ &= c^{-rx} \cdot m \cdot c^{rx} \ mod \ p \\ &= m \ mod \ p \end{aligned} \tag{3.2}$$

3.8.3 CP-ABE

We make use of the CP-ABE algorithm in Chapter 5 to protect the data contents bundled inside the SafeShare object. As discussed in the literature review in Chapter 2, ABE is a promising technique for private and secure data sharing in the Cloud. Ciphertext-Policy Attribute Based Encryption (CP-ABE) [105] involves encrypting data with an access control policy. A user can decrypt data if, and only if, the attributes included in his private key satisfy the access control policy. The scheme consists of the following four algorithms:

- **Setup**: Using a security parameter L as an input, the Setup algorithm outputs the public parameters PK and a master key MK. PK will be used for the encryption of data and MK for the generation of user attribute private keys.
- **KeyGen**: Takes as input the set of User Attributes UA, the Master Key MK, and outputs user attribute private key UK.
- **Encryption**: Takes as input the data D, an access control policy ACP, and public parameters PK, and outputs the ciphertext E.

- **Decryption**: Takes as input the ciphertext E, user attribute private key UK. If the set of attributes in the key UA satisfies ACP embedded in E, it returns D

3.8.4 XML

Extensible Markup Language (XML) [113] is an open-source markup language that is used to define rules for encoding documents that can be both human-readable and machine readable. XML documents are generic, flexible and structured. It is mostly used to share data around the world. Unlike HTML, XML does not specify fixed semantics or tag sets. The semantics can be defined by users in a way that would be convenient for sharing data. While we don't directly make use of XML in this thesis, we leverage XML via SOAP (described in 3.8.5) and also XACML (described in Chapter 6).

3.8.5 SOA

Service Oriented Architecture (SOA) is an architectural style whose main aim is to loosely couple services [114]. A service is defined as an atomic operation encapsulating a particular business process [115]. The services can then be published and discovered through a service registry (e.g, UDDI) by registering a WSDL (Web Service Definition Language) file within that platform [115]. The WSDL is an XML-based file which contains a description of the web service interface. SOA aims to make resources available to participants in a network as independent services that are accessed in a standardised way [116].

A web service is designed to support interoperable machine-to-machine software interaction over a network [117]. Web services are used to implement SOAs. One of the core web service strategies is the Simple Object Access Protocol (SOAP). SOAP is the

protocol specification for message exchange among web services and is based on XML [118]. XML messages are sent to the called web services via different transport protocols such as HTTP, FTP, etc. A SOAP message tends to be rather large due to the extensibility nature of XML. In our work, we make use of SOAP to demonstrate the secure and private sending of data via web services.

3.8.6 SCO

We leverage the idea of Self-Controlling Objects (SCOs) in Chapter 5, which was introduced by Squicciarini et al. [32]. SCOs provide an effective way to protect data from being illegally redistributed. Data contents and access policies are encapsulated and bundled in these objects. The objects can then control who can access data and under what conditions it can be accessed. To manage updates, such as data modification, SCO Networks (SCONs) are used. They communicate with all of the identical copies around the world to ensure that data is kept up-to-date for collaborating users. We leverage the concept of SCOs in our work, to provide stronger yet flexible security protection that allows data owners to share data with many users and prevents the leakage of data by dishonest users.

3.8.7 TED

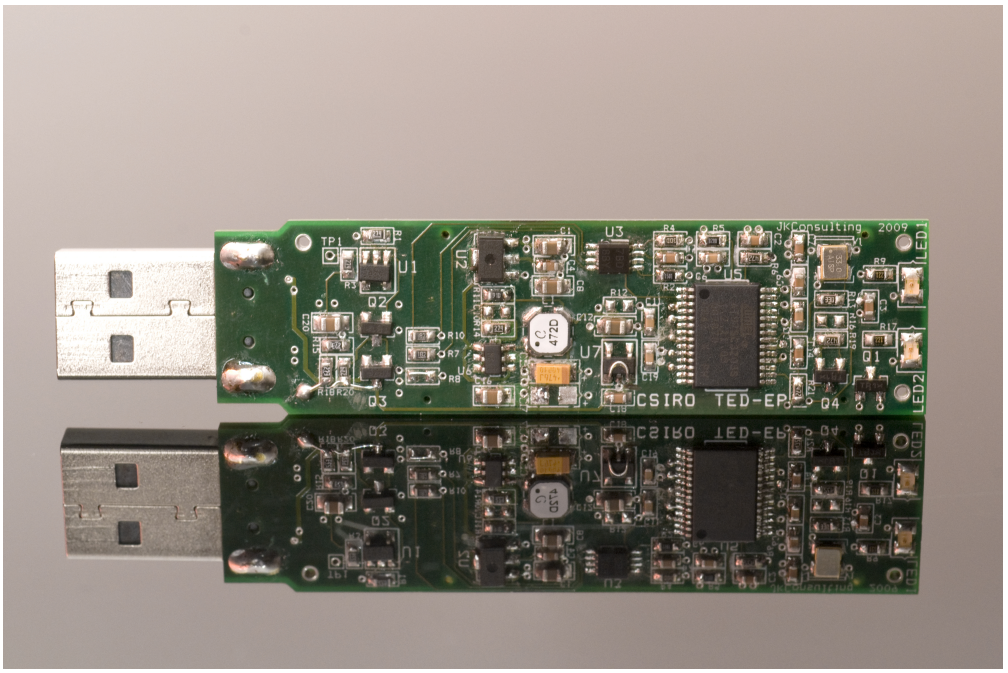


FIGURE 3.4: TED

In Chapter 5, we extend our work on our software-based SafeShare object by using a hardware device to provide stronger security and privacy of data contents. Zic et al. [109] and Nepal et al. [110] have proposed the use of a small, self-contained USB-sized computer that incorporated a hardware-based TPM. This device was called the Trust Extension Device, or TED. This enables secure transactions to occur with an institution using an untrusted machine. TED was not originally designed for Cloud-based applications. We leverage the use of TED in our work to demonstrate secure data sharing and access in the Cloud environment, since it can help to prevent dishonest, authorised consumers from illegally redistributing sensitive data to other consumers who do not have the relevant permissions. Also, since TED is a hardware-based security mechanism, it will provide a much stronger protection compared to software implementations, which

can be easily bypassed. There are three components to the TED enterprise architecture [109]:

- The TED Issuer and Manager that is responsible for generating digital keys, issuing and revoking the TED, and possibly being responsible for the device's manufacture.
- A Privacy Certifying Authority that is responsible for verifying that the TED is valid and authentic.
- An Application Server, deployed within the enterprise, to perform the basic transactions required from the customer.

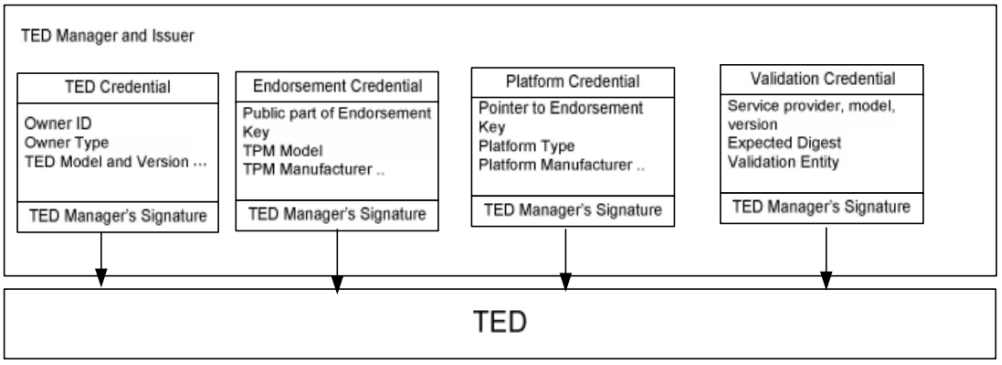


FIGURE 3.5: TED's Credentials managed by TED issuer and manager

3.8.7.1 TED issuer and manager

Thus, for a single TED, the TED manager will generate the following data:

- The TED Credential containing data that identifies the person/client to whom the TED is issued by the enterprise. The details of the client are signed by the enterprise (TED manager).

- The Endorsement Credential includes the public part of the endorsement key that is unique to each TED. The TPM manufacturer signs the endorsement key. This is done by the TED manager in our enterprise architecture.

- The Platform Credential includes the TED's operating environment consisting of VM software and VM OS. In our architecture, the TED manager signs the details of the platform. It is possible to have an independent third party supplying the platform description.

- The Validation Credential includes service component descriptions consisting of their digests, which are loaded into the TED. One could have an independent validation manager. In our simple enterprise architecture, this is also achieved by the TED manager.

Since the TED contains its own Linux-based OS and software, it can also run in-built applications developed during the manufacturing of the TED device. In our work, we use the TED to deploy our own application, which will carry out cryptographic operations of symmetric keys, as explained in our protocol in the next section. The cryptographic operations will be carried out without the knowledge of the host OS. We now discuss the implementation details of TED.

3.8.7.2 Privacy Certifying Authority

The TCG uses a trusted third party, the privacy certification authority (Privacy CA), to verify and authenticate the TPM. The same concept is used in the TED. Each TED is issued with the credentials, including an RSA key pair called the Endorsement Key (EK). The Privacy CA is assumed to know the credential details, along with the public parts of the Endorsement Keys of all TEDs. That is, the TED manager supplies the credential

details to the Privacy CA.Whenever a TED needs to communicate with the enterprise, it generates a second RSA key pair, called an Attestation Identity Key (AIK), and sends an identity key certification request to the Privacy CA, which contains: (a) an identity public key; (b) proof of possession of identity for the private key; and (c) the endorsement certificate containing the TED's endorsement public key. The privacy CA checks whether a TED issuer has signed the endorsement certificate. If the check is successful, the privacy CA returns an identity certificate that is encrypted with the TED's endorsement public key. The TED can then provide this certificate to the application server to verify and authenticate itself, with respect to the AIK. If the TED is reported as stolen or lost, the Privacy CA can compute the corresponding public key and tag it as a rogue TED.

3.9 Summary

We first defined the problem statement of this thesis and then derived the formal description of the problem. We then detailed our system assumptions and also our trust and threat model, which will be used to build upon in the remaining chapters of our thesis. In the next chapter, we present our solution to the key management problem. This allows the data owner to share data with many data consumer's while being able to efficiently revoke consumers on-the-fly.

Chapter 4

An Efficient Solution to the Key Management Problem

4.1 Introduction

In this chapter, we describe and detail our solution to the key-management problem. We develop a solution that will enable private and secure data sharing and collaboration in the Cloud, which will allow for efficient data consumer revocation. In other words, we develop a key-partitioning solution where the data owner can revoke a data consumer without having to re-encrypt the data and re-distribute the new encryption key to the remaining users. Thus, the burden on the data owner is significantly reduced. We demonstrate our ideas in the health domain, using three different application scenarios.

4.2 Our Approach

The main idea is that the data is encrypted using any AES symmetric encryption algorithm. The encrypted data is then stored to the Cloud. The symmetric key used to encrypt the data is then encrypted using the ElGamal public key. Thus, the only way to decrypt the symmetric key is by using the ElGamal private key.

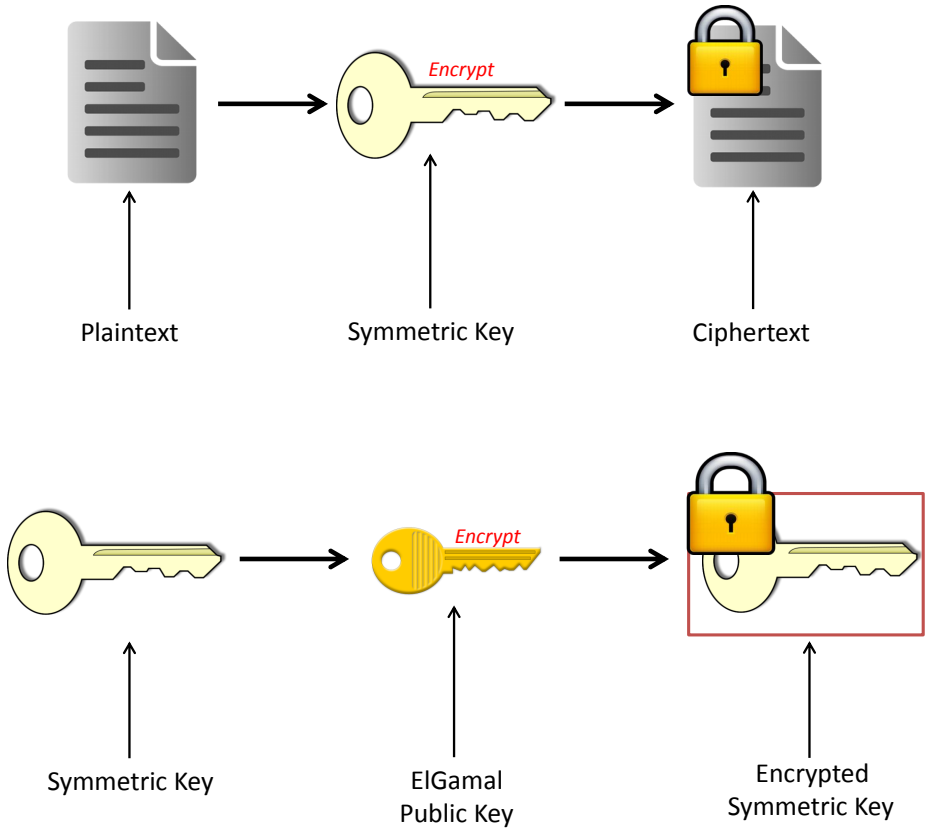


FIGURE 4.1: Encrypted data and key

Figure 4.1 illustrates data and key encryption. The data is first encrypted with a symmetric key and that symmetric key is then encrypted using the ElGamal public key of the data owner. That is,

$$\begin{aligned} E_k(d) &= C \\ G_{pub}(k) &= K \end{aligned} \tag{4.1}$$

where k is the symmetric key, E is the symmetric encryption operation, C is the ciphertext, G is the ElGamal encryption operation, *pub* is the ElGamal public key of the data owner and K is the encrypted symmetric key.

Since the ElGamal algorithm represents its public and private keys as large numbers, this makes key partitioning possible and hence, partial decryption is also possible.Therefore, if we were to partition the ElGamal private key C into two parts A and B such that A + B = C, the symmetric key could be partially decrypted using A and the partially decrypted key can then be fully decrypted using B. Our combined symmetric and asymmetric encryption scheme is highlighted in the diagram below:

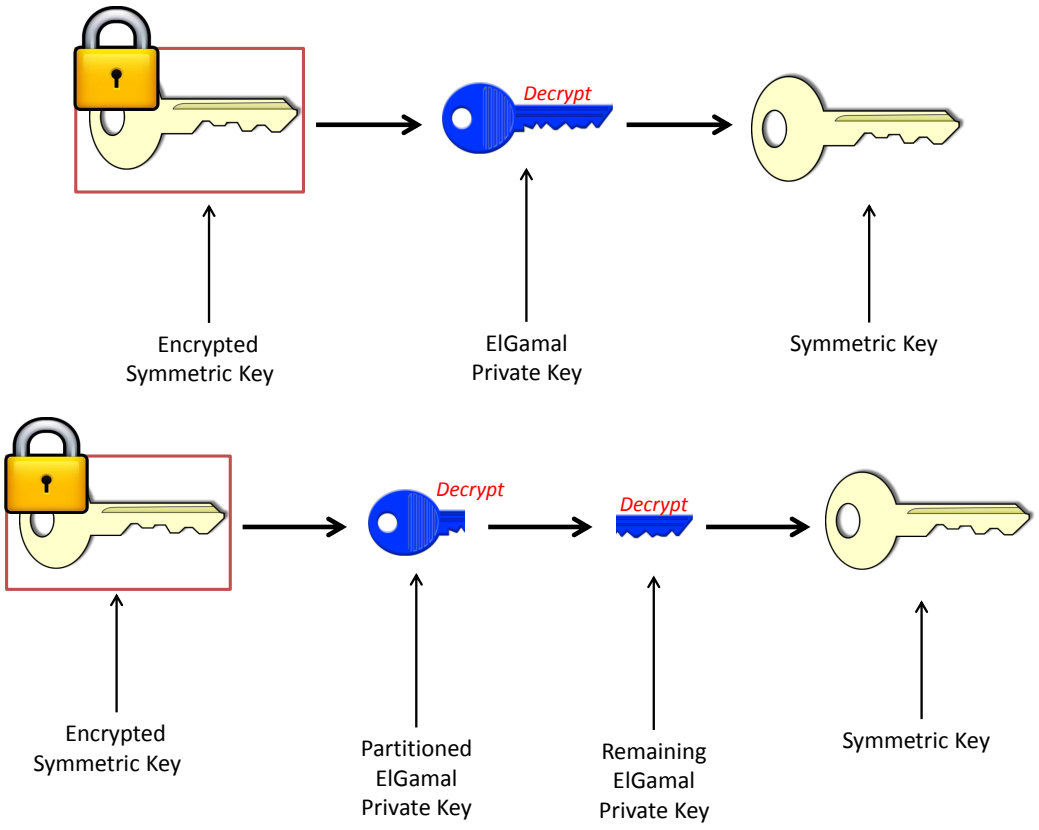


FIGURE 4.2: Key Partitioning

Figure 4.2 illustrates the key-partitioning technique when decrypting the data. The first example shows a standard ElGamal decryption without key partitioning. The second example shows the same result with key partitioning. We provide a more formal description, below:

$$H_{priv}(K) = k \tag{4.2}$$

$$\begin{aligned} priv &= priv1 + priv2 \\ H_{priv1}(K) &= Z \\ H_{priv2}(Z) &= k \end{aligned} \tag{4.3}$$

where $priv$ is the ElGamal private key of the data owner, $priv1$ and $priv2$ are partitions of $priv$, H is the ElGamal decryption operation, and Z is the partially decrypted key.

The above key partitioning is equivalent to $H_{priv2}(H_{priv1}(K)) = k$. We have separated it out to show that the partially decrypted key Z can still be fully decrypted later, with the remaining key partition $priv2$.

Thus, in our data-sharing example, if the Cloud Service Provider (CSP) obtains key partition A and the data consumer obtains key partition B, secure data sharing can occur as follows. The CSP first decrypts the encrypted symmetric key using A, revealing a partially decrypted symmetric key. At this point, the CSP does not know the full plaintext of the symmetric key; he still sees illegible ciphertext. It is also not possible for the CSP to know the value of B without knowing the value of C. The CSP would have to make a guess for B in order to decrypt the symmetric key and since B is likely to be represented using a large number, the guessing is not as easy but is still possible. In our work, we also prevent the CSP from knowing the value of A. We send the partially decrypted symmetric key to the Cloud for storage, instead of just the encrypted symmetric key.

The CSP can then send the partially decrypted symmetric key to the data consumer and the consumer can decrypt the symmetric key using B. Note also that the consumer does not know or cannot incur the value of A when he receives the partially decrypted symmetric key, and only sees a jumble of ciphertext. Thus, a data owner can easily revoke a data consumer simply by requesting the CSP to delete A. This ensures greater efficiency in consumer revocation, as the owner does not have to re-encrypt the data and re-distribute keys.

4.3 Remote diagnosis of patients with cardiac arrhythmias

We now showcase our key-partitioning algorithm in a scenario where a patient wishes to securely and privately send cardiac-related health data to doctors.

4.3.1 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 [119] and the Internet, and the fact that mobile phones are now a way of life amongst people of all ages due to their 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 [120]. 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 [121]. Consequently, the demand for cutting the cost of healthcare has increased, and there is now a growing need for the remote care of patients at home, particularly 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 [122],[123],[124].

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 for the patient, as the health professional obtains a more complete medical history. This is mainly due to not having to repeat the 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 [125]. 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 the secure sharing of telecare data in the Cloud. The Cloud, as an enabler for mobile telecare, can help to provide the effective treatment and care of patients due to its benefits such as on-demand access anywhere and at any time, 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 [17], as they have direct access to stored data.

As discussed in Chapter 2, 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 [73]. 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; for example, the trivial solution to data sharing as discussed in Chapter 1.

We present 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, since we leverage and build on this protocol in Chapter 5. Our protocol will also be efficient, and handles the problem of user revocation and large data sizes. We demonstrate the efficiency through performance tests and we also run a few experiments on our prototype and evaluate it for feasibility of use in the real world. To the best of our knowledge, no other work has focused on the private and secure sharing of health data via the Cloud. Many other works in mobile eHealth focus on securing communication over the Internet, and do not consider data sharing [126], [127].

4.3.2 Related Work

Recently, there have been works that have focused on the integration of mobile and Cloud technologies for health monitoring. Fortino et al. [128] introduced the BodyCloud architecture, which enables the management and monitoring of body sensor data via the Cloud. It provides the functionality to receive and manage sensor data in a seamless way from a body sensor network (BSN). BodyCloud also comprises a scalable framework that allows for the support of the multiple data streams required for running concurrent applications.

Bellifemine et al. [129] and Fortino et al. [130] have presented the SPINE framework. This open-source framework allows developers to rapidly prototype and manage BSN applications. There are two main components of the SPINE framework: the coordinator

side, which is implemented on a PC or smartphone, and the BSN node side. On the coordinator side, SPINE provides application developers with an intuitive interface to the BSN, while on the node side, SPINE provides developers with 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 of Pandey et al. [131] integrates mobile and Cloud technologies with electrocardiogram (ECG) sensors, to enable the remote monitoring of patients with heart-related problems such as cardiac arrhythmias. The patient connects the sensors to their body and then runs 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 status. The scheme also implements middleware in the Cloud. There are web services for users to analyse their ECG and 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 this scheme, to facilitate a secure health-monitoring application that enables the user to share health data in a secure manner with other doctors and nurses, and which will be able to efficiently revoke users, without placing too great a burden on the user. Our system is also generic, in that it is not limited to ECG monitoring and can also monitor other health data, such as pulse and temperature.

AliveCor is a remote app-based ECG monitoring system [132]. 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; therefore, it is possible that an intruder will be able to steal health data with a certain amount of effort.

CardioComm Solutions have also demonstrated their remote patient ECG monitoring service, HeartCheck Smart Monitoring [133]. The system allows for rapid access and for physicians to better review the ECG data in order to assess how the patient should be treated. However, it does not specifically focus on privacy and security aspects, such as the confidentiality of data as it is being transmitted to the Cloud or as it is stored within the Cloud.

Gradl et al. [134] have also developed an Android-based application that allows for the 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. [135] address the usefulness of ECG data collected from patients themselves using mobile devices, and the issues that this presents. They do 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 leverage the mobile health idea and additionally incorporate privacy and security mechanisms, to guarantee patient confidentiality; a crucial requirement in today's health industry.

4.3.3 The Health Monitoring System

4.3.3.1 Scenario

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

4.3.3.2 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 anywhere, at any time. The system should also not depend on the patient or doctor's geographical location. The system should be scalable to handle many patients and healthcare 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.

4.3.3.3 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 or 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 deployed in the CSP. 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 this will be repeated until the user stops the app.

When a doctor wants to view the patient's health data, they simply run an application on their computer. The application makes calls to the web service to retrieve the authorised patient's health data. The doctor can then view, interpret and analyse the patient's 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, and at any time.

4.3.3.4 Data Schema

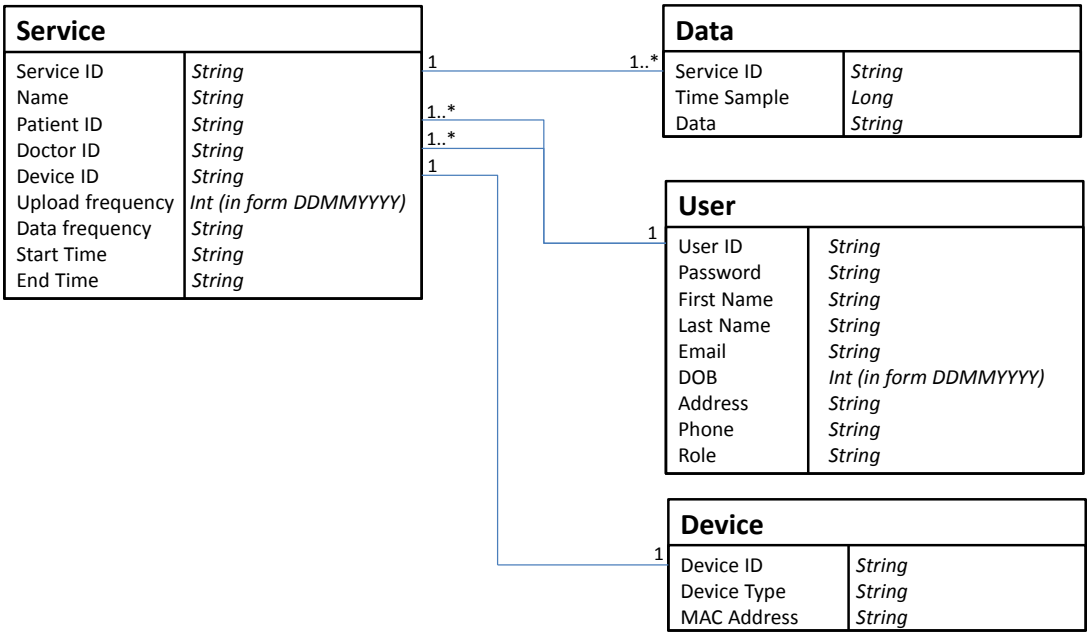


FIGURE 4.3: Health Monitoring Database Schema

Fig. 4.3 illustrates our schema for the health-monitoring system. Note that the data model is a generic, medical data service for the proof of concept, and is not focused on one particular medical service, such as a heart intensive care service.

The "User" table contains all of our system users, 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 being monitored, 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 one part of a service.

4.3.3.5 Privacy Issues using a CSP in Remote Healthcare

Since the Cloud is at the forefront of many privacy and security attacks, and many privacy attacks come from within the CSP itself, as insiders usually have direct access to data and may steal data to sell to third parties in order to gain profit, the entire database in the CSP needs to be encrypted. This means that 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, from gaining 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 patient's data, and the patient decides to revoke a specific doctor's 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 they revoke a doctor or nurse's 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 of their patient's 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 patient's data or that they will always keep the key a secret. For example, in the health domain, there are standards such as HRIPA which is followed in NSW, Australia [56] or HIPAA (Health Insurance

Portability and Accountability Act) which is followed in the US [55]. These standards aim to protect and enforce the confidentiality of a patient's health-related data and keep the data confidential from anyone unless authorised by the patient. In other words, any entity should not access a patient's health information without the patient's consent. As a result, hospitals and health organisations are reluctant to adopt Cloud technology as a privacy breach can be devastating, especially in terms of cost [136]. In our work, we provide a solution which leverages the Cloud to help ensure health data is kept private and secure.

4.3.4 Data Model and Protocol

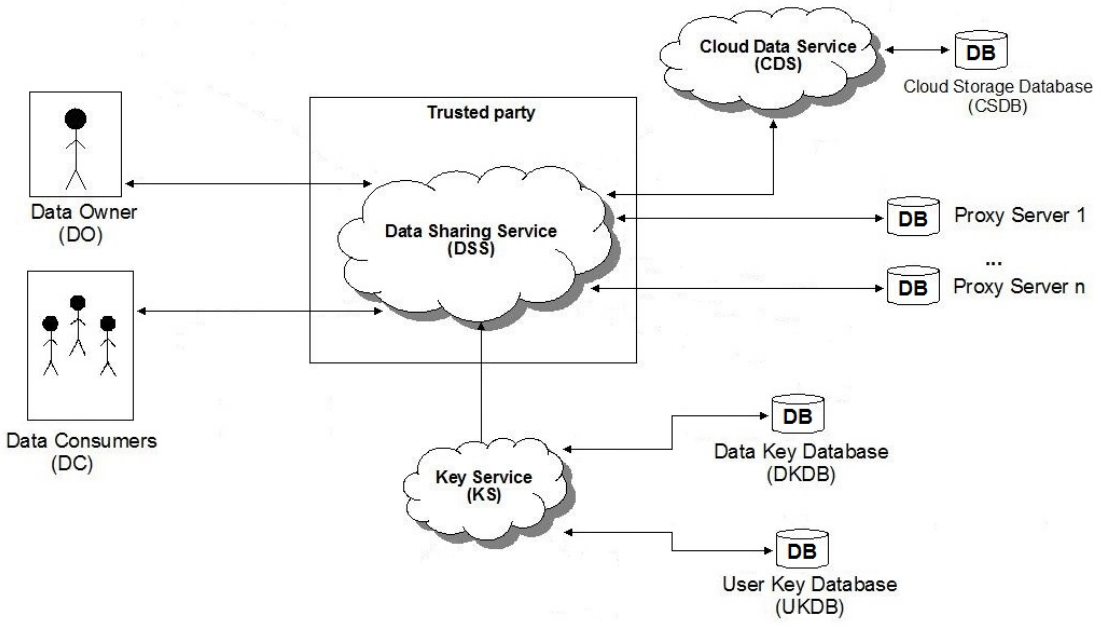


FIGURE 4.4: Health Monitoring Data Sharing Model

We now introduce our secure model and protocol that will enable data sharing amongst a group of users specified by the data owner. Our technique is not limited in any way to medical applications and can be applied to other Cloud applications. Our protocol

builds on the work of Tran et al. [25], as described in Chapter 2, and handles the collusion problem.

4.3.4.1 Data Model

Figure 4.4 enhances the eHealth monitoring system illustrated in Figure 4.3, by adding a security layer that enables efficient and 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 that the DSS is fully trusted. However, this makes it particularly vulnerable to attackers; therefore, the DSS itself needs to be protected. In order to achieve this, the DSS can be modelled as a trusted private Cloud provider that is 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 the 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 of the key pieces in the proxy database that correspond to the calling user. The key is then used to decrypt the data in the Cloud. When the data owner requests that a user's access is revoked, their key pieces in the proxies are simply removed and the original data need not be re-encrypted, nor does there 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.

4.3.4.2 Protocol

We now discuss our data-sharing protocol in detail. The protocol has four steps: *initialisation*, *consumer authorisation*, *authorised data access*, and *consumer revocation*. It is also important to note that we assume the DSS to be fully trusted, in that it will always honestly follow the protocol. As mentioned in Chapter 3, we also assume the CSDB to be honest-but-curious, in the sense that that it will carry out the steps of the protocol as expected, but is always curious to find out any sensitive information. We make use of ElGamal encryption in our work and build upon the work of Tran et al. [137] to provide a more secure platform for data sharing. The following table contains brief definitions of the abbreviations used in our protocol.

DO	Data Owner	The owner of the data; 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 the 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 that stores encryption keys which are themselves encrypted (see model)
UKDB	User Key Database	The database that stores all users, including DC and DO private keys (see model)

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 identification for the data owner (3). The DSS then sends the user identification, 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 identification, the encrypted data and the encrypted key, to the DSS (7). The DSS generates a data identification for the data (8). The DSS then sends the data identification and the encrypted data to the CDS (9) for storage. The DSS finally sends the data identification and the encrypted key to the KS (10).

Consumer Authorisation

1	DC $\rightarrow$ DO	{access_request, d_m}
2	DO $\rightarrow$ DSS	addUser(u_{DO}, d_m, $x_{\mathrm{n+1}}$)
3	DSS $\leftrightarrow$ CDS	Verify u_{DO}, d_m exists. If not, exit here.
4	for (all proxy i)	
	DSS $\rightarrow$ proxy i	{u_{DO}, key_piece_request}
	proxy i $\rightarrow$ DSS	x_i
5	DSS	Compute $x_1 + x_2 + x_3 + ... + x_n + x_{\mathrm{n+1}} = x$
		Generate $x_{\mathrm{u1}} + x_{\mathrm{u2}} + x_{\mathrm{u3}} + ... + x_{\mathrm{un}} + x_{\mathrm{u(n+1)}} = x$
		Generate u_{DC}
		Generate $\{p_{\mathrm{DC}}, b_{\mathrm{DC}}, c_{\mathrm{DC}}\}$, x_{DC}
6	DSS $\rightarrow$ KS $\rightarrow$ UKDB	{u_{DC}, u_{DO}, d_m, $\{p_{\mathrm{DC}}, b_{\mathrm{DC}}, c_{\mathrm{DC}}\}$}
7	for (all proxy i)	
	DSS $\rightarrow$ proxy i	{u_{DC}, u_{DO}, d_m, x_{ui}}
8	DSS $\rightarrow$ DO $\rightarrow$ DC	{u_{DC}, $x_{\mathrm{u(n+1)}}$, 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 identification 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 identification, the data identification, and key piece (2). The DSS then verifies whether the data identification and data owner identification exist, with a call to the CDS (3).

If the CDS returns false, then the DSS notifies the 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, are equivalent to the secret key x (5). The DSS will also generate a random user identification as well as a public/private key pair, using ElGamal encryption for the DC (5). The DSS will then send the DC's user identification, the public key and identifiers such as the DO user identification and data identification, 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).

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: $(c^r, (c^r)^{x-x_{1\mathrm{u}}-x_{2\mathrm{u}}-...-x_{\mathrm{nu}}}.k \ mod \ p)$
8	DSS	$D_{x_{\mathrm{u(n+1)}}}((c^r, (c^r)^{x-x_{1\mathrm{u}}-x_{2\mathrm{u}}-...-x_{\mathrm{nu}}}.k \ mod \ p))$
		$\rightarrow (c^r, (c^r)^{-x_{\mathrm{u(n+1)}}}.(c^r)^{x-x_{1\mathrm{u}}-x_{2\mathrm{u}}-...-x_{\mathrm{nu}}}.k \ mod \ p)$
		$\rightarrow (c^r, (c^r)^{x-x_{1\mathrm{u}}-x_{2\mathrm{u}}-...-x_{\mathrm{nu}}-x_{\mathrm{u(n+1)}}}.k \ mod \ p)$
		$\rightarrow (c^r, k \ mod \ p)$ 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 obtains the encrypted key from the DKDB via a call to the KS (2, 3). The DSS then calls each proxy server to obtain 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 obtains 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 obtains the corresponding DC's public key from 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).

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's access rights to data, he simply calls the DSS to request the revocation of the user's 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 users will be affected, since only the key pieces corresponding to the user are removed. All other key pieces corresponding to other users remain in the proxy database. Since the data

does not need to be re-encrypted, nor does there 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.

4.3.4.3 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 of 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 $E_{\{p,b,c\}}(k)$ to reveal the key k and then decrypt the data itself to reveal m. If only one proxy was used, as with the work of Tran et al. [25], 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 modelled 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. This ensures the data remains private from both the CSP and unauthorised users. Thus, it also ensures that health standards are abided by [55]. 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 of the key pieces of all of the 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 against privacy attacks. However, too many proxies can reduce the reliability of the whole system. Thus, there is a trade-off between data privacy and the reliability of the data service, which is beyond the scope of this work.

3. *Consumer Revocation* – When a DC is revoked access rights to the data, his corresponding key pieces are simply removed from all of the proxies, which 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 or the full secret key x. Also, even if a revoked user colludes with all of the proxies, he will not be able to retrieve all of the keys, since his pieces have been removed. Even if he steals another user's key pieces he still cannot recover x; consequently, the system is secure against revoked users colluding with the proxies.

4. *Large Data Sizes* – The work of Tran et al. [25] does not handle large data sizes effectively, as the ElGamal cryptography algorithm discussed can only provide cryptography operations with data up to a certain size. Since our focus is within the context of health applications, and data within the health domain tends to be very large, the protocol discussed in [25] is unsuitable. 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, as the symmetric key is unlikely to be too large.

4.3.5 Implementation and Evaluation

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

4.3.5.1 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 [138]. 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 were 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 the patient or doctor, we created a simple Java application that simply makes calls to obtain and receive data, as well as authenticate the users. %Figure 6 displays a snapshot of our system in action.

For the sensors, we used the AliveECG Heart Activity Monitor [139] with Ambu sensors [140]. The sensors connect to the Heart Activity Monitor to measure the person's ECG activity. ECG stands for electrocardiogram and can be used to measure the electrical

activity of the heart. The Heart Activity Monitor can then send the ECG data to an SD card or other devices, via Bluetooth technology.

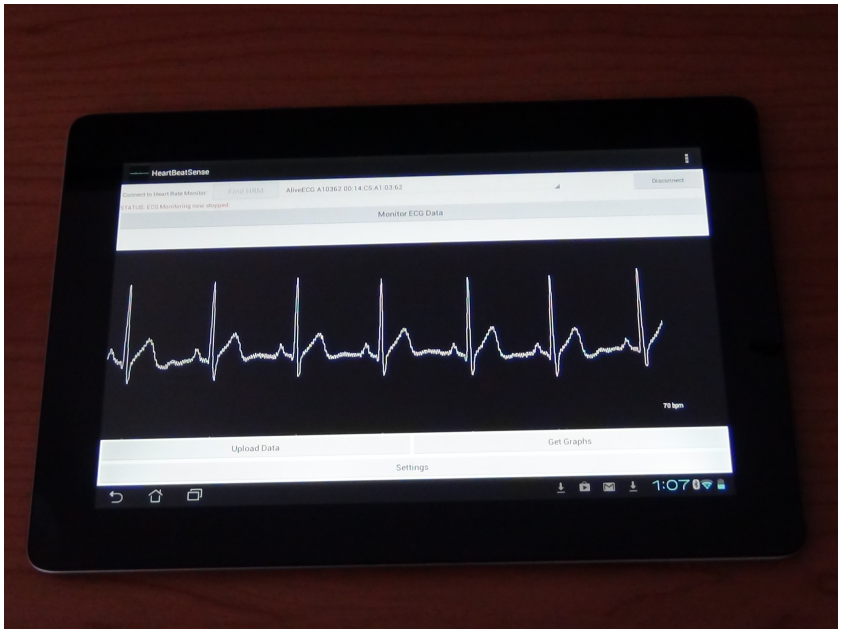


FIGURE 4.5: HearbeatSense app

4.3.5.2 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 full plaintext form and hence administrators themselves would not know the full credentials of the user. These credentials are required for using the system. Also, all data is encrypted before sending to the Cloud for storage and it remains encrypted at all times. This prevents unauthorised users from being able to retrieve any sensitive information whether the data is in transit and/or on the Cloud.

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. Thus, an attacker will not be able to find any trace of health data on the mobile device, even when stolen.

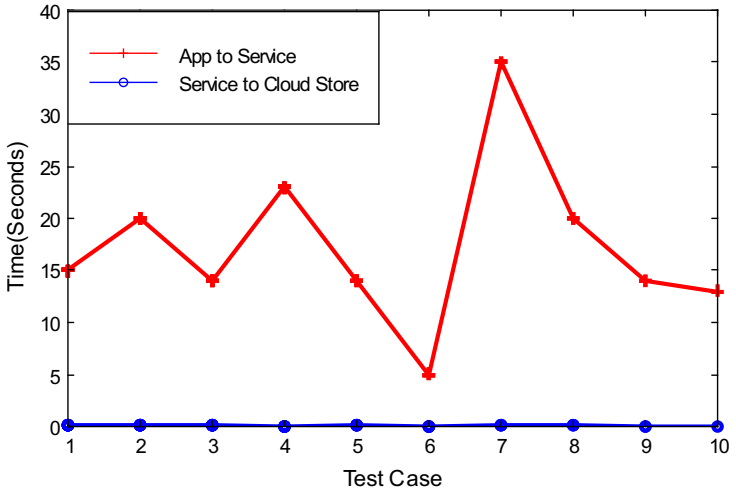
3. *Sniffing attacks* – It is possible for an attacker to retrieve data as it is being sent from the mobile client to Cloud storage. However, our system first encrypts data using a symmetric key, and the symmetric key itself is then encapsulated via the user's public key. The attacker would have to know the user's secret key (unknown even to the user) to decrypt the symmetric key and consequently the data, making our system attractive to use in such scenarios.

4.3.5.3 Performance Tests

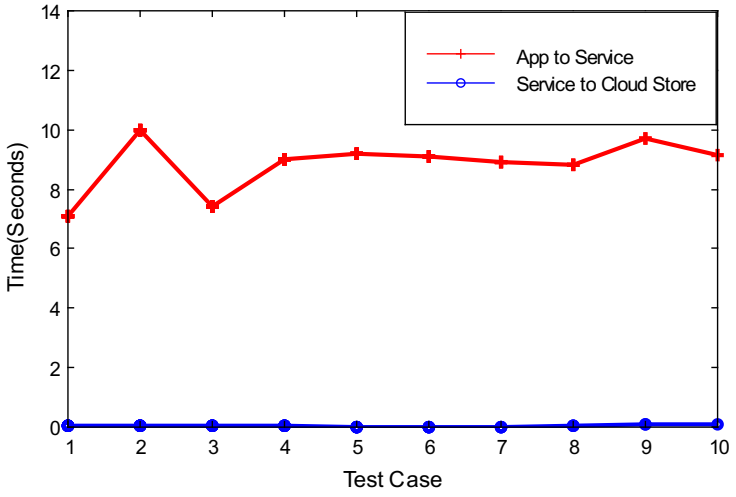
We carried out a number of performance tests on our system, primarily the uploading and downloading of ECG data. The purpose of the performance test was to test whether such a system will be feasible for use by everyday people. Each of the performance tests were carried out on 10 seconds of ECG data, or 3,000 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 the service to Cloud storage. We also 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 4.6.



(A) Uploading times with security protocol



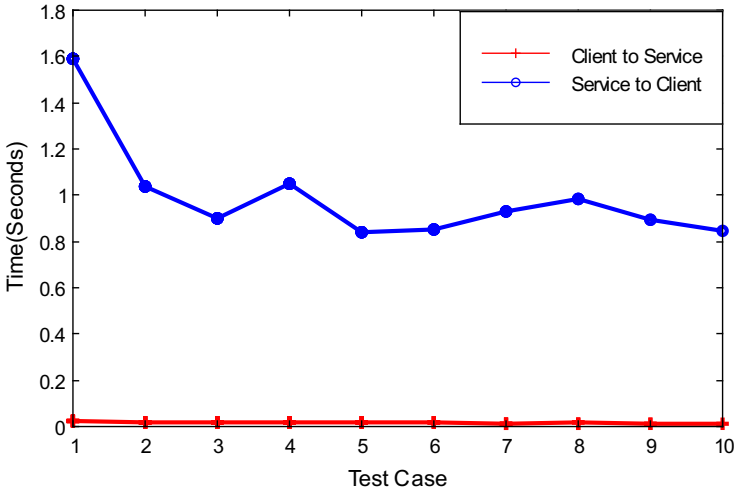
(B) Uploading times without security protocol

FIGURE 4.6: Uploading times

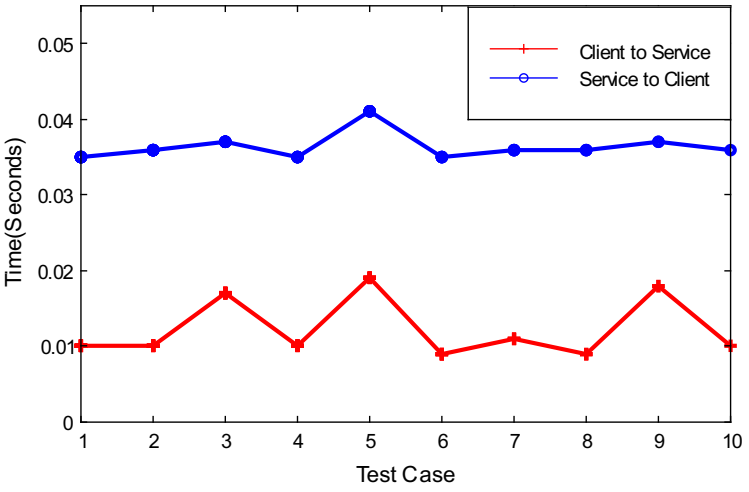
We found that in both cases, with and without using the security protocol, it took significantly longer for the app to send the data to the Cloud service, compared to the service storing the data to Cloud storage. This is expected, since the mobile device first carries out operations on the data and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

with the Cloud service. With the security protocol in place, the delay is even longer, which 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 then again without the secure data-sharing protocol. Figure 4.7 illustrates our test case results.



(A) Downloading times with security protocol

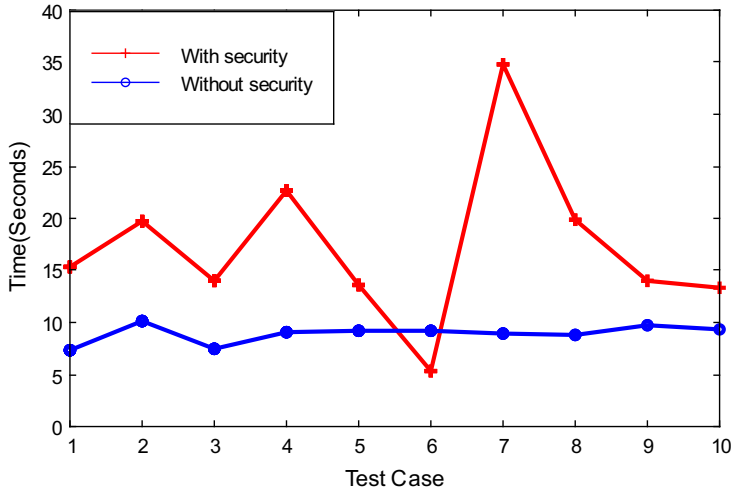


(B) Downloading times without security protocol

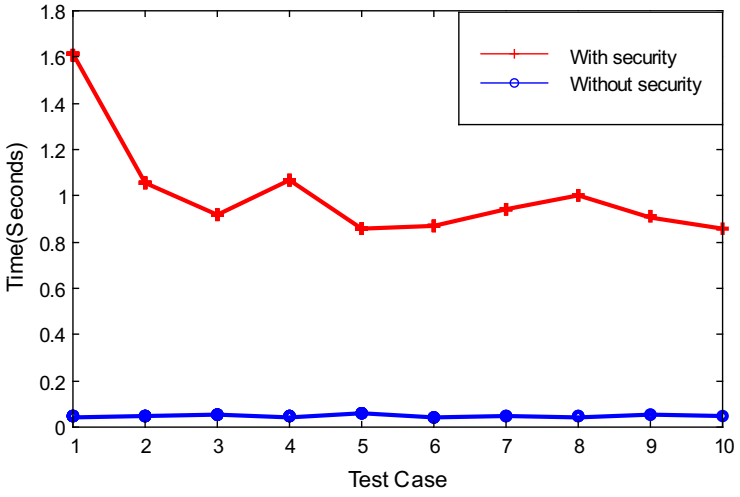
FIGURE 4.7: Downloading times

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 it to the user was slightly longer, it still returned the results in less than one second.

The total uploading and downloading times are highlighted in Figure 4.8. We measure the overhead introduced with the security protocol in place, compared to a system with no security mechanism whatsoever.



(A) Uploading overhead



(B) Downloading overhead

FIGURE 4.8: Uploading and Downloading Overhead

From our results, the security protocol introduced significantly more overhead compared to without the security protocol. 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 approximately 1-2 seconds. With the security protocol in place, the average download time was 1.00 second, with a standard deviation of 0.21 seconds; without the security protocol, the average was 0.05 seconds, with a standard deviation of 0.01 seconds.

4.3.5.4 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 a mobile device was used to send large packets of data. The patient retrieves data on a desktop PC and since a PC has much more 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 could have contributed to the 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, might be concerned regarding the time it takes to download data from the database, as they may not want to wait for long periods of time to retrieve ECG data for analysis. From the performance tests, the downloading times were very small and thus efficient, making our system feasible for deployment 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 encryption of the data, 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 while their ECG is being monitored. When downloading data, it is more important that data arrives quickly so that doctors can provide more rapid treatment options. Hence, this makes our solution feasible for use 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, one-minute intervals, five-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.

4.4 Secure eHealth Self Management in the Cloud

We now apply our key-partitioning technique in the environment of eHealth self-management. We further improve on the previous section by involving doctors and potential users of our system, in the context of mental health. The following work is based on the paper published in JMIR [141].

4.4.1 Introduction

A new type of socio-technical challenges has arisen with the advent of eHealth and Big Data technologies. For example, ubiquitous and wearable health systems collect data through sensors and mobile apps, and store it in the servers of multiple commercial service providers. Furthermore, a growing number of people share this sensitive medical information through social networks such as Facebook and Twitter. This is significantly different to the traditional health service, where service providers kept tight control over patient data.

It has been argued that these new technologies can lead to positive health outcomes, as they are evidence of people self-managing their illness [142]. Some of the ways in which self-management can have a positive impact include supporting the patient's motivation to look after their health, greater levels of engagement and understanding about the condition.

Furthermore, these new technologies may help improve population health by helping researchers learn about the drivers of different pathologies, or how people's behaviour is affected by social influence and public health promotion campaigns [143]. The information posted to social networks can prove invaluable to assisting doctors and counsellors to better understand patient behaviours and symptoms, and can help to provide support and/or consultation. Social networks are now being leveraged to provide people with a better lifestyle and health, without the need to continually visit the doctor's clinic.

However, privacy [144], trust and security issues associated with health data make patients hesitant to post sensitive health information and share it with health providers [78]. Since data is not ephemeral, it will be stored in servers and shared, and all stakeholders need to worry about its lifecycle; not just who can access and manage it at a particular point in time, but also who will be able to do so in the future. There is a strong need to provide patients with a guarantee that their sensitive health information will only be visible to the doctors and/or counsellors, or others they wish to share it with at a particular point in time.

Microsoft HealthVault [145][146] provides a next step to allowing patients to store and manage their health and fitness information, as well as share the data securely with their friends and family. The encryption is done within HealthVault and does not rely on the patient to generate and distribute keys. The patient can decide who specifically

can view his health information. In comparison, our system gives the patient greater control over their health information and they can choose to store their health data on any Cloud service provider that they wish. The patient can distribute encryption keys themselves to people they wish to share the data with, and do not need to rely on commercial services, which may be untrustworthy.

4.4.1.1 Mental Health Scenario

For the evaluation discussed in this study, we created a fictitious, but quite common scenario: collecting data and providing support.

The best way of designing and then evaluating a security feature is through a minimum viable application in a realistic scenario. The security feature would be applicable in other scenarios, but the reification into concrete terms with users, and evaluate the design on scalability and non-functional requirements. Our application emulates one where data is collected to provide support to people at risk of mental health issues in the workplace.

We chose this scenario because it was relevant to our research and because of its significance. There is evidence of increased work stress, sleep disorders and depression in the workplace [147]. As a result, there is a need for the means through which an organisation can provide support and feedback in a convenient and secure manner. In order to detect people at risk, information is needed. This information may come from the people themselves or their friends, reporting problems at home or at work that are affecting their lives and their mental state. It could also come from managers, OH&S reports, or other sources such as other eHealth systems. Regrettably, in many cases, people fail to seek help when they need it due to a number of reasons, including the lack of time or

access to resources, stigma, and trust. For example, regularly visiting a clinic can be costly for patients and doctors. For patients, this also involves the time and effort spent visiting the clinic, particularly for rural and disabled patients. For doctors, eHealth may allow them to prioritise differently and tend to patients who cannot travel. Others have highlighted the possibility of using eHealth services to reduce healthcare costs [148].

We also speculated that certain aspects, characteristic of mental health issues, would make the importance of trust and privacy more relevant to users. Trust and stigma also make it harder for people to seek help or share information about their mental health. In workplace wellbeing programs, for example, employees might be less likely to share information if they feel that it could be used by their employers. Trust is in great measure a consequence of the software design of systems and apps used to collect and manage the data.

Figure 4.9 highlights the methodology we used to carry out our work. Our method was based on the waterfall model. We first define the requirements of our work. That is, to develop a system that allows patients to share their personal health information securely and privately, while ensuring the system is usable. We use a fictitious scenario to assist in defining the requirements of the system. We then review state-of-the-art literature to explore the existing works/technologies that attempt to address this. We then build on these works and develop new technology. Finally, we test our developed system through performance and scalability tests and evaluate the system in terms of usability.

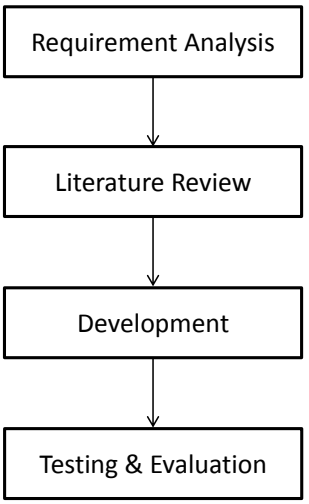


FIGURE 4.9: Development Method

In this project, we evaluate the security model through a prototypical smartphone app. Using a mobile phone app, patients can report and receive help, wherever they are. In the field of mental health, for example, studies have also shown that the use of smartphone apps can support significant reductions in depression, stress and substance use [149].

Our contribution is a new way of protecting data, without revealing the full encryption key to both the user and the Cloud provider. We propose a system that is designed to be highly scalable, providing the ability to share data with many users, such as doctors and nurses, while allowing the simple revocation of a user without the need to re-encrypt the data every time a user revocation occurs. We focus on creating a secure and usable system that will enable patients to share mental health information with doctors and mental health specialists, from the comfort of their own home.

4.4.2 Data Model

Figure 4.10 demonstrates our system data model.

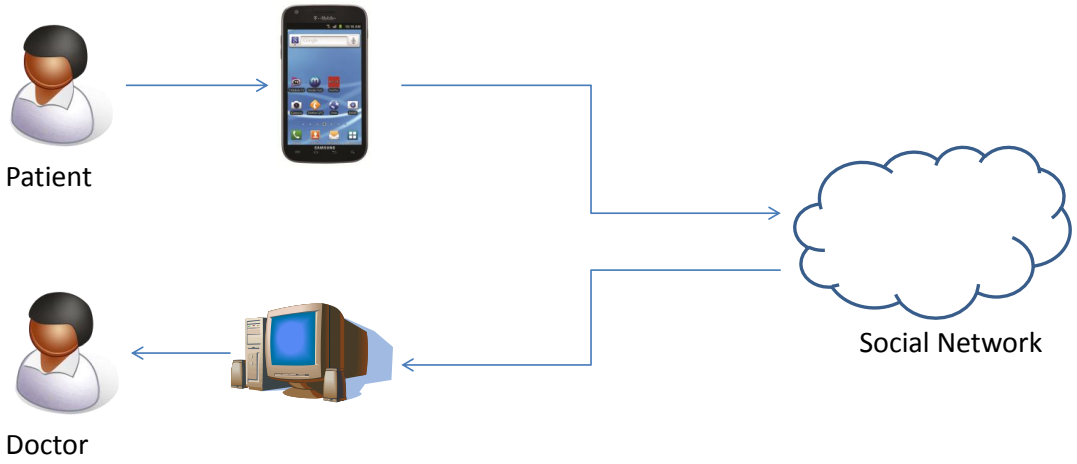


FIGURE 4.10: Data Model

The model we used to test our application assumes a patient who monitors and tracks their health and activity data through a smartphone app. The app may then connect to, and store the data in, a social network such as Facebook, Fitbit or other Cloud-based service provider using an Application Programming Interface (API). An authorised doctor can log into and retrieve the patient's data, and use it for analysis and diagnosis. We use a social network to demonstrate our ideas. The patient and doctor are then assumed to have logged in to their social network account using their credentials.

For the sake of our evaluation, we have simplified the application so that it provides the most common features found in commercial products. Our prototype app allows the patient to enter a text value (e.g., the description of an activity), a number value (e.g., the amount of time spent) and an image. The app also includes a button used to encrypt the text, number and image and send the data to a Cloud server which is used to represent the social network. In our work, we developed a local Cloud server that does encryption/decryption operations.

One of the main limitations of our work is that current social networks cannot automatically carry out encryption/decryption. However, we mainly wanted to demonstrate

the potential capability of our system should a social network provide this feature in the future. Another limitation of our work is that, once the doctor has fully decrypted the patient's health data, there is no way to revoke access. This is currently beyond the scope of our paper. The doctor however, would not be able to view any further health information posted by the patient.

One of the main goals of our system is to make it simple to use for both patients and doctors. Our system is not designed to replace existing health record systems but provide a convenient way for patients and doctors to communicate with each other remotely whilst ensuring privacy and security of health data. In terms of privacy, we offer a solution that enables the patient to define who can access their personal health data. We do not focus on the other aspects of privacy such as determining when the data was accessed, how the data was accessed, and to what extent the data is communicated. In terms of security, we provide solutions to availability through the use of the Cloud and confidentiality in terms of allowing only authorised doctors to access the data. We do not focus on integrity or accountability in this work.

4.4.3 Protocol

To describe the protocol, we assume that the patient's public and private key pair has already been generated and stored in the app.

Data Storage

The patient first runs the prototype app and inputs a text string and a number value, and uploads an image onto their smartphone. When the patient presses the "Send" button, the app will then generate an arbitrary symmetric key and encrypt the text, number and image. The symmetric key will then be encrypted using the public key. The

encrypted data contents and encrypted symmetric key will then be sent to the social network, for storage.

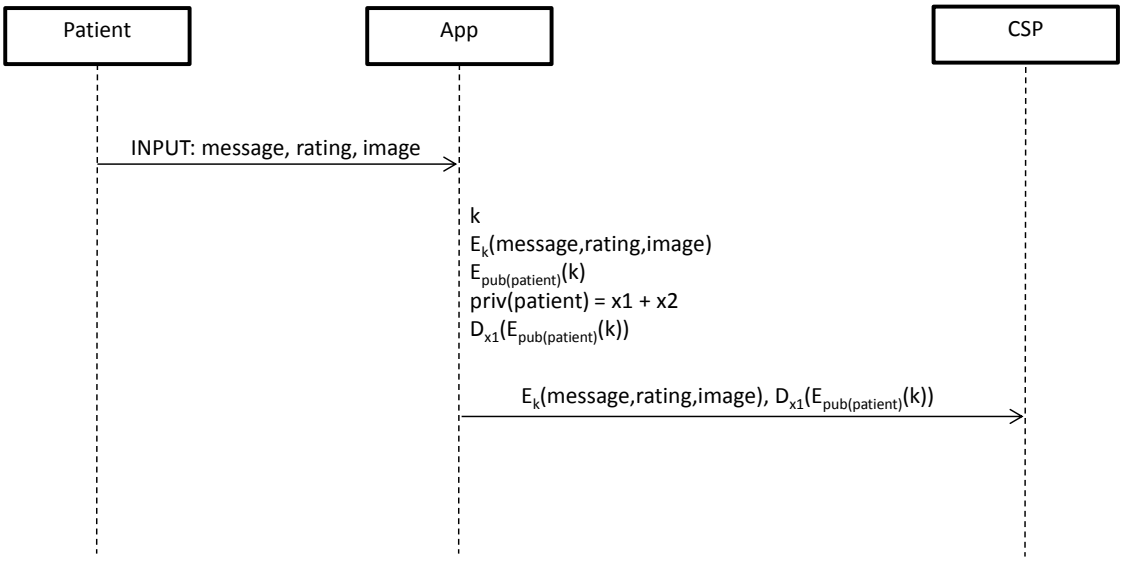


FIGURE 4.11: Data Storage Protocol

Data Sharing

When the patient decides to share the data with a doctor, they press the "Share" button on the app and enter the doctor's social network username. The app will then partition the patient's private key into two random parts. The first partition will be sent to the social network and the other will be sent to the doctor. By doing this, the untrusted social network has no knowledge of the full private key, since the other partition is stored on the doctor's local machine.

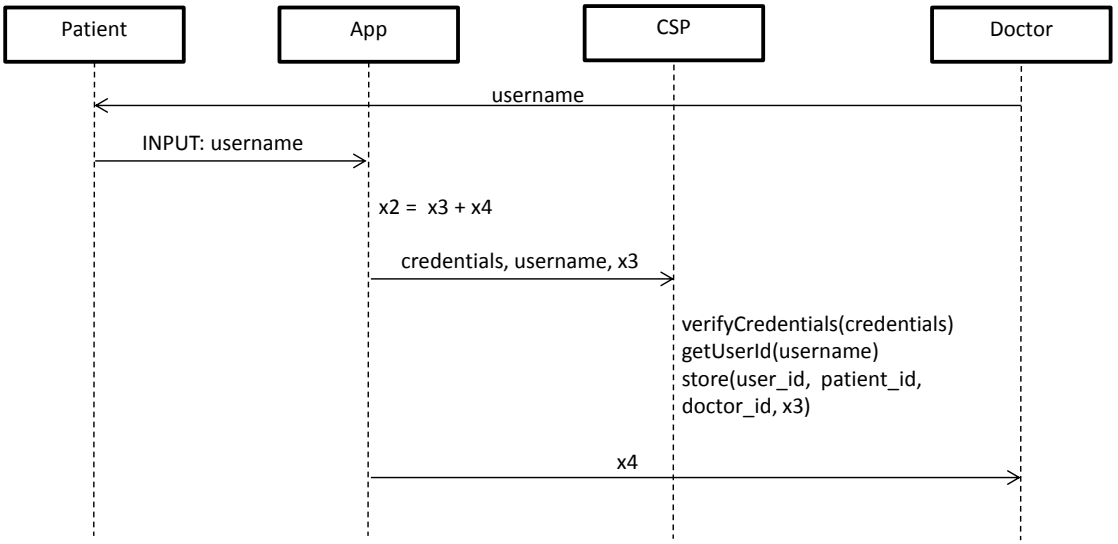


FIGURE 4.12: Data Sharing Protocol

Data Access

When the doctor wishes to access the patient's data, they simply call the social network to retrieve the data. The social network partially decrypts the symmetric key using the partial key supplied by the patient, and sends the encrypted data contents and partially decrypted symmetric key to the doctor. The doctor uses the partial key supplied by the patient to fully decrypt the symmetric key and finally decrypt the data contents. The standard method of accessing data involves the data consumer downloading the encrypted data from the Cloud and decrypting the data on his own machine, using the encryption key supplied by the data owner. In our protocol, the data consumer does not have access to the other half of the key, which prevents the data consumer from ever knowing the full encryption key.

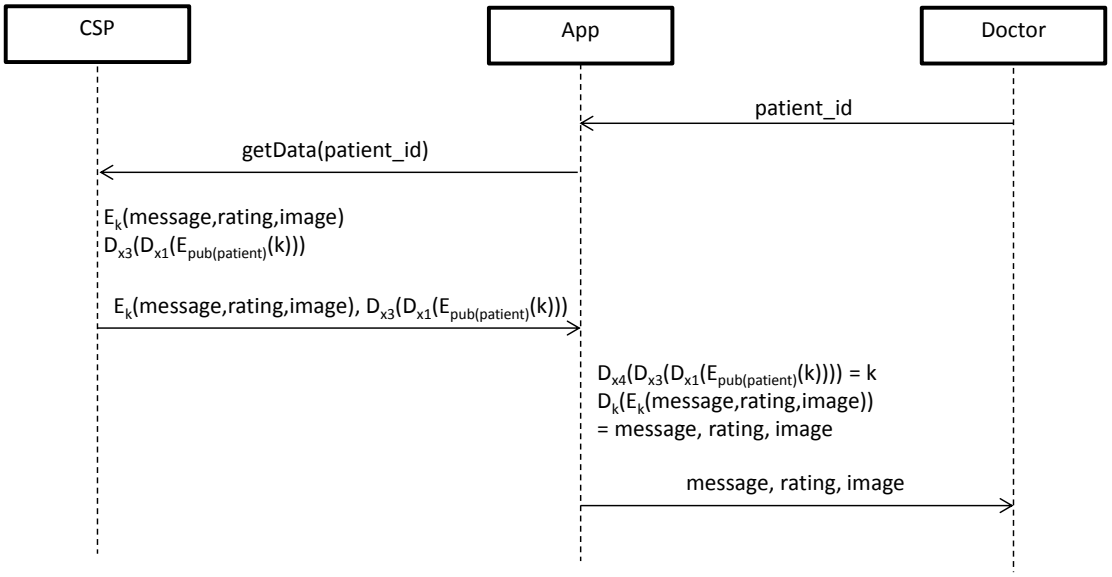


FIGURE 4.13: Data Access Protocol

Access Revocation

When the patient decides to revoke a specific user's access to his eHealth data, the patient sends a request to the social network platform to remove the doctor's partial key entry from storage. If the doctor attempts to download the data from the social network, he will only see the encrypted text ("ciphertext"). The doctor will not be able to fully decrypt/read the data without the partial key. In the trivial solution described earlier, the data owner would have to re-encrypt the data and re-distribute the new encryption key to all of the remaining consumers in the group, thus placing a burden upon the data owner. In our solution, since the data consumer has no knowledge of the other half of the key partition stored in the Cloud, the data owner would simply have to delete that key partition. Thus, he need not worry about re-encryption and the re-distribution of keys.

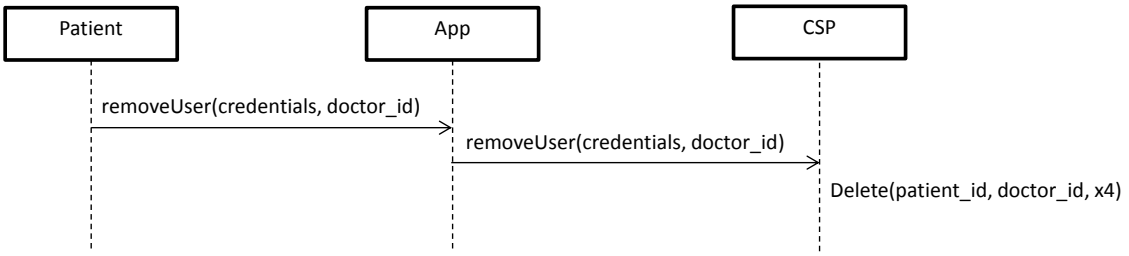


FIGURE 4.14: Access Revocation Protocol

4.4.4 Security Analysis

Formal Analysis

To verify the security of our protocol, we have used an automatic cryptographic verifier tool called ProVerif [150], which has been used extensively in research work [151]. The tool tests the protocol against all types of adversary attacks, such as man-in-the-middle attacks. Our protocol was found to be secure against such attacks. We tested the storage of eHealth data by the patient and the retrieval of data by an authorised doctor. Specifically, we tested the Data Storage and Data Access phases of our protocol. We also note that the protocol phases we test here are similar to the protocol phases in Section 4.3.4.2. The differences mainly being the scenario and technology used.

We first modelled the behaviour of the symmetric and asymmetric encryption, ElGamal encryption/decryption, and digital signatures. Below is an example of how we modelled the ElGamal encryption and decryption, as well as key partitioning.

```
fun elgenc(bitstring, epkey): bitstring.
reduc forall m: bitstring, esk: eskey, elpk: epkey;
   elgdec(elgenc(m,epk(esk)), esk, elpk) = m.
reduc forall e: eskey, e1: eskey; add(e1, sub(e, e1)) = e.
```

We then modelled the DO by following the logic of the protocols defined in this chapter. In other words, we modelled the DO generating a new symmetric key k and encrypting the data s with that key. We then modelled the DO encrypting the symmetric key with the ElGamal public key *epkey*. The DO then partially decrypts the symmetric key using the key partition $els1$. The DO sends a bundle that includes the encrypted data a and partially decrypted symmetric key e to the Cloud, via a public communication channel c. The DO partitions the already partitioned key $els2$ into two parts and sends this along with the ElGamal public key $elpA$ and client identification x to the Cloud for storage, and sends the remaining key partition $elsCc$ directly to the DC, via the private channel w. The code below shows how this was modelled in ProVerif.

```
let dataOwner(pkA:pkey, skA:skey, pkB:spkey, elsA:eskey, elpA:epkey) =
   new k:key;
   let k1 = tobitstring(k) in
   let a = senc(s, k) in
   let b = elgenc(k1, elpA) in
   new els1:eskey;
   let els2 = sub(elsA, els1) in
   let e = elgdec(b, els1, elpA) in
   new data_id: bitstring;
   let d = bundle(a, e, data_id, els1) in
   out(c, d);
   in(w, x:bitstring);
   new elsCs:eskey;
   let elsCc = sub(els2, elsCs) in
   let o = consumerBundle(x, elsCs, elpA) in
   out(c, o);
   out(w, elsCc).
```

The Cloud provider model simply retrieves the encrypted data bundle and consumer key bundle from the DO, via the public communication channel c. When requested by the DC, it would further partially decrypt the encrypted symmetric key h using the DC key partition that was sent in the *clientBundle*. Another bundle containing the encrypted data and the new, partially decrypted symmetric key will be sent to the DC via the public communication channel v.

```
let Server(pkB:spkey, skB:sskey, pkA:pkey) =
    in(c, f:bitstring);
    let g = getData(f) in
    let h = getKey(f) in
    let i = getDataId(f) in
    let prk = getPartialKey(f) in
    out(v, i);
    in(c, clientBundle: bitstring);
    let cid = getConsumerData(clientBundle) in
    let csk = add(prk, getConsumerKeyPartition(clientBundle)) in
    let cpk = getConsumerPubKey(clientBundle) in
    let k = elgdec(h, csk, cpk) in
    let l = bundle(g, k, cid, csk) in
    out(v, l).
```

Finally, the DC model retrieves the key partition *cek* from the DO via the private communication channel w. The DC then retrieves the encrypted data bundle after making a request, and uses the key partition *cek* to fully decrypt the partially decrypted symmetric key and then fully decrypt the encrypted data to reveal s. The code, below, highlights this exact process.

```
let Client(client_id:bitstring, elpA: epkey) =
```

```
in(v, data_id:bitstring);

out(w, client_id);

in(w, cek: eskey);

in(v, serverBundle:bitstring);

let n = getData(serverBundle) in

let enc_key = getKey(serverBundle) in

let partk = add(cek, getPartialKey(serverBundle)) in

let r = elgdec(enc_key, partk, elpA) in

let t = tokey(r) in

let (=s, k:key) = sdec(n, t) in

0.
```

Each of the processes of the DO, Cloud provider and DC were run simultaneously, to simulate realism.

```
( (!dataOwner(pkA,skA,pkB, elsA, elpA)) | (!Server(pkB,skB,pkA)) |

(!Client(client_id, elpA)) )
```

The figure below illustrates the security mechanisms used in our system. The mobile app requires username and password credentials in order to be able to use our system. All health data that is sent to the social network is encrypted and is sent securely via HTTPS/SSL. The social network also has privacy controls which the patients can adjust to suit their needs.

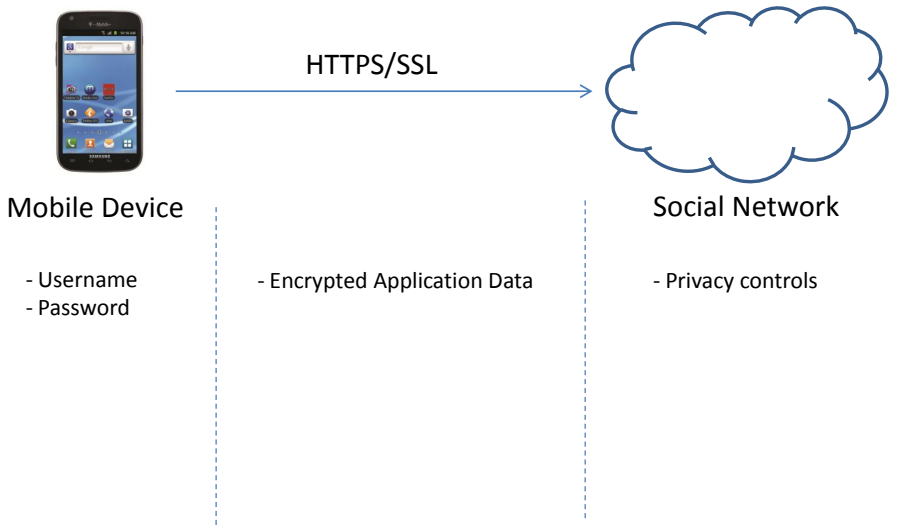


FIGURE 4.15: Secure Communication Paths

Informal Analysis

We now provide a brief security risk analysis of our work.

- *Insider Attacks* - Our protocol prevents insider attacks since the data is never fully decrypted in the untrusted Cloud at any circumstance. The data remains encrypted at all times on the untrusted Cloud Servers as well as on untrusted public communication channels.

- *User Revocation* - Revocation of a doctor from data access can be achieved efficiently without having to re-encrypt the data each time. The doctor's key partition is simply removed from the Cloud storage. This way, if the revoked doctor now attempts to access the health data, he will not be able to retrieve the full plaintext without the remaining key partition.

- *Update Secrecy* - Since health data is constantly changing, the patient may wish to update their health data. This is made possible in our protocol, as long as the updated version is encrypted with the same symmetric key that was used to

encrypt the original health data, the patient may update their health data any number of times as they wish. Hence making our solution feasible to be deployed in a real-world scenario.

- *Mobile Stealing* - In the event someone steals the patient's smartphone, they will not be able to access the personal health information as they would need to know the patients credentials such as email id and password in order to access the smartphone app. Hence, a patient does not need to be tied down to only one smartphone device and can keep changing their device as often as they would like without any loss of personal health information.

4.4.5 Usability Analysis

In total, we recruited 5 medical professionals and 15 students to carry out the usability testing of our eHealth application. According to Nielsen [152], the minimum number of participants required in a usability study is 5. We chose to recruit medical professionals, due to their experience with patients and health issues. They were also the most likely potential users of our system. The medical professionals included two doctors, two medical officers and one medical intern. We chose also chose young people (ie. students) since they were the most likely to use smartphones and would be likely potential end users of the system. We recruited students aged over 18. 17 were over 25 (85%) and 3 were within 18 - 25 years of age (15%). We obtained ethics approval to carry out the study. All students reported having a fair amount of experience using mobile apps.

To carry out the usability tests, we provided participants with a 4" LG smartphone with Android OS and a 10" ASUS Eee Pad tablet, which contained our secure eHealth app.

We also launched our web service, which would interact with the smartphone to store and retrieve eHealth data and enable the sharing with, and revocation of, other users.

All 20 participants were given the same demo. Each participant was first introduced to the main idea of our secure eHealth system. We then asked the participants to carry out simple tasks such as:

- Report current mood,
- Share information with another user,
- Show that the other user can view the user's mood submission,
- View mood submissions, etc.

Each participant was told that their mood submission was encrypted, and they were shown the backend of their stored mood submission. Participants then carried out our trust and usability questionnaire.

We asked the participants to think aloud while taking notes. Finally, participants answered a short questionnaire, with questions related to trust and security [153] [26], and usability (the USE questionnaire [154]). The questionnaire asked the participant to assess our system based on trust and security, ease of use, and satisfaction, based on a 7-point Likert scale.

We investigated the relationship between how trustworthy and secure our system is and how useful our system is to everyday users.

We have illustrated the user interface of our MindFeedback prototype app in Figure 4.16. The figure displays the screenshots of the login view for both patient and doctor (a),

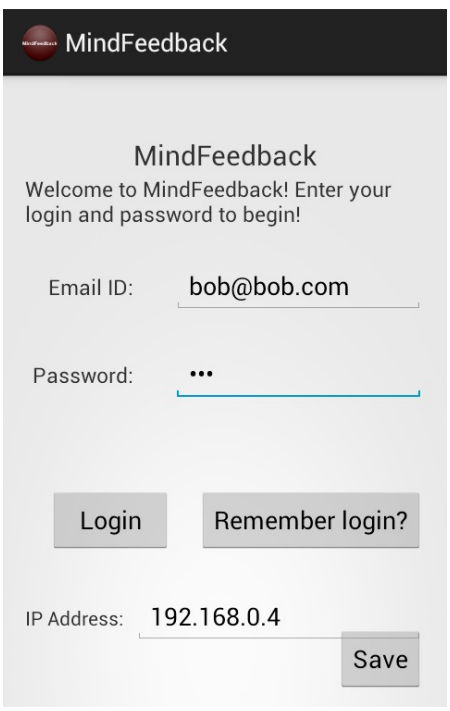
Question	Demographic Questions
1	What is your age range?
2	Are you male or female?
3	What is your ethnicity?
4	What is the highest level of school you have completed or how the highest level of degree you have received?
	7 Point Lickert Scale Questions
	Trust and Security
5	When I'm connected to the Internet, I am concerned about exposing my health information to the public.
6	I am not too concerned about what others see when I post my health-related information on the Internet.
7	This system has made me more aware of what I may be exposing to others on the network.
8	I feel safer when using the system.
9	Personal information which I input is managed carefully and will not be leaked to the outside.
	Ease Of Use
10	It is easy to use.
11	It is user-friendly.
12	It requires the fewest steps possible to accomplish what I want to do with it.
13	Both occasional and regular users would like it.
14	I can use it successfully every time.
15	The app is tedious.
16	I require written instructions to use it.
17	It is difficult to recover from mistakes.
	Satisfaction
18	I am satisfied with it.
19	It works the way I want it to work.
20	The app could be better.
21	The app wasn't as satisfactory compared to other health apps.
	Feedback
18	Would you like to provide any other feedback on our system?

the patient view when entering the mood information (b) and the doctor view when receiving the patient data and optionally providing feedback (c).

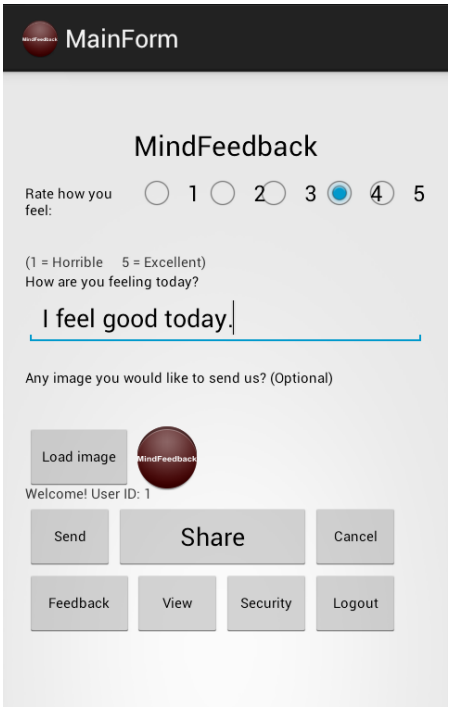
Evaluation by Potential Users

We conducted a quantitative-based usability evaluation. Figures 4.17 to 4.19 illustrates the responses from potential users for trust/security, ease of use and satisfaction respectively. The X-axis represents the weighted average of responses from potential users. Values above 5 indicate that participants agreed and values below 5 indicate they disagreed.

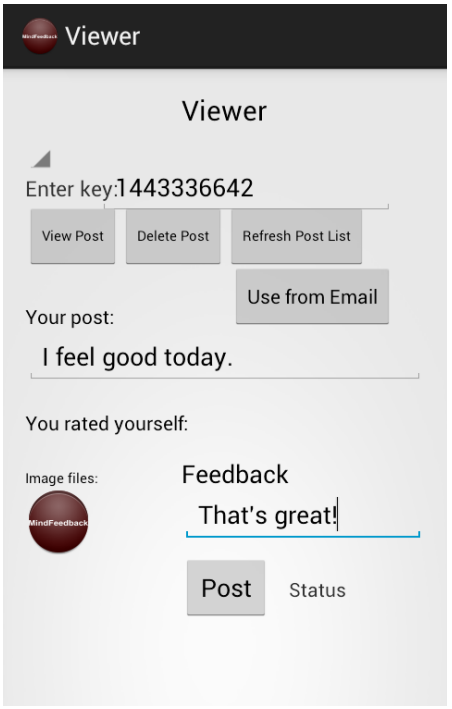
From our trust and security results, 10 out of 15 participants (67%) had at least some



(A) a



(B) b



(C) c

FIGURE 4.16: Screenshots of Mindfeedback app with login view (a), patient view (b) and doctor view (c).

concern over what others see when they post health-related information on the Internet. 12 out of 15 participants (80%) felt that their data would be kept private and secure when using our system, and were also made more aware of the type of information that they may be exposing over the Internet. In regards to whether their personal information will be managed carefully and not leaked to the outside, nearly half of the participants agreed. Participants did mention that some form of training or a video demonstration would have communicated the security of the system a lot more effectively.

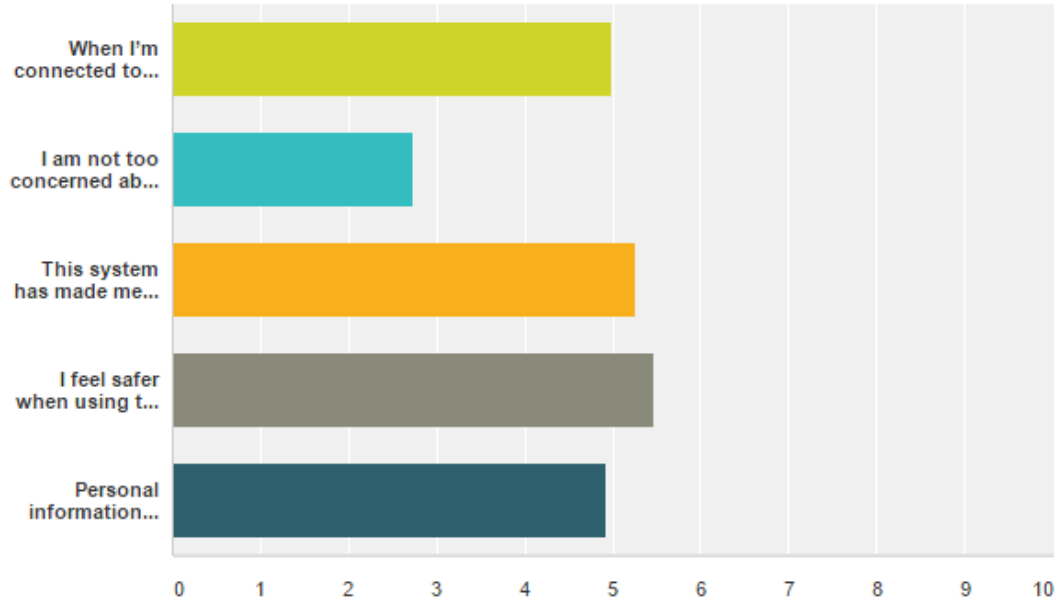


FIGURE 4.17: Security Responses from potential users

From our ease-of-use responses, we found that 11 out of 15 participants (73%) found our system easy to use and learn, user-friendly, and were able to use it successfully, every time. However, 4 out of 15 participants (27%) did find the app a little "tedious" to work with initially, and required some of instructions to understand the system a little better. Overall, the satisfaction of the app was mostly positive. 13 out of 15 participants (87%) were satisfied with our app and found that it worked in the way they wanted it to. However, most agreed that the app could have been improved. For instance, participants

provided feedback that the app could have had a better looking and a more intuitive interface.

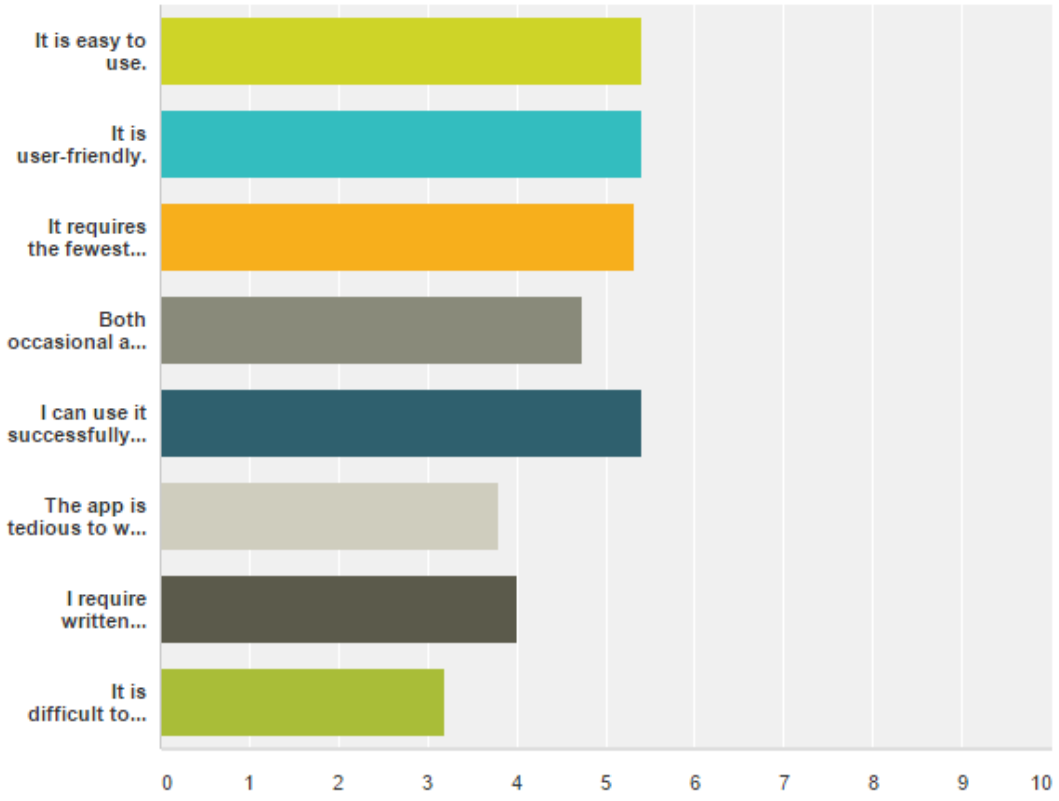


FIGURE 4.18: Ease-Of-Use Responses from potential users

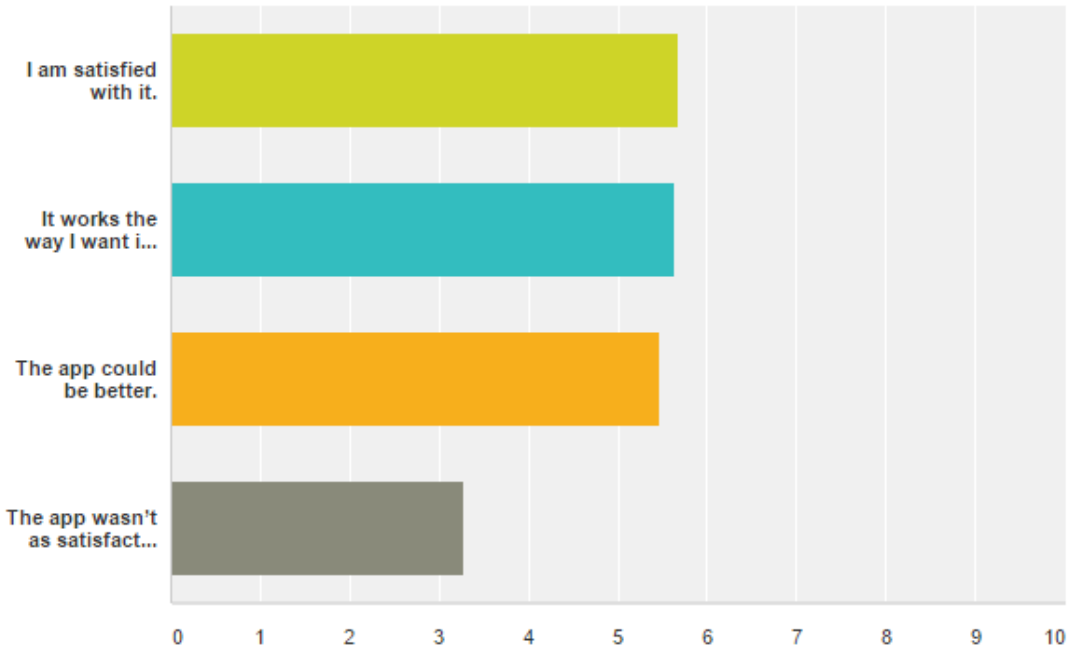


FIGURE 4.19: Satisfaction Responses from potential users

Evaluation by Medical Professionals

We also performed an identical usability evaluation with the five medical professionals. Figures 4.20 to 4.22 illustrate the responses from medical professionals for trust/security, ease of use and satisfaction respectively. Similarly, the X-axis in these figures represent the weighted average of responses from medical professionals. Values above 5 indicate that participants agreed and values below 5 indicate they disagreed.

From our trust and security results, 2 out of 5 participants (40%) were strongly concerned about exposing health information over the Internet while the rest were partially concerned. After using our app, 4 out of 5 participants (80%) felt that the personal information they entered into the app would not be leaked to the outside. Results were mainly positive about feeling safer when using the system and being more aware of what they might be exposing to others on the network. In terms of feedback, participants reported that users would not understand the key process and that it might need to

be accompanied with images, for better understanding. We needed to better showcase the trivial solution of data sharing, as described in the introduction, and how our system solves the issues of the solution. Another participant reported that the two-part encryption was ideal.

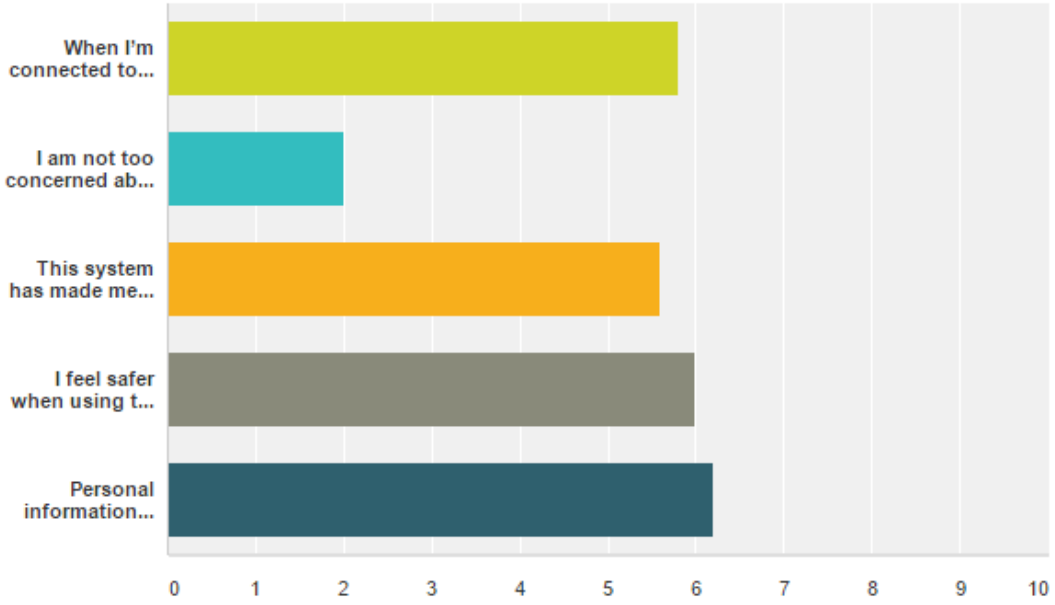


FIGURE 4.20: Security Responses from medical users

In terms of ease of use, 3 out of 5 participants (60%) agreed that the app required the fewest steps possible, in order for them to accomplish what they wanted to with the app. Results were also mostly positive, in terms of the app being user-friendly, easy to use, and the ability to use it successfully, every time. However, a few participants agreed that some form of written instructions was needed to make this app usable. Overall, medical professionals found our system satisfactory. 4 out of 5 participants (80

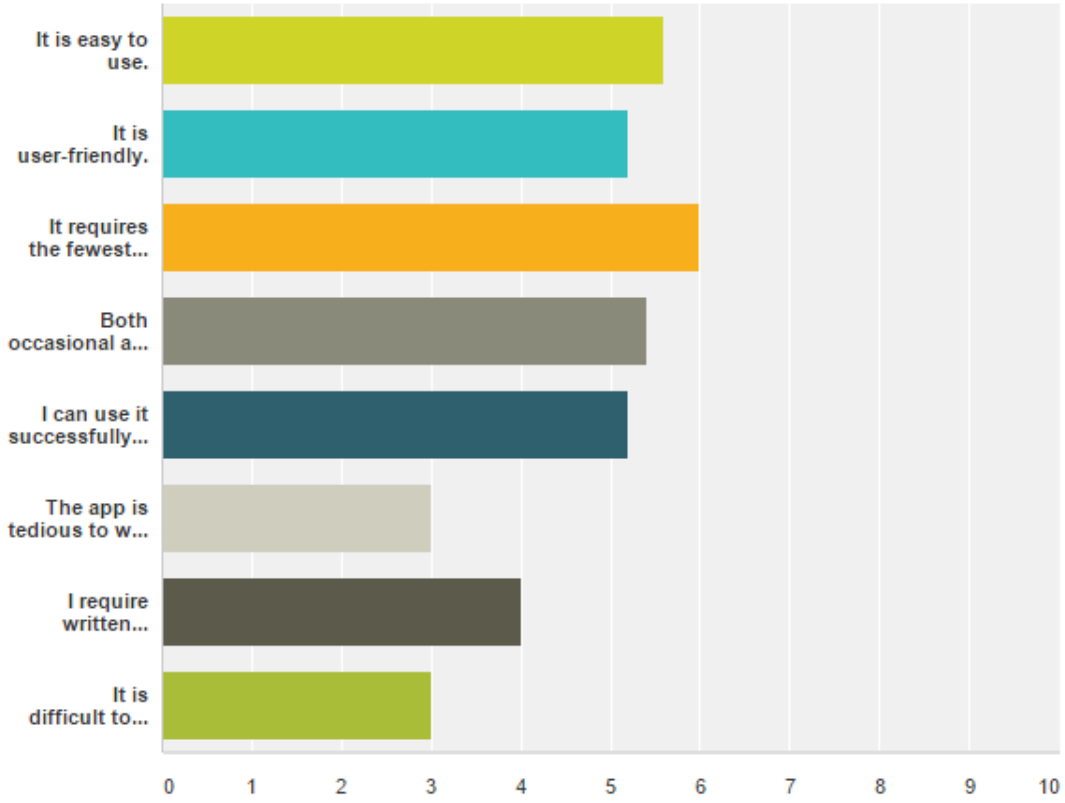


FIGURE 4.21: Ease-Of-Use Responses from medical users

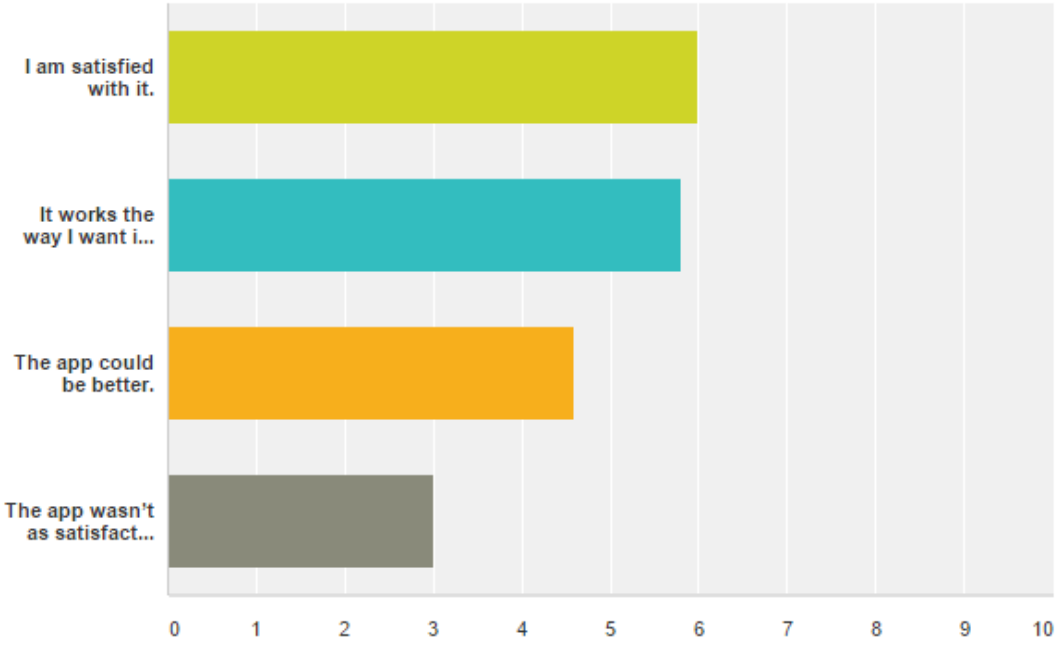


FIGURE 4.22: Satisfaction Responses from medical users

Similar to the potential users, 2 out of 5 medical professionals (40%) also felt that the app could have been better. For instance, most of the feedback involved improving the user interface. Doctors reported that a notification system for the app would have been very handy. The notification system could pop-up or beep and alert a patient when a doctor has provided feedback. For more serious medical problems, the notification system could forward the patient's request to an emergency unit or mental health crisis team, in the event that the doctor cannot respond out of hours. Most doctors provided positive feedback about the security of the app. One participant noted that the two-part encryption might be frustrating for older patients, and that such a system perfectly suits teenage patients.

4.4.6 Performance Tests

As measure of performance we tested the overhead introduced in our system, regarding the storage and retrieval of eHealth information, with simple AES encryption/decryption of similar text data. In regards to the eHealth information, we entered dummy data to the MindFeedback app. Figure 4.16 contains the screenshots of the dummy data we used. We carried out 20 test cases and measured the time taken for each test case. To carry out the tests, we used the ASUS Eee Pad Transformer Prime TF201 Tablet [138] with Android OS, to run our MindFeedback app. We used a HP Notebook running Windows 8 with Intel Core i5 and 4GB RAM, to run the AES encryption/decryption operations and to also interface with our app, in order to retrieve performance time information from MindFeedback.

In our performance tests, we measured the overhead introduced by our system compared to a simple AES encryption and decryption operation. We first measured the overhead

introduced by uploading the patient's health data to the Cloud server. Figure 4.23 illustrates the results of our upload performance tests.

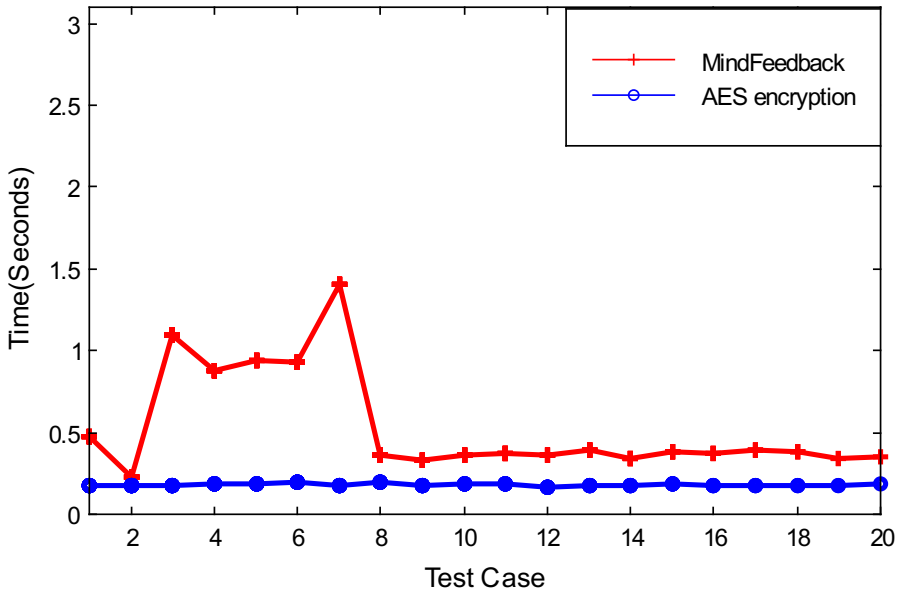


FIGURE 4.23: Upload Overhead

The diagram clearly highlights the overhead of our system compared to a simple AES encryption solution. The mean time for the simple AES symmetric encryption was 0.18 seconds, with a standard deviation of 0.006 seconds. However, the mean time for the MindFeedback tests was 0.485 seconds, with a standard deviation of 0.09 seconds. The overhead is accounted for the additional encryption of the symmetric key, followed by the partial decryption of the symmetric key through the ElGamal encryption algorithm. There was also some network latency overhead.

We also measured the overhead introduced by our protocol for the download or retrieval of the patient's health data. Figure 4.24 highlights the results of our performance tests.

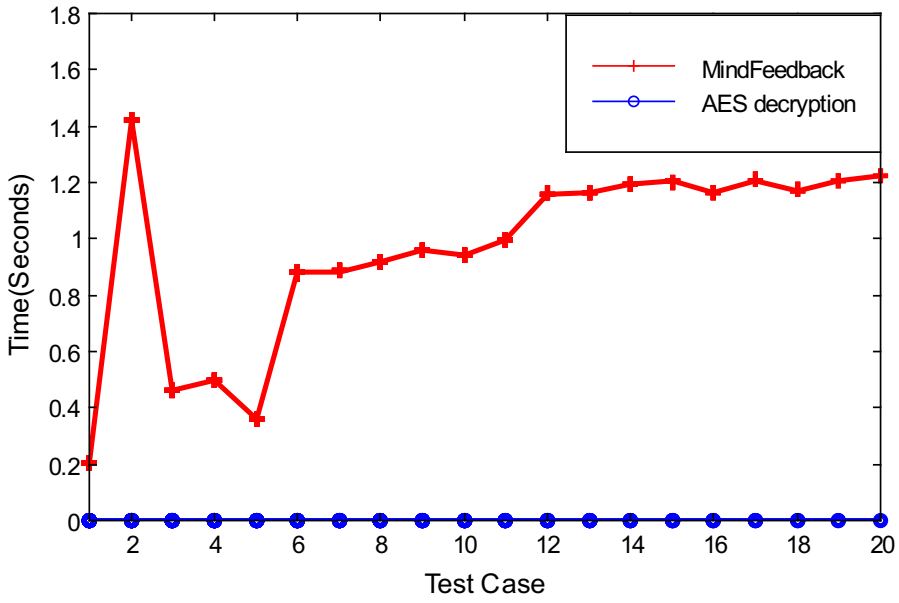


FIGURE 4.24: Download Overhead

As seen in the diagram, the system only had a slight overhead compared to a simple AES decryption operation. The mean time of the AES decryption tests was 0.001 seconds, with a standard deviation of 0.0003 seconds. The mean time of the MindFeedback download tests was 0.961 seconds, with a standard deviation of 0.332 seconds. Note that the patient's encrypted key used to protect health data is first partially decrypted in the Cloud server and then fully decrypted on the patient's smartphone. This is then followed by an AES symmetric decryption using the key on their smartphone, thus accounting for the overhead.

4.4.7 Scalability Analysis

In the scalability tests, we measured the maximum load distribution that our locally-deployed SOAP web service could handle. We used a scalability tool that made calls to the login and getData methods of our Cloud service. The maximum number of threads we were able to run concurrently without the system becoming a bottleneck, was 200.

We carried out the tests on a HP Notebook running Windows 8 with Intel Core i5 and 4GB RAM.

Below is our scalability distribution over the 200 threads, for both calls to login and calls to retrieve the data from the Cloud service.

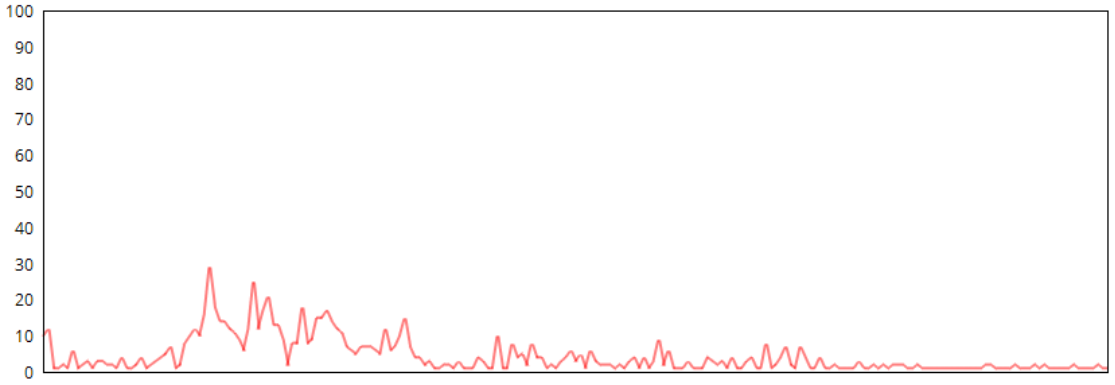


FIGURE 4.25: Distribution of login performance

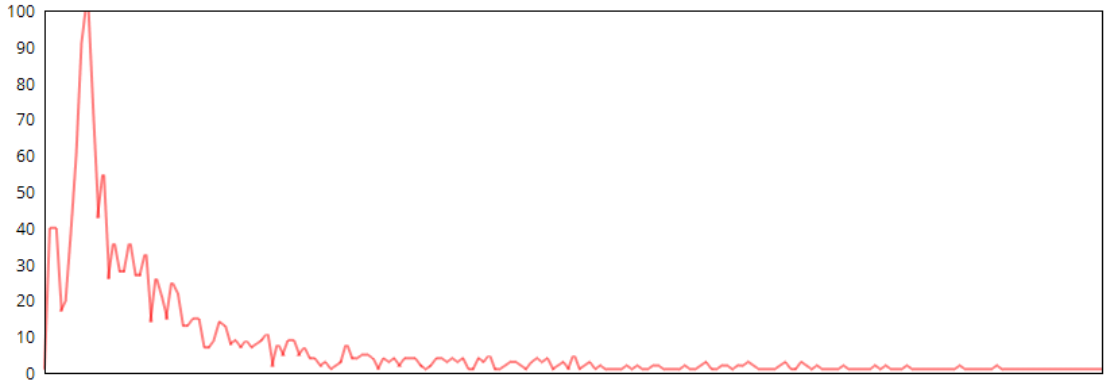


FIGURE 4.26: Distribution of data access from Cloud performance

The diagrams highlight the near-ideal bell curve distribution. Our system could withstand up to 200 concurrent calls to our web service, which makes it more feasible for use in a real-world scenario.

4.5 Summary

In this chapter, we presented a solution to the key-management problem. We first described our key-partitioning technique and then applied our solution to a number of real-world scenarios. Our technique is simple and practical enough to be deployed in a real-world scenario. We showed that our solution does not require the data owner to re-encrypt and re-distribute keys, every time a data consumer is revoked. Through our scenarios, we also showed that doctors and potential users found our system to be private and secure, an important factor when sharing health-related data. Our approach not only reduces the burden on data owners, but also prevents malicious insiders from stealing critical data from the data owner.

Chapter 5

Secure Protection of Outsourced Data

5.1 Introduction

In this section, we provide solutions to prevent security attacks related to data sharing in the Cloud. The following section is based on the work published in [155].

5.2 SafeShare

There is now a growing demand from data owners to have better access control over their data. This includes being able to control how and where their data should be accessed, viewed, modified and distributed. Once the data is outside the perimeter of the data owner's local machine, the data owner will no longer be able to have any control over their data. When a data owner sends a file to a recipient, he cannot guarantee that the recipient will use the data in the way that is expected by the data owner. For instance, a

dishonest recipient can easily redistribute the data to other, unauthorised users via email attachments and USB drives. Furthermore, in Cloud storage, data is usually replicated a number of times and stored in multiple locations around the world, to account for greater availability, which at the same time makes data destruction nearly impossible to guarantee.

In terms of access control, many works focus on using a promising technique called Attribute-Based Encryption (ABE), where users with certain attributes can access the data if those attributes satisfy the access control policy set out by the data owner [102][24][101][100]. However, once the data is decrypted at the user's site, that user can then redistribute the decrypted data to other users, relatively easily. This limitation hinders the convenience of data sharing and collaboration in distributed environments. In our solution, we attempt to make the access control and monitoring as transparent as possible to the user.

5.2.1 The SafeShare System

We now introduce our *SafeShare* architecture and in particular, discuss our secure data sharing model and protocol.

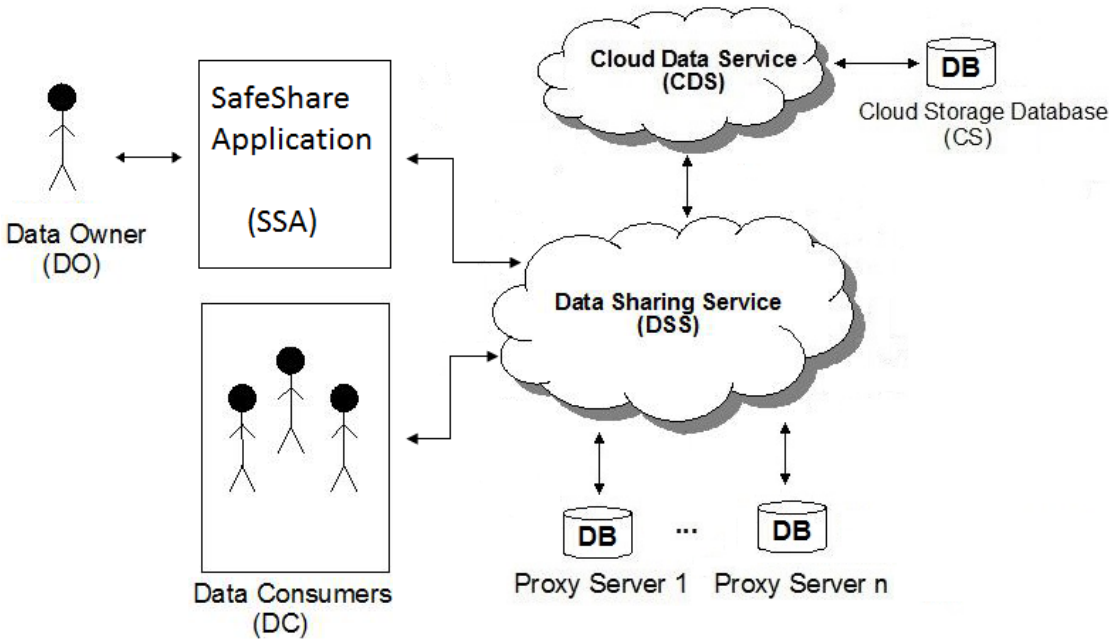


FIGURE 5.1: SafeShare Data Sharing Model

5.2.1.1 Model

Our secure data sharing model is highlighted in Figure 5.1. Table 1 summarises the role of each entity in our data sharing model. We assume that the DSS is honest-but-curious, in the sense that the system will follow the protocols strictly, but is always curious to find out any information about stored data. Note that in our data model, we assume the CDS, DSS, and Proxy Services to be semi-trusted, making our solution attractive for use in a real-world scenario. We also assume the DC to be a semi-honest user, since it will not be possible to completely prevent the DC from redistributing data to other unauthorised users, as the DC can find other avenues, such as taking screenshots, photographs, etc. However, we attempt to make it difficult for the curious DC to redistribute data based on commands from the software stack, such as copy/paste. Hence, the DO can feel comfortable to some degree that in most cases, their data will not be misused by curious DCs. We also note that our work makes use of obfuscation techniques to help prevent the curious DC from sharing the DO's data without the DO's knowledge. Code obfuscation

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
SSA	SafeShare Application	An application the DO runs to generate a SafeShare object and store to Cloud (see model)
DSS	Data Sharing Service	The 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)
CS	Cloud Storage Database	The database containing encrypted data (see model)
α	Symmetric Encryption	Symmetric encryption algorithm
δ	Symmetric Decryption	symmetric decryption algorithm
β	CP-ABE Encryption	CP-ABE encryption algorithm
θ	CP-ABE Decryption	CP-ABE decryption algorithm
γ	El-Gamal Encryption	El-Gamal encryption algorithm
τ	El-Gamal Decryption	El-Gamal decryption algorithm

TABLE 5.1: Abbreviations

is a weak protection mechanism and hence the assumption of a curious DC; however, we attempt to address this in future work.

As the basis of our work, we make use of SCOs as used in the work of Squicciarini et al. [32], and we call this object *SafeShare*. We build upon SCOs and add a mechanism within the object that will prevent users from carrying out operations denied by the DO. Figure 5.2 illustrates our SafeShare object. Each SafeShare object encapsulates encrypted data contents using both symmetric key encryption and CP-ABE. We also make use of ElGamal encryption to enable efficient user revocation, which cannot be achieved by CP-ABE alone. Each SafeShare object also contains an access control policy governing which users can access data, what users can do with the data, and any jurisdiction policies associated with the data. A hash is also kept of the original data for integrity purposes. Finally, a log file is kept, which logs user operations on the data for auditing and accountability purposes. Each SafeShare object also contains operations such as access to data and/or makes copies of the data, if permitted by the data owner.

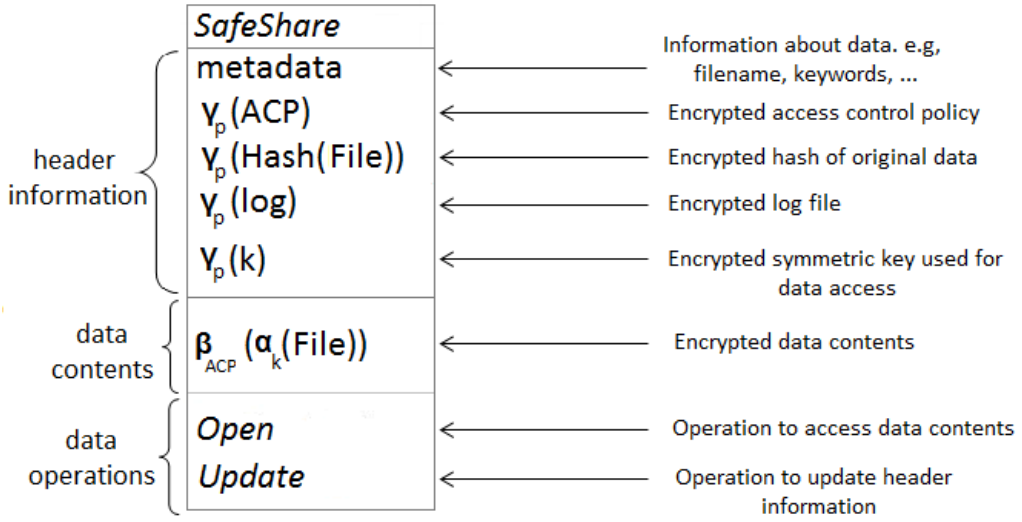


FIGURE 5.2: SafeShare object

5.2.1.2 Overview

We now provide a brief overview of our system. First, the DO stores his data in the Cloud by running the SSA and inputting his data and access permissions. The SSA carries out the encryption of the data and access control policies, and encapsulates them into a SafeShare object along with secret keys, hash value, log, etc. The SafeShare object is then sent to the DSS and consequently the CDS, for storage.

When a DC requests access to the data and the DO approves, the DO calls the SSA to authorise the DC. The SSA calculates new, private key pieces for the DC, which add up to the private key used to decrypt the data. Each proxy server stores a part of the DC's key partition, and the remaining key partition is sent to the DC through a medium of the DO's choosing, such as USB, email, telephone, etc.

Once the SafeShare object is decrypted, the DC calls the *Open()* method to access data, and provides his CP-ABE private key, as well as his supplied key partition. The SafeShare object verifies with the ACP which operations are allowed on the data, and

decrypts the data and keys, if the attributes in the CP-ABE key satisfy the policy set out in the encrypted data.

Once the conditions have been met, the background process will then begin, which monitors for any unauthorised operation (see next section).

When a DO wishes to completely revoke a particular user's access to his data, he simply calls the DSS to remove the DC's corresponding key partitions in the proxy servers.

5.2.1.3 Background monitoring

The background monitoring process aims to prevent a curious DC from carrying out unauthorised operations on the DO's data. Unless specified by the DO in the ACP, the background process does not allow the DC to modify, copy and/or paste the file to a USB or another folder, for sharing.

The background process first creates a temporary folder and stores the decrypted file in the folder. The folder is made available to the DC. When the DC attempts to make a copy of the decrypted file through usual commands, such as Ctrl-C on Windows, the background process checks with the ACP whether the operation is allowed, and if not, immediately deletes the file in the temporary folder, leaving the clipboard empty. Similarly, when the DC is not allowed to make modifications to his file, the background process continually checks whether the file is modified. If the file is modified, the new file is immediately deleted and the user would need to run the SafeShare object again, to retrieve the original file. A log file is also kept if the DO requires additional control and is periodically flushed to the Cloud, to prevent large data sizes on the DC's machine. The log file contains information such as the DC id, IP address and the data operation, such as READ or WRITE. The DO can then retrieve the log file from the Cloud,

for auditing and/or accountability purposes. The log file can be sent daily, weekly or monthly depending on what the DO requires.

While the background monitoring helps to reinforce stronger access control of the data, it is not very difficult to circumvent the process. For example, the DC could read the file using a number of methods at the system level, command level and even in backup. Our background monitoring process only controls unauthorised operations only from the UI and clipboard and thus is more suitable for the everyday user. We acknowledge that this is a limitation of our work and we will build upon this limitation in Section 5.3.

5.2.1.4 Protocol

We now discuss our SafeShare protocol in detail. The protocol has five stages: *data storage*, *data retrieval*, *consumer authorisation*, *authorised data access*, and *consumer revocation*.

Data Storage

①	DO → SSA	INPUT: $FILE$, P, PK
②		Generate {p,b,c}, x
③		Hash($FILE$)
④		Generate log and key k
⑤		$\alpha_{\mathrm{k}}(FILE) = m$
⑥		$\beta_p(m) = M$
⑦		$\gamma_{\{\mathrm{p,b,c}\}}(P) = ACP$
⑧		$\gamma_{\{\mathrm{p,b,c}\}}(Hash(FILE)) = H$
⑨		$\gamma_{\{\mathrm{p,b,c}\}}(log) = L$
⑩		$\gamma_{\{\mathrm{p,b,c}\}}(k) = K$
⑪		Create SafeShare object SS
⑫	SSA → SS	$ACP, H, L, K, M, x_{\mathrm{n+2}}$
⑬		Obfuscate SS file
⑭	DO → DSS	credentials, SS
⑮	DSS → CDS	verifyCredentials(credentials)
⑯	CDS → DSS	u_{DO}
⑰	DSS	Generate d_{FILE}
⑱	DSS → CDS → CS	$u_{\mathrm{DO}}, d_{\mathrm{FILE}}, SS$

At the data storage stage, the DO first inputs his file, policies and public parameters from ABE to SSA on his PC ①. The application will generate public and private key pairs from the initialisation stage of the ElGamal encryption algorithm ②. A hash of the file will then be calculated ③. An empty log file will also be generated, as well as a symmetric key ④. The file will first be encrypted by symmetric key k ⑤ and then, using the access control policy, by the CP-ABE public key ⑥. The access control

policy, data hash, log file and symmetric key will also be individually encrypted by the ElGamal public key ⑦ - ⑩. The application will then generate and store the encrypted file along with the encrypted access control policy, data hash, log file and symmetric key, in an object file that we call SafeShare ⑪ - ⑫. Also, one partitioned key piece will also be stored in a variable in the source code of the SafeShare object (12), and will then be obfuscated ⑬, so that it will be extremely difficult to reverse engineer the code to find out the key piece value. This can only be found out through the running of the executable. The DO then sends the SafeShare object, along with its credentials, to the DSS ⑭.The DO keeps the secret key x on his machine. The DSS verifies the user credentials ⑮ - ⑯, and then generates a data identification ⑰. The SafeShare object is then sent to Cloud storage ⑱. Note that we assume that the DO exists in the database with a user identification. If the DO does not exist in the database, intuitively, the data will not be stored.

Consumer Authorisation

①	DO	Calculate $x - x_{\text{n+2}}$
②		Generate $x_{\text{u1}} + ... + x_{\text{u(n)}} + x_{\text{u(n+1)}} = x - x_{\text{n+2}}$
③	DO → DSS	auth(credentials,$email, d_{\text{FILE}}$, $\{x_{\text{u1}}, ...x_{\text{u(n)}}\}$)
④	DSS → CDS	verifyCredentials(credentials)
⑤	CDS → DSS	u_{DO}
⑥	DSS → CDS	verifyUserExists(email)
⑦	CDS → DSS	u_{DC}
⑧	for (all proxy i)	
	DSS → proxy i	$\{u_{\text{DC}}, u_{\text{DO}}, d_{\text{FILE}}, x_{\text{u(i)}}\}$
⑨	DO	Generate attribute set and corresponding private key pk_{DC}
⑩	DO → DC	$pk_{\text{DC}}, x_{\text{u(n+1)}}, \{p, b, c\}$

The DO first calculates the secret key value x minus the key partition value stored in the SafeShare object ①. The DO partitions this value into $n + 1$ random pieces, where n represents the number of proxy servers ②. The DO sends his credentials, the DC's identifier (i.e., email), the data identification and n key partitions, to the DSS ③. The DSS, after verifying whether the DO and DC exist ④ - ⑦, then stores the DC's key partitions to each of the proxy servers ⑧. Finally, the DO generates a private key using the CP-ABE KeyGen algorithm, to generate a key for the DC that provides access to the data ⑨. The CP-ABE key, along with the remaining key partition and public key, is sent to the DC, and he now gains access rights ⑩.

Data Retrieval

①	DC → DSS	credentials, d_{FILE}
②	DSS → CDS	verifyCredentials(credentials)
③	CDS → DSS	u_{DO}
④	DSS → CDS	$u_{\text{DO}}, d_{\text{FILE}}$
⑤	CDS → DSS	SS
⑥	for (all proxy i)	
	DSS → proxy i	$u_{\text{DC}}, d_{\text{FILE}}$
	proxy i → DSS	$x_{\text{u(i)}}$
⑦	DSS → SS	update($x_{\text{u(i)}}$)
⑧	SS	$\tau_{x_{\text{u1}}}(ACP)$
		$= \tau_{x_{\text{u1}}}(\gamma_{\{\text{p,b,c}\}}(P))$
		$= (c^r, (c^r)^{-x_{\text{u1}}}.c^{\text{rx}}.P \ mod \ p)$
		$= (c^r, (c^r)^{x-x_{\text{u1}}}.P \ mod \ p)$
		Similarly for H, L and K
⑨	Repeat step ⑦ for	RP = Remaining ACP cipher:
	all n key pieces	$(c^r, (c^r)^{x-x_{1\text{u}}-\ldots-x_{\text{un}}}.P \ mod \ p)$
		RH = Remaining cipher of H:
		$(c^r, (c^r)^{x-x_{1\text{u}}-\ldots-x_{\text{un}}}$
		$.Hash(FILE) \ mod \ p)$
		RL = Remaining cipher of L:
		$(c^r, (c^r)^{x-x_{1\text{u}}-\ldots-x_{\text{nu}}}.log \ mod \ p)$
		RK = Remaining cipher of key:
		$(c^r, (c^r)^{x-x_{1\text{u}}-\ldots-x_{\text{nu}}}.k \ mod \ p)$
⑩	DSS → DC	SS

At this stage, the DC (or DO) downloads the SafeShare object from the Cloud, ready to be accessed. The DC sends his credentials and data identification to the DSS ①. The DSS verifies whether the user is legitimate ② - ③ and if so, calls the CDS to retrieve the SafeShare object ④ - ⑤. The DSS will also retrieve the DC's key partitions from the proxy servers ⑥. The DSS then calls the *update()* method of the SafeShare object and also sends the key partition value ⑦ - ⑧. The object then uses the key partition to decrypt the access policy, data hash, log file and symmetric key. The object then updates itself with these new values ⑨. The SafeShare object, containing the partially decrypted contents, are then sent to the DC ⑩.

Authorised Data Access

①	DC $\rightarrow SS$	Open($x_{\mathrm{n}+1}, pk_{\mathrm{DC}}$)
②	SS	$\tau_{x_{\mathrm{u}(\mathrm{n}+2)}}(\tau_{x_{\mathrm{u}(\mathrm{n}+1)}}(RP))$
		$= (c^r, (c^r)^{x-x_{1\mathrm{u}}-\ldots-x_{\mathrm{u}(\mathrm{n}+2)}}.P \ mod \ p)$
		$= (c^r, (c^r)^{x-x}.P \ mod \ p)$
		$= (c^r, P \ mod \ p)$
③		Similarly, repeat step 2 for RH, RL and RK:
		P, Hash($FILE$), log, k
④		$\theta_{pk_{\mathrm{DC}}}(\beta_P(m)) = m = \alpha_k(FILE)$
		$\delta_k(m) = \delta_k(\alpha_k(FILE)) = FILE$
⑤		Create monitoring folder F
⑥	$SS \rightarrow F$	$FILE$
⑦		monitorInBackground()
⑧		IF (operation violates ACP)
		Delete $FILE$ immediately
⑨		IF (ACP contains doLog=true)
		log each operation to log file.
		Periodically upload log to DSS and clear log in SS

At this stage, the DC accesses the data encapsulated within the SafeShare object. The DC simply runs the *Open* function of the object, passing on his stored key piece and CP-ABE private key ①. The SafeShare object then decrypts each of the metadata contents using the DC's key piece and then later, the key piece stored within the source code, to reveal the full metadata as well as the fully decrypted symmetric key ② - ③. Note that this key value is stored in the executable binary as it is running, hence it is extremely difficult for the DC to ever find out this key value. The SafeShare object will use the

fully decrypted symmetric key to decrypt the file. If the attributes of the private key satisfy the ACP, then the file will be decrypted to reveal the data encrypted by k. The data is then decrypted fully by k ④. The SafeShare object, after checking whether the user is authorised to access the data, will generate a temporary folder and store the file in that folder ⑤ - ⑥. The SafeShare object will then monitor the file and temporary folder in the background, simulating a watchdog ⑦. If an operation violates the ACP, such as no copying or no modification, the file will be deleted immediately ⑧. If the DO explicitly states in the ACP that logging should be enabled, the watchdog will log any operations to the log file and update the object with the latest log. The SafeShare object will periodically flush the log file to the DSS, to ensure that the file sizes do not exceed a maximum range ⑨.

Consumer Revocation

①	DO → DSS	deleteUser(credentials, u_{DC}, d_{FILE})
②	DSS → CDS	verifyCredentials(credentials)
③	CDS → DSS	u_{DO}
④	for (all proxy i)	
	DSS → proxy i	removeKeyPiece(u_{DO}, u_{DC}, d_{FILE})
	proxy i → DSS	Remove x_{ui}

When the DO decides to revoke a user's access rights to data, he simply calls the DSS to request that the user's rights to the data are revoked ①. The DSS will then verify the credentials of the user ② - ③ and then, provided that the user exists, remove the corresponding key pieces of the user in each of the proxy databases ④. Note that the data does not need to be re-encrypted, and none of the other users will be affected, since only the key pieces corresponding to the consumer are removed. All other key pieces

corresponding to other consumers still remain in the proxy database. Since the data does not need to be re-encrypted, nor does there need to be any key re-distribution, the model is efficient and has a runtime of O(n), where n is the number of proxy servers.

5.2.2 Security Analysis

We now analyse our model and protocol from a privacy and security perspective.

1. *Data confidentiality* – Data remains encrypted at all times, whether it is in transit, within the Cloud provider, or on the DC's machine. The only time the data is decrypted is when the *Open()* method is called, since the class file contains the remaining key partition. Since the SafeShare object is obfuscated, the value of the key partition will be hard to reverse engineer. Without the key partition value, even if the attacking user has all of the other partitions, he still does not possess knowledge of the full ElGamal private key and cannot decrypt the data. Since key partitions are stored in different proxy servers (possibly modelled and implemented on different CSPs), unauthorised data access becomes extremely difficult. This is due to the fact that compromising all CSPs is almost impossible [156].

2. *Unauthorised redistribution* – The symmetric key is also encrypted with the ElGamal private key, and also requires the key partition value in the obfuscated source code to retrieve the full CP-ABE key needed to decrypt data. When the object method is run, and the user has fulfilled all other requirements needed for the object, the object will decrypt the data itself and then monitor operations on the decrypted data in a background process, as a watchdog. If the ACP denies redistribution, the watchdog code will prevent the DC from copying the decrypted data to another folder; for instance, sending an email attachment. This can also

help prevent attacks related to covert channels. The watchdog is non-intrusive upon user behaviour and only monitors actions on the relevant data owner's files, to check for any unuathorised operation. It does not interfere, nor log the user actions on other files and applications.

3. *User revocation* – User revocation involves simply removing a DC's key partitions from the proxy. By doing so, the DC can never recover the full ElGamal private key. The key partition stored on the DC's machine will be rendered useless without the other key partitions. The DC also never knows the full value of the private ElGamal key, unless it is leaked by the DO. Without the full key, it is nearly impossible to decrypt the CP-ABE key needed to decrypt the data. The attacker would have to guess the final key partition in order to fully decrypt the CP-ABE key. Similarly, without the full key, it is also extremely difficult to decrypt the metadata information such as the ACP, hash value and log file.

4. *Auditing/Accountability* – If the DO has highly confidential data, for extra security, he can explicitly state that all operations on the data are logged. All operations on the data will be noted by the watchdog and appended to the log file. The log file will be periodically sent to the DSS web service, which the DO can access at any time, from anywhere, to keep track of the usage of his data and to satisfy his auditing/accountability requirements.

5. *Data Integrity* – While the watchdog is running in the background, the updated hash value can also be periodically sent to the DSS. The DSS will update the previous hash value to the current one in the database. The DO can later check whether his data has been tampered with, and can also hold accountable whoever tampered with the data.

5.2.3 Implementation and Evaluation

We now provide the implementation details of our system, followed by experimental results and an evaluation.

5.2.3.1 Implementation

We developed our prototype of the system using the Java programming language. The DSS, CDS and Proxy servers were developed using Java, Apache Tomcat and Apache Axis2. We also used MySQL for data storage. The CP-ABE scheme, developed by Wang [157], implements the work of Bethencourt et al. [105]. The code was developed using the Java Pairing-Based Cryptography Library (jPBC library) [158]. To implement the SafeShare object, we made use of executable JAR files. The SSA was implemented using Java. ProGuard [159] was used to obfuscate and shrink the JAR file.

5.2.3.2 Experimental Results

We carried out a number of performance tests on our system. In particular, we measured the overhead introduced in our SafeShare object, in comparison to sharing data using only the CP-ABE scheme. We measured the performance of generating and opening SafeShare objects. The purpose of these tests was to determine whether our system would be feasible for use in a real-world scenario. To carry out the tests, we used a dual-core ASUS laptop with 2GB memory, 350GB storage and Windows Vista Operating System, as well as a Dell XP 8500 PC with Intel Core i7 and 8GB memory, running Windows 8.

For each of the tests, we used a number of files with sizes ranging from 1KB to 100MB, to reflect different application requirements. The files used in the tests ranged from simple text files, to documents, and also different image formats. For larger files, we used video files with different formats and 3D graphics files. For each file, the performance tests were run a number of times. The average time was then calculated, and displayed in the figures below. Using the same set of files, we also measured performance for the CP-ABE scheme of Bethencourt et al. [105]. This allowed us to measure the performance overhead of our SafeShare object.

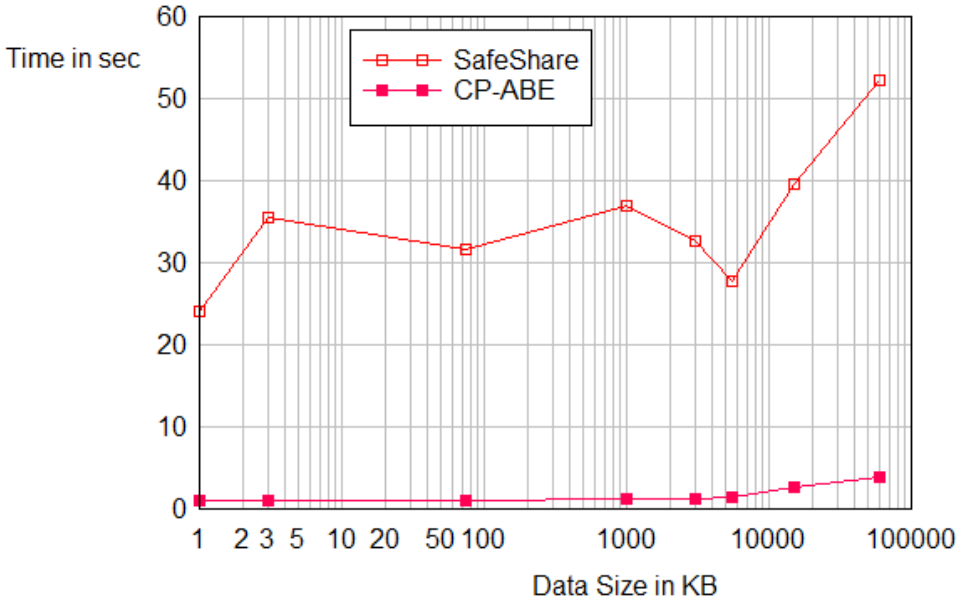


FIGURE 5.3: Data encryption overhead

Figure 5.3 highlights the results of our first tests. From the figure, it is clear that SafeShare objects took much longer, compared to a simple CP-ABE encryption scheme. This is due to the SafeShare object having to encrypt the access control policy, hash and log file, as well as the data itself, and then package all these contents into a JAR file. Most of the time was spent in the generation and packaging of contents to a JAR file. As file sizes increased, the data generation times increased considerably for SafeShare objects, with the largest file size taking 50 seconds, while the CP-ABE scheme only

demonstrated a small increase in time. The slight dip in the overhead of files around the 10MB mark, is accounted for by testing carried out in different times; it is nevertheless still similar to all of the file sizes below 10MB.

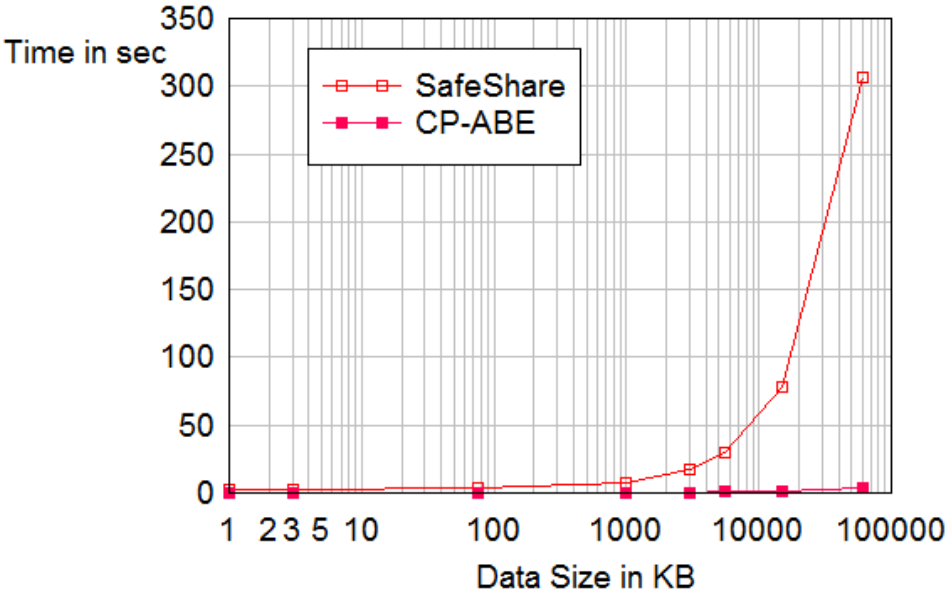


FIGURE 5.4: Data access overhead

The overhead introduced when a user carried out data access was also measured, using the same set of files. Figure 5.4 illustrates the results of our tests where again, we measured our SafeShare scheme with the CP-ABE scheme. This is the result of running the *Open* operation of the SafeShare object. We found that for files up to 10MB in size, the overhead in access times was nearly negligible. However, once the file sizes became larger, the access times increased exponentially, leaving a large overhead compared to the CP-ABE scheme. For files with sizes of 60MB, the user would have to wait approximately five minutes each time, to access data. The access times for the CP-ABE scheme remained minimal, no matter how large the file was.

5.2.3.3 Evaluation

From the performance tests, we found that our SafeShare object was comparably slower to encrypt and decrypt. Regarding the encryption, this is mainly to do with the generation of JAR files. Data access times increase exponentially for larger data sizes; however, as processing power increases and the generation of JAR files optimises, access times will improve in the near future. Also, in the near future, it may be possible to split the process and run them in parallel threads to significantly improve performance times. In our current solution, users may be willing to wait a little longer for highly confidential data to generate and access data. For instance, a business user might be willing to wait for a little more than a minute to access 15MB of highly confidential paperwork. This makes our solution highly attractive to users who are more concerned with privacy rather than performance times.

5.3 Hardware data encapsulation using TED

We now improve on the software-based SafeShare and leverage a hardware device called the TED, for the stronger privacy and security of data. Hardware-based mechanisms are generally more difficult to attack compared to software-based mechanisms, as this would require some form of physical effort. The work in this section is based on [160].

5.3.1 Introduction

The Bring Your Own Device (BYOD) [161] concept is a growing trend that allows enterprise employees to bring their own devices, such as laptops, smartphones, and tablets, to their workplace, and use the devices to access sensitive data within the

organisation. Key management and key stealing issues associated with the Cloud can be addressed using BYOD, since the employee device contains one key that is unique to the device and uses that key for secure sharing. However, when applying BYOD to data sharing in the Cloud, a number of issues are introduced [161][162]. If the employee's device contains malware, it could inject the malware into enterprise servers. Also, it is possible that the employee's device could leave traces of very sensitive enterprise data, later on.

The main vulnerabilities of introducing BYOD to data sharing in the Cloud include:

- *Key management* - Storing keys and certificates inside the device is complex and unsafe. Since there is no control over the consumer's device, it is not guaranteed that the keys stored inside the device will never be revealed to outsiders.
- *Data protection* - Once the consumer obtains the data, there is no guarantee that the consumer will then redistribute the data to other unauthorised consumers, such as friends or family.

In our work, we focus primarily on the former, and leave the latter as an aspect of future work. We make use of a small, self-contained USB-sized computer which incorporates TPM, called the Trust Extension Device (TED) [163], in order to address the key management limitations of BYOD. The attractive part of our solution is that the consumer's untrusted computer can simply be viewed as IO, while the secure data-sharing operations are carried out inside the TED. From the consumer's perspective, the TED is viewed as a black box. Although BYOD is currently a more cost-effective approach compared to TED, we foresee that in the future, most smartphones, tablets and notebooks will have similar capabilities to the TED. Our system is heterogeneous, in that it can be used under any operating system, platform, etc.

DO	The owner of the data; decides who has access permission to the data
DC	Any consumer who has permission to access data provided by the DO
DSS	The untrusted data sharing Cloud service used to store and retrieve data (see model)
TTI	The Trusted TED Issuer service that manufactures TED devices and distributes them to DCs, offline
PCA	The trusted service that verifies whether a DC is who they say they are
TED	Physically-based TPM model used to attest and retrieve encrypted data from the DSS

Table 5.2: Entities

5.3.2 Security Model and Protocol

We now introduce our system architecture and in particular, discuss our secure data sharing model and protocol. Our model and protocol is not limited to a specific application, but can be applied to a variety of different application scenarios. Our system is heterogeneous and is not limited in terms of operating system, platform, etc. Our system is also transparent, in the sense that the data consumer can use it as an "out-of-the-box" tool, without having to run any configuration. Note that all operations carried out by the TED are completely transparent to the host OS. The host OS will never be able to retrieve any information from the TED.

5.3.2.1 Model

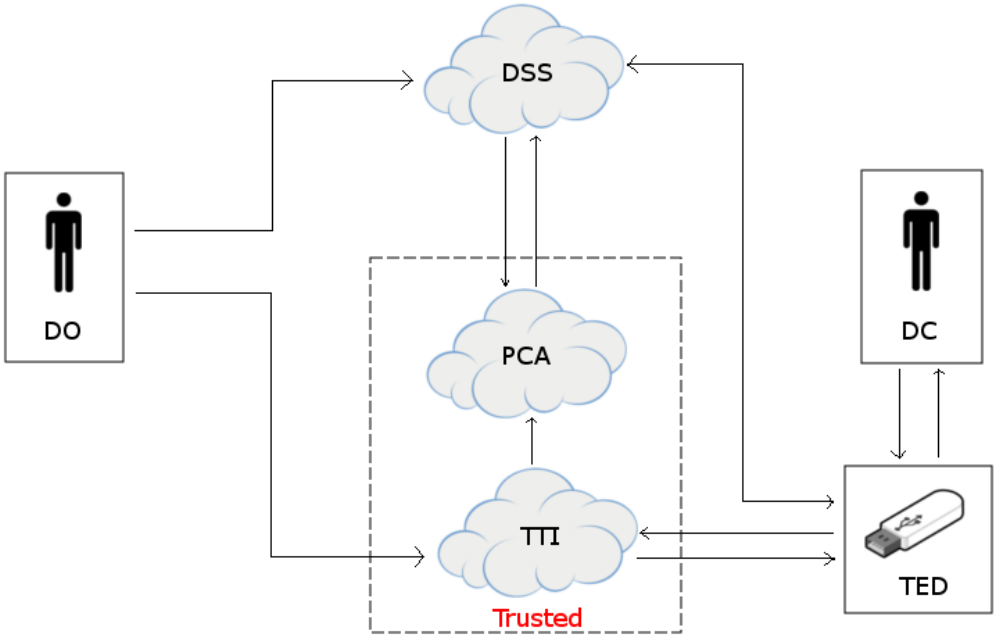


FIGURE 5.5: Data Model

Our secure data sharing model is highlighted in Figure 5.5. The Data Sharing Service (DSS) is an untrusted Cloud Service Provider used to store the Data Owner's (DO) data. The Trusted Ted Issuer (TTI) manufactures TED devices with the embedded cryptographic keys that are needed to decrypt data for Data Consumers (DC), provided that DCs have bought/registered the DO's data. A Private Certificate Authority (PCA) is called by the TED device, whenever the DC requires data access. The PCA verifies that the DC really is who they say they are, as well as registered to access the data. If successful, the PCA grants data access. A number of DCs will use the TED to securely connect with the TTI and PCA, to be authenticated and verified, and if successful, retrieve the data. Zic and Hardjono [164] propose a similar model where they leverage the use of TED to measure the integrity of client devices when they access Cloud-based applications and services. Whereas, in our work, we leverage TED to ensure that the

authorised DC is able to access the data without ever knowing the key used to decrypt the data.

We assume that the DSS is honest-but-curious, in the sense that the system will follow the protocols strictly but is always curious to find out any information about stored data. Note that in our data model, we assume the PCA and TTI to be trusted. We also assume the TED to be trusted. We assume the DC to be a semi-honest consumer, since it will not be possible to completely prevent the DC from redistributing data to other unauthorised consumers, as the DC can find other avenues, such as taking screenshots, photographs, etc. However, we attempt to make it difficult for the curious DC to redistribute data. Hence, the DO can feel comfortable to some degree that in most cases, their data will not be misused by curious DCs.

5.3.2.2 Protocol

We now discuss our secure data sharing protocol in detail. The protocol has four stages: *Data Storage and Set Up*, *Data Consumer Registration*, *Data Access*, and *Consumer Revocation*.

Data Storage and Setup

①	DO	$x = x_1 + x_2$
②		$E_k(data)$
③		$E_{(p,b,c)}(k)$
④		$D_{x1}(E_{(p,b,c)}(k))$
⑤	DO $\rightarrow$ DSS	$\{E_k(data), D_{x1}(E_{(p,b,c)}(k))\}$
⑥	DO $\rightarrow$ TTI	x_2

Data Storage and Set Up: First, the DO generates an ElGamal public and private keypair and then partitions the private key into two parts ①. The DO then generates a random symmetric key and encrypts data with it ②. The symmetric key is then encrypted with the public ElGamal key ③, and is then partially decrypted by the first half of the key partition ④. The encrypted data and encrypted key are then sent to the DSS for storage ⑤. The DO also sends the second key partition to the TTI, to enable data sharing ⑥. It is assumed that the DO sends the key partition value through a secure medium, as decided upon by the enterprise, such as email, over the telephone, by mail, or in person, to name a few.

Data Consumer Registration

①	DC → TTI	$register(credentials, d_{id})$
②	TTI	$x_2 = x_a + x_b$
③	TTI → PCA	$\{credentials, d_{id}, x_a\}$
④	TTI → TED	$\{credentials, EC, x_b, ...\}$

Data Consumer Registration: The DC registers with the TTI using his credentials, in order to gain data access ①. We do not detail the registration process, since it is outside the scope of this work. We assume that registration can simply involve, for example, the DC purchasing the data item through a website. The website is also assumed to have knowledge of the DC, via credential information. The TTI partitions the DO's supplied key piece, again, into two parts ②. One of the two parts is generated randomly; therefore, both pieces will always be different for each consumer. The TTI then sends the DC credentials and one of the newly generated key partitions, to the PCA for storage ③. The TTI will then manufacture, offline, a TED device embedding

Data Access		
①	DC → TED	$Open()$
②	TED	pub_{AIK}, prv_{AIK}
③	TED → DSS	$\{credentials, pub_{AIK}, sign(EC, prv_{TTI})\}$
④	DSS → PCA	$\{credentials, pub_{AIK}, sign(EC, prv_{TTI})\}$
⑤	PCA	$verify(sign(EC, prv_{TTI}), pub_{TTI}) =$
		$EC = \{pub_{EK}, d_{id}, TPMmodel, ...\}$
⑥		$verifyCredentials(credentials)$
⑦		$sign(pub_{AIK}, prv_{PCA})$
⑧	PCA → DSS	$\{d_{id}, x_a, sign(pub_{AIK}, prv_{PCA})\}$
⑨	DSS	$D_{xa}(D_{x1}(E_{(\mathrm{p,b,c})}(k)))$
		$D_{x1+xa}(E_{(\mathrm{p,b,c})}(k))$
⑩	DSS → TED	$aenc(\{D_{x1+xa}(E_{(\mathrm{p,b,c})}(k)), E_k(data),$
		$sign(pub_{AIK}, prv_{PCA})\}, pub_{AIK})$
⑪	TED	$adec(aenc(\{D_{x1+xa}(E_{(\mathrm{p,b,c})}(k)), E_k(data),$
		$sign(pub_{AIK}, prv_{PCA})\}, pub_{AIK}), prv_{AIK})$
⑫	TED	$verify(sign(pub_{AIK}, prv_{PCA}), pub_{PCA})$
⑬	TED	$D_{xb}(D_{x1+xa}(E_{(\mathrm{p,b,c})}(k)))$
		$= D_{x1+xa+xb}(E_{(\mathrm{p,b,c})}(k))$
		$= D_x(E_{(\mathrm{p,b,c})}(k))$
		$= k$
⑭	TED	$D_k(E_k(data))$
		$= data$
⑮	TED → DC	$data$

the credentials, endorsement credentials containing the endorsement keys, data identification, TPM model, manufacturer, etc., as well as the second key partition generated by the TTI ④. The TED device will then be shipped to the DC's physical address.

Data Access: The data consumer runs the secure application in the TED device, to access data ①. The TED device carries out the attestation protocol (see section 3.2). First, the attestation key pair is generated ②. TED then sends its credentials, public attestation key, and signed endorsement credentials to the PCA via DSS ③, ④. The PCA verifies the signature to retrieve the endorsement credentials, and hence knows which TED device made the call ⑤. The PCA also verifies the consumer's credentials to ensure that the correct consumer is using the TED device ⑥. The PCA then signs the public attestation key to reveal the attestation identity certificate ⑦. The PCA retrieves the data identification and key partition from the consumer's database entry, and sends this along with the signature, to the DSS ⑧. The DSS uses the key partition to decrypt the partially decrypted key, to result in a further partially decrypted key ⑨.

The DSS encrypts the encrypted data, partially decrypted key and attestation identity certificate with the public attestation identity key, and sends this to the TED device ⑩. The TED application decrypts this using the private attestation identity key stored in the TED device ⑪. The application verifies the signature of the attestation certificate ⑫ and if verified, it uses the key partition stored in the TED storage to decrypt the partially decrypted key, in order to reveal the fully decrypted symmetric key ⑬. The symmetric key is then used to decrypt the data ⑭. Finally, the data is sent to the DC through the secure application ⑮.

Consumer Revocation

①	DO → TTI	$revokeConsumer(credentials, d_{id})$
②	TTI → PCA	$remove(credentials, d_{id})$
③	PCA	$delete(credentials, d_{id})$

Consumer Revocation:

When the DO wants to revoke a particular DC, the DO simply calls the revoke operation, sending a consumer credential such as the DC_id or username to the TTI ①. The TTI passes the credentials to the PCA, ordering the removal of the consumer ②. The PCA locates the consumer credentials in the database and deletes the entire row, including the key partition, hence rendering the data completely useless to the DC ③.

5.3.2.3 Security Evaluation

- *User Revocation* - In our protocol, user revocation can be achieved efficiently without the need to re-encrypt the data each time. The DC's key partition, as well as credential information associated with the data, is simply removed from the trusted PCA's database. This way, if the revoked DC attempts to access the

data, their corresponding TED will fail to be attested and, furthermore, will never be able to be decrypted without the key partition.

- *Collusion between DC and DSS* - In the event that a dishonest DC and the untrusted DSS collude, the DC will only be able to retrieve the encrypted data and the partially decrypted symmetric key. Without the remaining key partition, it is extremely difficult for the DC to decrypt the data, as the data decryption key will still remain illegible. Since the remaining key partition is encapsulated within the TED device, the DC would have to carry out hardware-based attacks in order to retrieve the remaining key partition. Hardware-based security mechanisms are much more difficult to attack; therefore, our protocol handles collusion attacks between DC and DSS.

- *Man-in-the-Middle Attacks* - Our protocol handles man-in-the-middle attacks, since every time a DC requests access to data, the TED application always attests with the DSS and PCA, using the attestation identity key. The encrypted data and keys are returned to the TED device with a signature from the PCA attesting that the TED device is legal and valid.

- *Update Secrecy* - When the DO wishes to replace his current stored data with a newer, updated version, he simply encrypts the data with the same symmetric key and sends it to the DSS to be replaced. Note that nothing else in the protocol needs to be changed, thus making our solution practical for deployment in a real-world scenario.

- *Losing TED* - In the event that a DC's TED device is physically lost or stolen, the DC immediately notifies the TTI (using his credentials) that the device has been

stolen. The TTI verifies the credentials, and notifies the PCA to remove the entry containing the DC's credentials and key partition.

- *Insider Attacks* - Our protocol is also secure against insider attacks, since there is never a stage in our protocol where the data is decrypted in the Cloud. The data remains encrypted at all times on the untrusted Cloud servers, as well as on untrusted public communication channels.

- *Unauthorised Data Redistribution* - In our protocol, we also attempt to prevent the unauthorised redistribution of data by friends, family, etc. It is impossible to completely prevent a dishonest DC from redistributing the data, since the TED eventually returns the full plaintext data. The DC can still redistribute the data using the plaintext. However, it is now more difficult to let an arbitrary number of unauthorised users from accessing and using the data. Also, the DC can physically hand over their TED to another user, which is outside the scope of this work. We will attempt to address this in future work.

5.3.3 Implementation and Evaluation

We now provide the implementation details of our system, followed by experimental results and an evaluation.

5.3.3.1 Implementation

We implemented our TED application using the C language. To run the TED application, we connected the TED device to a PC, and then created a simple Java application that is used to invoke and return commands to and from the TED device.

5.3.3.2 Evaluation

We measured the overhead introduced by the encryption and decryption of data using TED. The purpose of these tests was to determine whether TED would be feasible for use in a real-world scenario, with regards to secure data sharing. The host machine we used was an Intel Core i5 HP Pavilion Notebook with 4GB memory, 1TB storage, and 64-bit Windows 8 operating system.

To carry out the performance test, we first measured the time taken to encrypt a simple text file using AES encryption. We then measured the time taken to encrypt the text file using our simple Java application named "SecureShare," which follows the protocol. We carried out the test cases a number of times and calculated the average times. Using the encrypted text file, we then measured the time taken to decrypt the file using AES decryption, and the time taken to decrypt the file using TED.

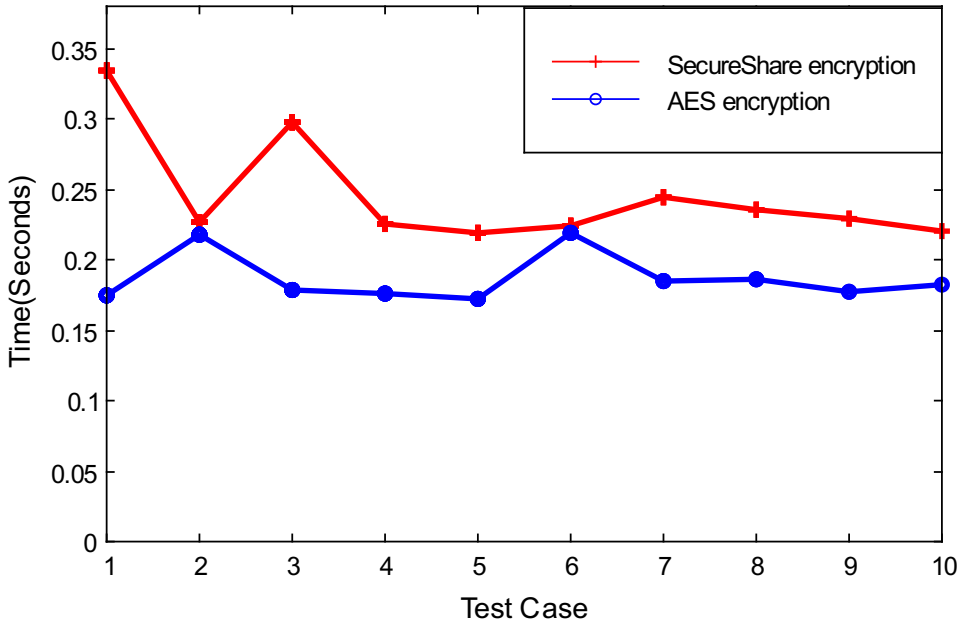


FIGURE 5.6: Encryption overhead

Figure 5.6 highlights the results of our first tests. From the tests, the overhead introduced

by our Java application, "SecureShare," is slightly higher than simple AES encryption. The average time taken to encrypt the text file using a simple AES encryption algorithm was approximately 0.19 seconds. The average time to encrypt the same text file using SafeShare was approximately 0.24 seconds. However, since the encryption process is under half a second, everyday users will find it feasible to use our SafeShare application to protect their files.

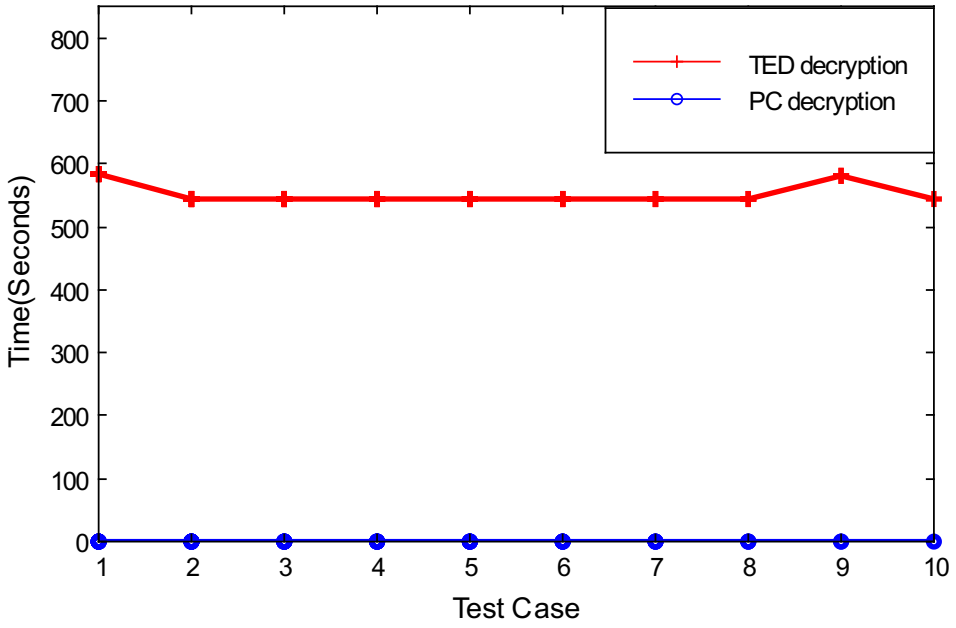


FIGURE 5.7: Decryption overhead

Figure 5.7 illustrates the overhead introduced by decrypting data using the TED device. From the figure, it is clear that the overhead of decrypting data using the TED is significantly higher than decryption using a simple AES algorithm. The average time to decrypt the text file using AES decryption was approximately 0.27 seconds, while the average time to decrypt the same text file using the TED, was 551.41 seconds.

5.3.3.3 Performance and Size Limitations

The poor performance of the decryption is mainly due to the limited resource and capability of the existing TED that was prototyped a few years ago, instead of our approach itself. For example, the data size that the TED currently supports is approximately 2-3KB. Simply adding larger CPU and memory to the TED devices will see drastic improvements in performance and size limitations. We foresee that in the near future, devices such as the TED will have greater capabilities, which will in turn drastically improve performance times. Furthermore, we envisage such capabilities to be incorporated into smartphones and notebooks, which will further enhance the usability of our system.

5.4 Summary

In this chapter, we first designed and implemented SafeShare, a software object that encapsulates both data and policy and allows private and secure sharing without leaking the data to the Cloud. SafeShare also incorporates a background monitoring mechanism that aims to prevent dishonest consumers from leaking data to other, unauthorised users. We then improved on SafeShare, and leveraged hardware data encapsulation via the TED. This was shown to provide stronger privacy and security compared to SafeShare, due to data being decrypted within the hardware device and not the host OS.

Chapter 6

SelfProtect Object

Our previous solutions have focused on providing private and secure data sharing, by addressing key management and security attacks. However, we assumed that the data consumer is honest and will not leak the data to unauthorised outsiders. Once the data is given away to the Cloud and/or data consumers, the data owner effectively loses full control over their own data. The data consumer can copy, modify and distribute the data to friends and family, without the knowledge of the data owner. Thus, there is now a strong demand for data owners to regain some degree of control over their own shared data, no matter where the data is stored and accessed. In this chapter, we present technologies that we have developed to allow the data owner to regain some level of control over their data. This chapter is based on the work published in [165].

6.1 Introduction

The amount of content being generated and shared is increasing at a rapid pace [1][12][166]. The types of content include photos, movies, music, eBooks, business documents, health

data, etc. While content sharing provides many benefits, content owners lose almost full control of their content once it is given away to consumers. Currently, Digital Rights Management (DRM) is used to give the content owner some form of control [167]. The unauthorised use of contents can be:

- *Unauthorised and unlimited access* - Consumers are able to access the content as many times as they wish and can access the content from anywhere, at any time.
- *Unauthorised copy* - Consumers can make a copy of the content without the permission of the content owner.
- *Unauthorised modification* - Consumers can modify the content in any way that they wish, without the permission of the content owner.
- *Unauthorised distribution* - Consumers can illegally distribute the content to friends, family or peers, via USB transfer or email attachments.

Thus, there is a strong demand for content owners to have some degree of control over their content at any time, and from any location.

Some solutions exist in the literature and industry. However, these solutions and technologies have various limitations, including:

- *Vendor specific* - Solutions that require specific hardware and/or OS just to access the data. The content cannot be accessed anywhere else and is thus limited. For example DataSafe [33] requires specific hardware, and Microsoft's password protection only works on Microsoft products [168].
- *Non-structured* - These solutions focus on protecting one resource only and not an entire folder (or folders) of resources. For example, Tchao and Serugendo [169]

have proposed SmartContent, which aims to protect one resource. Munier [170] has proposed the idea of self-protecting documents. This protects content within the document and does not apply to other content types, such as movie files.

- *Non-flexible* - Access control implemented in these solutions does not use standard policy language. For example, Squicciarini et al.[32] have proposed a SCO that uses JAR files, which results in weaker security, even when obfuscated.

6.1.1 Scenario

We envisage a scenario where patients can share health documents with doctors from different hospitals, in a new way. Our work in previous chapters has focused on ensuring that the data is only accessible by those who have permission to do so (via key management), and protecting the data from security attackers (via secure data-sharing protocols). However, patients have little control over enforcing exactly how the data is to be used by authorised doctors. A doctor may inadvertently or intentionally distribute the patient's private health information to other unauthorised entities, without the knowledge of the patient. The patient might also wish to have their medical information only available within the confines of the hospital.

Thus, patients need a way to specify relatively complex rules; for example, ensuring that the documents can only be accessed by doctors in specific hospitals and also during working hours. In addition to allowing the patient to specify these policies, mechanisms need to be put in place to ensure that these policies are enforced at all times. The patient's ability to create his own policies for how his medical data should be used, as well as his ability to enforce them, will go a long way to improving patients' trust in using the Cloud for sensitive data-sharing purposes.

6.2 Overview

In our work, we introduce SelfProtect Object (SPO), a software object that bundles content and policy files. The DO first runs a tool to specify the exact rules for how the data is to be used. For example, specifying the IP addresses to indicate the location from which the data can be accessed and/or the times that the data can be accessed. The tool will then convert these rules into an XACML-based policy file. The DO then runs the SPO generator tool and inputs the data content(s) that is to be protected, as well as the XACML policy file. The SPO generator creates an SPO file bundling both the data content(s) and policy file. The SPO file can now be distributed everywhere.

The SPO file encapsulates both the content and policy file, and data access can only occur through an API call. An application, such as Microsoft Word, will contain a plugin. The plugin will make the API call to the SPO to request data access. When a call is made to access the data, the SPO will first verify whether the policy has been followed. It will retrieve attributes needed for decision making, such as the current time or IP address. If the policy conditions are adhered to, the SPO will release the data contents to the calling application. If the policy is not adhered to, then the SPO will deny access to the calling application.

For example, consider a policy inside the SPO, stating that a Microsoft Word document can only be accessed during business hours. An authorised DC then opens Microsoft Word and makes a call to request access, via the plugin to access the data during business hours. The SPO will first obtain the current time from the DC's machine and evaluate it with the policy. The SPO will then release the document to Microsoft Word, and the DC now has access. If the DC again attempts to access the data outside of business

hours, the policy evaluation will result in a "Deny" and the SPO will not release the document to Microsoft Word. Instead, a popup will be displayed, informing the DC that access to the document has been denied.

6.3 Design

We now provide the SafeProtect architecture, the key components of SafeProtect, and API design.

6.3.1 Architecture

Figure 6.1 illustrates the architecture for our SelfProtect object and its application.

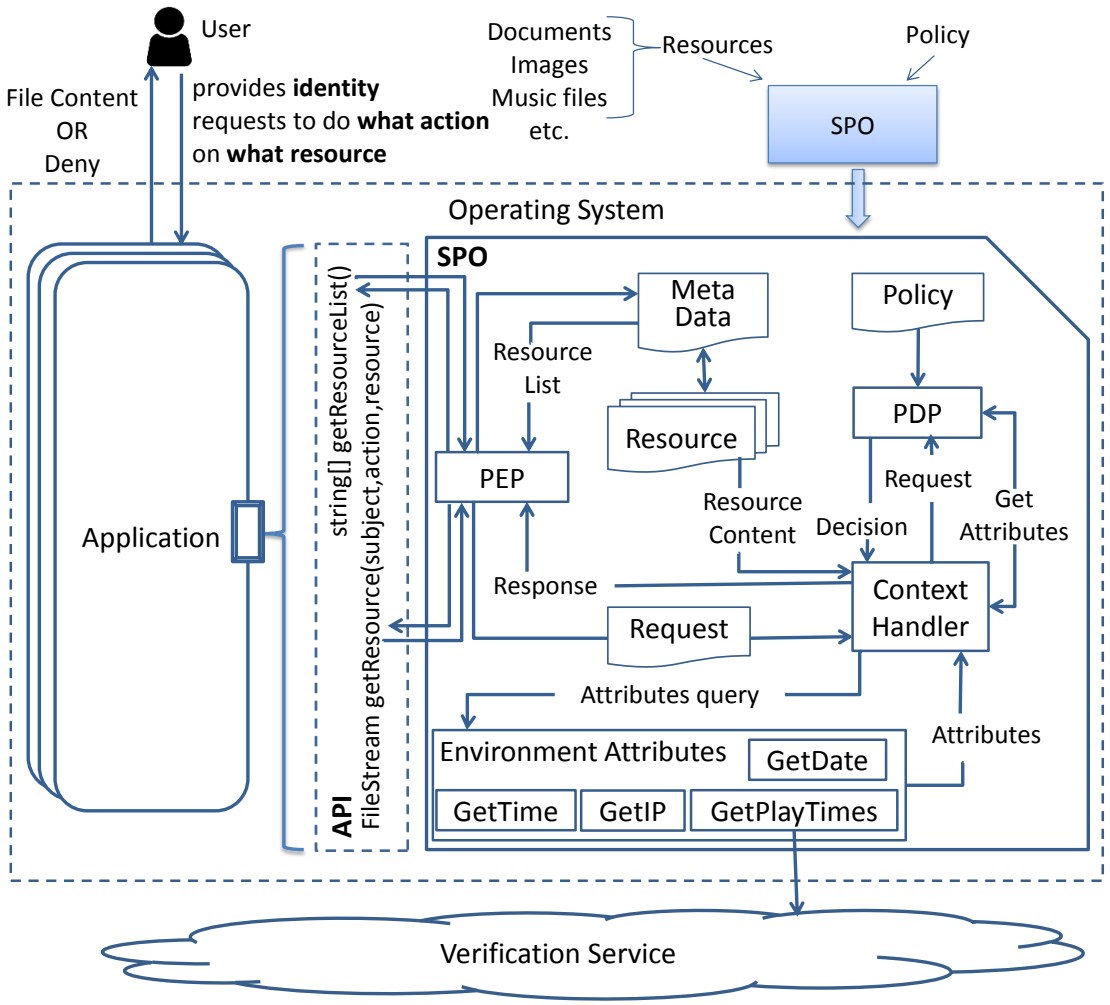


FIGURE 6.1: SPO Architecture

6.3.2 Key Components

Each of the components in the SPO architecture are described as follows:

- **Policy:** Specifies a set of rules and conditions that must be abided by, in order for the SPO to release the resource to the calling applications. The policy can be written in XACML policy language [29].

- **Resource:** Data contents that the user wishes to protect. These can include documents, images, music files, movie files, etc.
- **SPO Generator:** Takes the policy and resources as inputs and generates an SPO object as output.
- **User:** The subject who wishes to access resources contained in the SPO. It can be a username/password as well as a security token.
- **Application:** Used to access the resource. Can be any applications, such as Word to edit documents, or Media player to play movies and music files.
- **Verification Service:** A web service used to store persistent data, supporting the policy evaluation and enforcement in the SPO.
- **SPO:** The SelfProtect Object containing the policy and resources with access control mechanisms:
  - PEP - The Policy Enforcement Point. This obtains the request from the application and forwards it to the PDP, to obtain a decision. Once a decision is retrieved, it then acts upon that decision.
  - Environment attributes - These contain information such as play times, IP address, time and date.
  - Context Handler - Converts the request into XACML format and forwards it to the PDP. It also makes queries for environment attributes, to help PDP make a decision.
  - PDP - The Policy Decision Point. This evaluates the request by the application with the policy, and makes a decision on whether to permit access or deny access.

  - API - The abstract interface used by the application. It contains two methods (described in the next section) that are used by the calling application to access the resource.

We assume the user's operating system to be trusted, as well as the verification service.

6.3.3 API design

We now briefly describe the methods in our API.

- ***getResourceList*** **-** Outputs a structured list of all of the available resources contained within the SPO.
- ***getResource(subject, action, resource)*** **-** Takes as input a request with parameters *subject*, *action* and *resource*. If the policy is adhered to, the data contents will be output as a file stream.

6.4 Implementation

In this section, we detail the implementation of SPO by first discussing the technologies we used to build SPO, and then describing the protocol for content access within the SPO.

6.4.1 Selected Technologies

- **XACML:** We incorporate XACML policy language in our work. XACML [29] is a generic, open-source access control policy language built on XML. We chose XACML for our policy, since it can specify the complex conditions upon which the

data contents can be accessed. The policy language allows the content owner to develop a policy; for example, one that allows the data content to be accessed in a specific country, or within a specific time range within a day. We build a tool that allows the content owner to freely specify a wide range of conditions over their data.

- **.NET:** To prove our SPO concept, we built a SPO prototype on the .NET platform. All of our source code was compiled on C#. The SPO object itself is represented as a file with a .dll extension. We chose to build our system with .NET instead of Java, since Java would introduce privacy issues. If we were to have compiled the system with Java, the SPO object would have been represented as a JAR file. Contents in a JAR file are relatively easy to extract, even when obfuscated. For example, Squicciarini et al.[32] proposed Self-Controlling Objects that also use JAR files, which results in weaker security, even when obfuscated. In our previous work, we proposed SafeShare, which also used JAR files [155]. On the other hand, the contents in a DLL file are more private and secure, since it represents the contents as a binary. The data contents in the SPO can be encrypted persistently and decrypted on-the-fly, to provide another layer of protection at the cost of performance.

6.4.2 Protocol

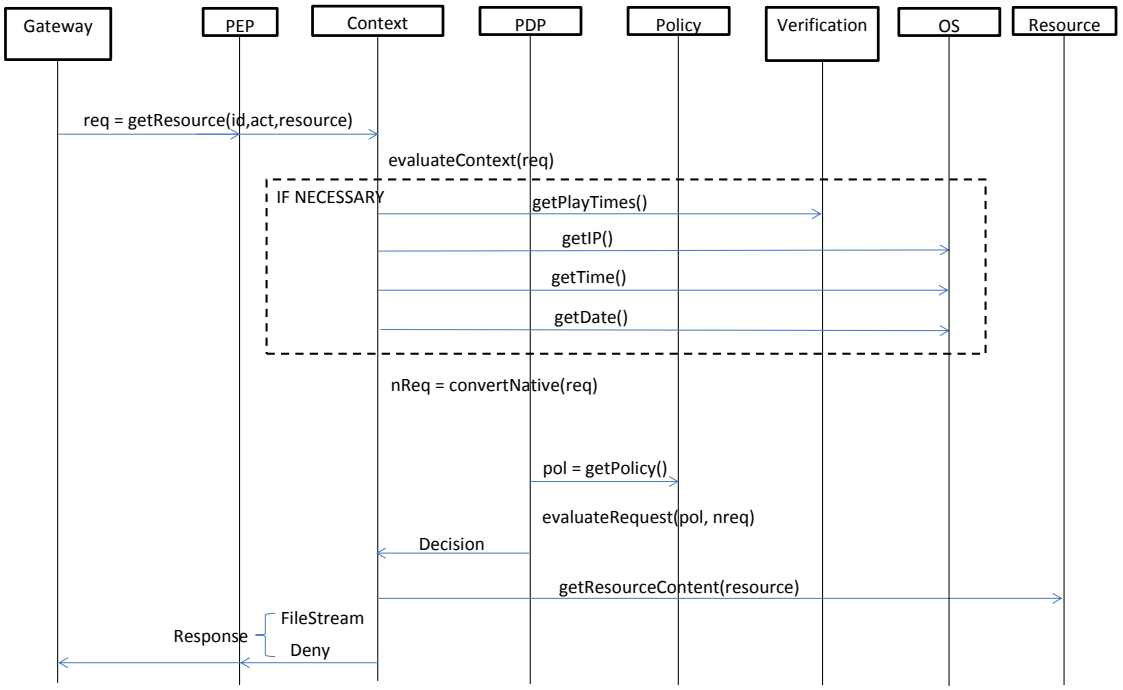


FIGURE 6.2: Resource access workflow within SPO

Figure 6.2 details the messaging sequence within the SPO, when the application makes a request to obtain a resource. The API sends the request to the PEP and similarly, to the context handler. The context handler evaluates the context and the request, and obtains environment attributes from the web server, if required. The context handler then uses the request parameters and environment attributes, and finally converts this to a native XACML request format. The native request is sent to the PDP. The PDP evaluates the request against the policy document and returns a decision of permit or deny. The decision is returned back to the context handler. The context handler then obtains the resource and returns it back to the PEP. The calling application obtains the file stream of the resource from the PEP if permitted.

6.5 Demonstration

We provide a demonstration of our system.

6.5.1 Policy Generation

We now provide a basic example of our system. Consider a policy as illustrated by our custom policy generator, in Figure 3. The content owner specifies that subjects *Donna* and *John* are allowed to access the document *testWord.docx* between 9am - 11am and 1pm - 5pm. Also, the document can only be accessed within the specified date range. Additionally, the content owner specifies that the document can only be accessed from machines with IP addresses 127.0.0.1 or 130.155.202.27, as well as only being able to access the resource a maximum of five times. When the content owner presses the "Generate XACML" button, an XACML policy file will be generated. Below is a simple example of our XACML policy file showing subjects, actions and resources, as well as a play-time condition.

```
<Subjects>
   <Subject>
      <AttributeValue DataType="...#string">
          Bob
       </AttributeValue>
   </Subject>
 </Subjects>
<Resource>
   <AttributeValue DataType="...#string">
                testWord.docx
   </AttributeValue>
```

```
</Resource>

<Action>

     <AttributeValue DataType="...#string">

          read

     </AttributeValue>

   </ActionMatch>

</Action>

<Condition FunctionId="...">

  <EnvAttrDesignator AttributeId="play-times" DataType="...#integer" />

   <AttributeValue DataType="...#integer">5</AttributeValue>
```

6.5.2 SPO Generation

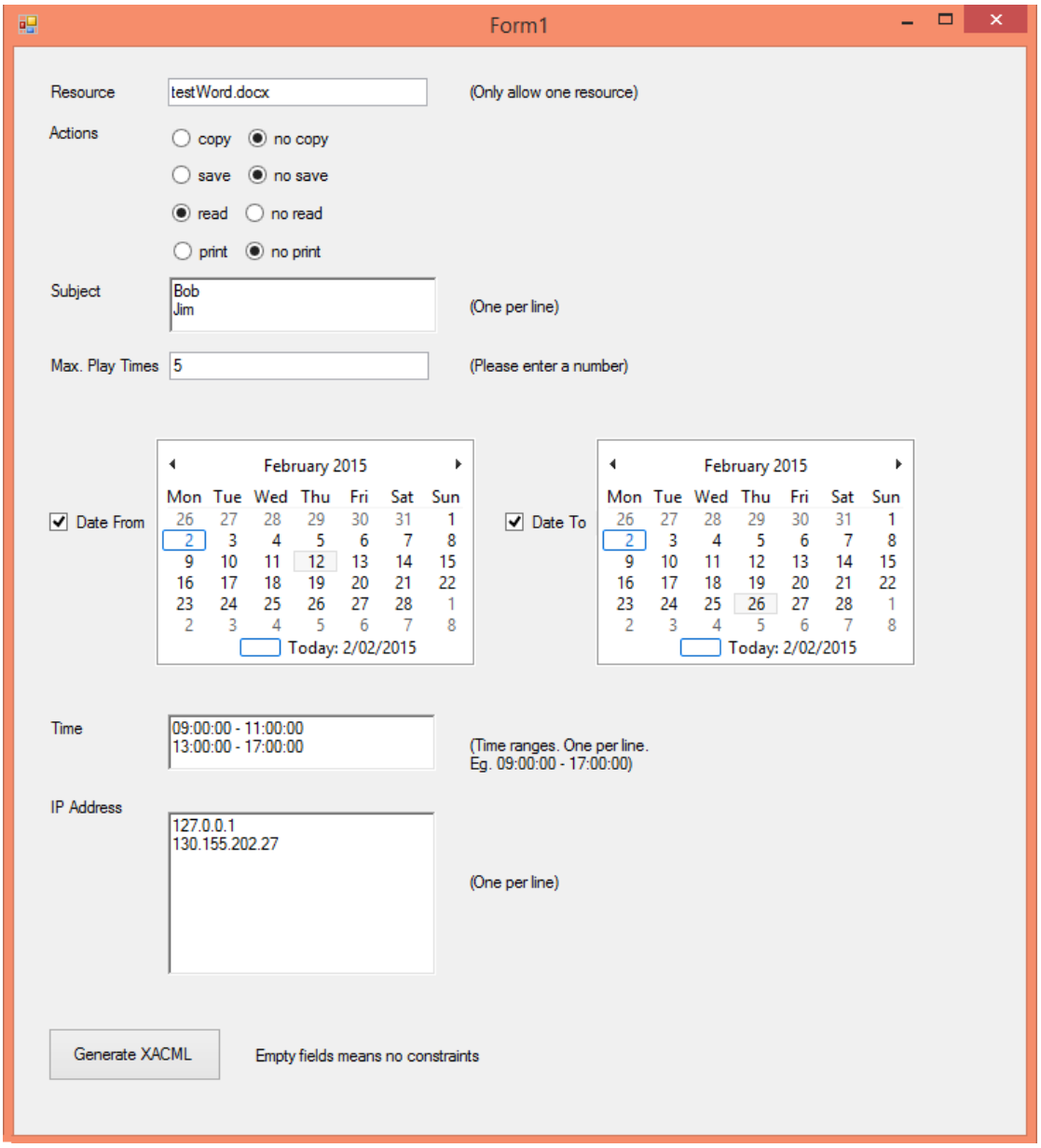


FIGURE 6.3: The Policy Generator

The content owner then generates an SPO with the SPO generator, and supplies the policy and the resource *testWord.docx* (shown in Figure 6.3. A new SPO will be generated, containing the policy and the document. The content owner is now free to store the SPO in the Cloud, or anywhere they wish.

6.5.3 SPO Consumer Access

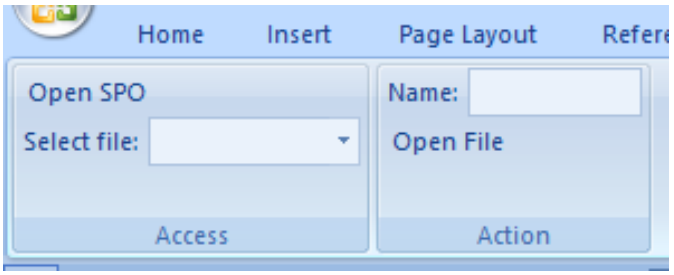


FIGURE 6.4: Microsoft Word SPO plugin

Subject *John* then uses Microsoft Word, which contains our custom plugin (as shown in Figure 6.4), to select the SPO and access the document. The plugin then sends a request to the SPO by calling the method *getResource* in the API. The SPO evaluates the request and either returns *testWord.docx* or *Deny*. If *John's* request fulfils the policy requirements, the SPO will permit access and return the document to Microsoft Word.

We also have a video demonstration on YouTube: https://www.youtube.com/watch?v=zB34wYtD4tE.

6.6 Summary

We designed and implemented SPO, which would allow data and policy to be encapsulated within a software object, similarly to SafeShare. The main difference with SafeShare was that SPO was represented as a DLL file, which provided far more security than SafeShare's JAR file. Also, SPO allowed a wide range of complex policies to be defined, due to being driven by XACML technology. SelfProtect was shown to be flexible, open, generic and structured.

Chapter 7

Conclusion

7.1 Conclusion

In this chapter, we outline our contributions and present future work.

7.1.1 Key Management

As discussed in Chapter 2, key management involves anything to do with a key, besides encryption and decryption, and covers the creation/deletion of keys, activation/deactivation of keys, transportation of keys, storage of keys, and so on. Most Cloud Service Provider's provide basic key encryption schemes for protecting data, or they might leave it to the user to encrypt their own data.

There is a strong need to encrypt data involved in the Cloud, in order to prevent malicious insider or outsider attackers from stealing the data. How do we handle the keys that are used for the encryption? Where should the keys be stored, and who has access to

those keys? How do we recover data if keys are lost? Both encryption and key management are very important to help keep data private and secure in the Cloud. Particularly in recent times, there has been a strong need for Cloud providers to adopt a robust key-management scheme for their services. However, there are still key-management issues affecting Cloud computing, as discussed in detail in Chapter 2.

The trivial solution to data sharing in the Cloud places a huge burden on the data owner, especially in terms of user revocation. If a data consumer has their access to the data owner's data revoked, the data owner would be forced to re-encrypt the data and re-distribute the keys to all of the remaining users in the group. This would be even more problematic if the group size was large, in excess of thousands. Thus, better key management is crucial to ensuring efficient data sharing in the Cloud that places little to no burden on the data owner.

We presented a key-partitioning technique that would allow efficient key management. To briefly describe the key-partitioning technique, the encrypted data key is partitioned into two (or possibly more) parts. The Cloud provider keeps one partition and the data consumer keeps the other. When a data consumer requests data access, the Cloud provider partially decrypts the data with the key, and sends this to the data consumer. The data consumer then fully decrypts, using the remaining key partition. This ensures that neither the Cloud provider nor the data consumer knows the fully decrypted key. We presented our idea through an application scenario involving the remote diagnosis of patients with cardiac arrhythmias, as well as a scenario involving the monitoring of mental health patients and providing them with feedback. In both scenarios, the performance overhead for storing and retrieving health information through our developed prototypes was negligible and thus feasible to use in a real world scenario. We also

carried out usability tests and scalability tests for the scenario involving the monitoring of mental health patients. The usability tests demonstrated the security, ease of use and satisfaction of our prototype app amongst potential users and medical professionals. The scalability tests showed that our system could handle a large number of concurrent requests.

7.1.2 Secure Data Sharing Protocol

As described in the literature review, malicious insiders remain the biggest threat when sharing data in the Cloud. This is due to insiders having direct access to the data contents. A CSP may try to steal the data to sell to third parties, in order to gain profit. Security hackers, or malicious outsiders, also remain a problem. They may carry out various security-related attacks, such as collusion attacks, man-in-the-middle attacks, sniffing attacks, etc. They try to expose vulnerabilities in the system in order to retrieve critical data. Thus, when sharing data in the Cloud, it is crucial that data is kept secure and private from both malicious insiders and outsiders, and that only those intended to access the data are able to do so.

We presented SafeShare, which leverages the concept of self-protecting objects by encapsulating the data and a policy file in a software object, and allows private and secure data sharing to occur. The data would only be revealed if the policy is adhered to. Once the data is revealed, a monitoring mechanism built into SafeShare monitors every operation of the revealed data, and either attempts to prevent an action from occurring or immediately notifies the data owner (via the Cloud) that an unauthorised operation has occurred. There were limitations with this work in terms of background monitoring and slow performance times. We then improved the SafeShare solution, and presented a hardware-based solution leveraging the use of the Trusted Extension Device (TED). A

data encryption key would be encapsulated inside the physical TED. The TED can be plugged into a computer via a USB. The TED will then attest that the device is legal and valid, with a trusted certifying authority. If proven to be valid, it downloads the encrypted data from the Cloud and decrypts inside the physical TED, and is displayed on the connected machine. In comparison to SafeShare, the performance was significantly improved in terms of encryption time but remained similar in terms of decryption time. However, this was mainly due to the limited capability and resources of the TED prototype at the time. Simply adding larger CPU and memory to TED would see drastic improvements in performance and size limitations.

7.1.3 Self-Controlled Data

Data owners lose full control over their own data once they are shared with other data consumers. Data consumers are free to do anything they want with the data owner's data. For instance, a dishonest authorised data consumer can easily take the data owner's data and send it (via email attachments and/or USB transfer) to their peers or colleagues, who do not have access permission. Furthermore, data owners may want to create more complex control over their own data, such as only allowing access to the data for a maximum of five times. Thus, there is a strong need for data owners to regain a degree of control over their own data contents.

Some current tools that data owners have at their disposal include access control matrices, ACLs, ACCLs and RBAC, to name a few. These can be used to provide control over who can access the data and what generic operations they are allowed to carry out on the data. They still lack complex control mechanisms. More recent research projects have introduced the concept of self-protecting objects, which encapsulate data and policies. Data is released from the object if, and only if, the policy is adhered to.

We created a software solution called SelfProtect Object (SPO), which similarly bundles data and policy files in software objects. SPO allows for the creation of a large variety of different, complex conditions since it uses XACML to represent the policy. The SPO can be loaded into an application (e.g., Microsoft Word) via a plugin, and the SPO will only release the data if the policy is adhered to.

We developed the SPO as well as the plugin to Microsoft Word which was used to communicate with the SPO and retrieve data if the policy was abided by. This was used to demonstrate the feasibility of our idea and we also carried out a demonstration at CCGrid 2015. This work was well recognised and achieved the Best Poster Award in CCGrid 2015.

7.2 Future Work

Throughout this thesis, we have provided various approaches to increase the trust of data owners in sharing data privately and securely with data consumers. We now highlight some future directions that potential researchers might be interested in exploring.

- Our key-partitioning technique allows data owners to securely share with, or revoke, consumers as they wish. Even though the consumer has no knowledge of the full ElGamal private key, they still obtain the symmetric key that was used to encrypt the data. The ElGamal key was used to encrypt the generic symmetric key and not the data, due to the inefficiency of encrypting relatively larger data sizes. Thus, if the symmetric key was exposed, the consumer could always decrypt the data, regardless of whether or not the data owner revokes the data consumer's access to the data. Further research could find ways to protect the symmetric key

from being exposed at the consumer end. Key management is also another potentially interesting area for further research. With the growing amount of shared data, both the Cloud provider and the data consumer could have an abundant number of keys, which might be difficult to manage. Further research could look for ways to better manage those keys and handle cases where keys are lost, stolen, etc.

- We used SafeShare as a way to securely share data in the Cloud and carry out monitoring of the data, no matter where it is accessed. We then further extended SafeShare and leveraged the hardware-based TED for stronger security. However, both SafeShare and TED had their own shortcomings. For instance, SafeShare is built using JAR files, which are not secure, and the background monitoring is too coarse-grained from only carrying out monitoring at the directory level. Future research could propose an alternative means of building SafeShare and carrying out more thorough monitoring, perhaps at the OS level. Since the TED has its own OS and storage component, this could perhaps involve having applications run in the TED, which will access the decrypted data, and using SafeShare to carry out monitoring inside the TED.

- Although our research work on self-controlling data objects helps to give data owners some confidence in terms of access control over their data, our work does not fully guarantee that the policy will be adhered to. Future research could look to further extend the SPO and allow more functionality. For example, the data owner could charge a small amount to access their data contents. This could benefit the creator of the content, as they are rewarded for the effort they put into creating their content. Also, SPO was built using the .NET platform. A potential researcher could compare the .NET platform with other platforms with regards

to representing the SPO, and find out which platform would further improve the security of the data contents inside SPO. Another future work could also explore data provenance or "permission of permissions". There is a need to protect the privacy and security of data that is derived from source data and passed on for future manipulation by other users.

To conclude, in this thesis, we researched and developed methods and technologies that would allow the data owner to confidently carry out the secure and private, sharing and collaboration of sensitive data, with data consumers.

Bibliography

[1] M. Healey. Why IT needs to push data sharing efforts. *Information Week*, 2010. URL http://www.informationweek.com/services/integration/why-it-needs-to-push-data-sharing-effort/225700544. Accessed: 15-10-2012.

[2] A. Mohamed. A history of cloud computing. *ComputerWeekly*, 2009. URL http://www.computerweekly.com/feature/A-history-of-cloud-computing. Accessed: 18-05-2015.

[3] P. Mell and T. Grance. The nist definition of cloud computing. *NIST Special Publication 800-145. National Institute of Standards and Technology, U.S. Department of Commerce*. URL http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf. Accessed: 15-10-2012.

[4] E. J. Giniat. Cloud computing: Innovating the business of health care. *Healthcare Financial Management*, 2011.

[5] ISO/IEC 17788:2014. Information technology – cloud computing – overview and vocabulary. *ISO Catalogue*. URL http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=60544. Accessed: 09-05-2016.

[6] S. M. Shariati, Abouzarjomehri, and M. H. Ahmadzadegan. Challenges and security issues in cloud computing from two perspectives: Data security and privacy

protection. *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, pages 1078–1082, 2015. doi: 10.1109/KBEI.2015.7436196.

[7] NIH Website. Nih data sharing policy and implementation guidance. *National Institute of Health (NIH)*. URL http://grants.nih.gov/grants/policy/data_sharing/data_sharing_guidance.htm. Accessed: 08-05-2016.

[8] R. Soni. How to get users to share social data with you. *Kissmetrics Blog*. URL https://blog.kissmetrics.com/get-users-to-share/. Accessed: 08-05-2016.

[9] C. Brooks. 8 benefits of online data storage. *Business News Daily*. URL http://www.businessnewsdaily.com/6294-benefits-of-online-data-storage.html. Accessed: 01-05-2016.

[10] A. Gellin. Facebook's benefits make it worthwhile. *Buffalo News*, 2012.

[11] H. A. Piwowar, R. S. Day, and D. B. Fridsma. Sharing detailed research data is associated with increased citation rate. *PLoS ONE*, 2(3):1–5, 03 2007. doi: 10.1371/journal.pone.0000308. URL http://dx.plos.org/10.1371/journal.pone.0000308.

[12] D. A. Riley. Using google wave and docs for group collaboration. *Library Hi Tech News*, 2010.

[13] R. Wu. Secure sharing of electronic medical records in cloud computing. *Arizona State University, ProQuest Dissertations and Theses*, 2012.

[14] D. Bender. Privacy and security issues in cloud computing. *Computer & Internet Lawyer*, pages 1–15, 2012.

[15] S. SeongHan, K. Kobara, and H. Imai. A secure public cloud storage system. *2011 International Conference on Internet Technology and Secured Transactions(ICITST)*, pages 103–109, 2011.

[16] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou. Security and privacy in cloud computing: A survey. *2010 Sixth International Conference on Semantics Knowledge and Grid (SKG)*, pages 105–112, 2010.

[17] F. Rocha, S. Abreu, and M. Correia. The final frontier: Confidentiality and privacy in the cloud. *Computer*, 44(9):44–50, 2011.

[18] R. Huang, X. Gui, S. Yu, and W. Zhuang. Research on privacy-preserving cloud storage framework supporting ciphertext retrieval. *2011 International Conference on Network Computing and Information Security*, pages 93–97, 2011.

[19] P. Andy. Salesforce.com scrambles to halt phishing attacks. *InternetNews.com*, 2007.

[20] F. Y. Rashid. Man-in-the-cloud attacks want your dropbox, google drive files. *PCMag*, 2015. URL http://au.pcmag.com/google-drive/36167/news/man-in-the-cloud-attacks-want-your-dropbox-google. Accessed: 26-09-2015.

[21] TripWire Guest Authors. Collusion attacks on ecommerce services. *TripWire*, 2014. URL http://www.tripwire.com/state-of-security/security-data-protection/collusion-attacks-ecommerce-services/. Accessed: 26-09-2015.

[22] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. *13th ACM Conference on Computer and Communications Security (CCS '06)*, pages 89–98, 2006.

[23] X. A. Wang and W. Zhong. A new identity based proxy re-encryption scheme. *2010 International Conference on Biomedical Engineering and Computer Science (ICBECS)*, pages 1–4, 2010.

[24] S. Tu, S. Niu, H. Li, Y. Xiao-ming, and M. Li. Fine-grained access control and revocation for sharing data on clouds. *2012 IEEE 26th International on Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW)*, pages 2146–2155, 2012.

[25] D. H. Tran, H. L. Nguyen, W. Zha, and W. K. Ng. Towards security in sharing data on cloud-based social networks. *2011 8th International Conference on Information, Communications and Signal Processing (ICICS)*, pages 1–5, 2011.

[26] A. Cavoukian. Privacy in the clouds. *Identity in the Information Society*, 1(1): 89–108, 2008.

[27] F. Sabahi. Cloud computing security threats and responses. *2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN)*, pages 245–249, 2011.

[28] J. Li, G. Zhao, X. Chen, D. Xie, C. Rong, W. Li, L. Tang, and Y. Tang. Fine-grained data access control systems with user accountability in cloud computing. *2010 IEEE Second International Conference on Cloud Computing Technology and Science(CloudCom)*, pages 89–96, 2010.

[29] Oasis Open. A brief introduction to xacml. *Oasis Website*, 2003. URL https://www.oasis-open.org/committee/download.php/2713/Brief_Introduction_to_XACML.html. Accessed: 03-02-2015.

[30] Abc4trust attribute-based credentials for trust. *ABC4Trust Website*, . URL https://abc4trust.eu/. Accessed: 19-05-2016.

[31] Au2eu. *AU2EU Website*, . URL http://www.au2eu.eu/. Accessed: 19-05-2016.

[32] A. Squicciarini, G. Petracca, and E. Bertino. Adaptive data protection in distributed systems. *3rd ACM Conference on Data and Application Security and Privacy (CODASPY)*, pages 365–376, 2013.

[33] Y. Chen, P. A. Jamkhedkar, and R. B. Lee. A software-hardware architecture for self-protecting data. *19th ACM Conference on Computer and Communications Security*, pages 14–27, 2012.

[34] Z. Xiao and Y. Xiao. Security and privacy in cloud computing. *Communications Surveys and Tutorials, IEEE 2012 Issue 99*, pages 1–17, 2012.

[35] D. Chen and H. Zhao. Data security and privacy protection issues in cloud computing. *2012 International Conference on Computer Science and Electronics Engineering*, pages 647–651, 2012.

[36] M. Zhou. Security and privacy in the cloud: A survey. *2010 Sixth International Conference on Semantics Knowledge and Grid (SKG)*, pages 204–220, 2010.

[37] J. Wang, C. Liu, and G.T.R. Lin. How to manage information security in cloud computing. *2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1405–1410, 2011.

[38] C. Saravanakumar and C. Arun. Survey on interoperability, security, trust, privacy standardization of cloud computing. *Contemporary Computing and Informatics (IC3I), 2014 International Conference*, pages 977–982, 2014. doi: 10.1109/IC3I.2014.7019735.

[39] S. Hosseinzadeh, S. Hyrynsalmi, M. Conti, and V. Leppänen. Security and privacy in cloud computing via obfuscation and diversification: A survey. *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 529–535, 2015. doi: 10.1109/CloudCom.2015.29.

[40] A. S. Raja and S. Abd Razak. Analysis of security and privacy in public cloud environment. *Cloud Computing (ICCC), 2015 International Conference*, pages 1–6, 2015. doi: 10.1109/CLOUDCOMP.2015.7149630.

[41] M. Kumari and R. Nath. Security concerns and countermeasures in cloud computing paradigm. *2015 Fifth International Conference on Advanced Computing Communication Technologies*, pages 534–540, 2015. ISSN 2327-0632. doi: 10.1109/ACCT.2015.80.

[42] N. Oza, K. Karppinen, and R. Savola. User experience and security in the cloud – an empirical study in the finnish cloud consortium. *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 621–628, 2010.

[43] Y. Wang. The role of saas privacy and security compliance for continued saas use. *2011 7th International Conference on Networked Computing and Advanced Information Management (NCM)*, pages 303–306, 2011.

[44] R. B. Parker. A definition of privacy. *Rutgers Law Review*, pages 275–296, 1974.

[45] NIST. Glossary of key information security terms. *NIST Website*, . URL http://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7298r2.pdf. Accessed: 18-05-2015.

[46] Microsoft TechNet. Motivations of a criminal hacker. URL http://technet.microsoft.com/en-us/library/cc505924.aspx. Accessed: 15-10-2012.

[47] Crucial Paradigm Web Solutions. Hacking attacks - how and why. URL http://www.crucialp.com/resources/tutorials/website-web-page-site-optimization/hacking-attacks-how-and-why.php. Accessed: 15-10-2012.

[48] Wikileaks. URL http://wikileaks.org. Accessed: 15-10-2012.

[49] P. Crosman. Are cloud services safe? icloud breach revives debate. *American Banker*, 2014. URL http://www.americanbanker.com/issues/179_170/are-cloud-services-safe-icloud-breach-revives-debate-1069736-1.html. Accessed: 26-09-2015.

[50] H. Orman. The morris worm: a fifteen-year perspective. *IEEE Security Privacy*, 1(5):35–43, 2003. ISSN 1540-7993. doi: 10.1109/MSECP.2003.1236233.

[51] J. Kahn. The homeless hacker v. the new york times. *Wired*, 2004. URL http://archive.wired.com/wired/archive/12.04/hacker.html. Accessed: 26-09-2015.

[52] Ruhr. Cloud computing: Gaps in the 'cloud'. *NewsRx Health & Science*, pages 1–15, 2011.

[53] K. Zunnurhain and S. V. Vrbsky. Security attacks and solutions in clouds. *CloudCom2010 Poster*, 2010.

[54] A. P. Schwab, L. Frank, and N. Gligorov. Saying privacy, meaning confidentiality. *The American Journal of Bioethics*, pages 44–45, 2011.

[55] U.S. Department of Health and Human Services. Hipaa privacy. *U.S. Department of Health and Human Services Website*, 2012. URL http://www.hhs.gov/ocr/privacy/hipaa/understanding/index.html. Accessed: 20-10-2014.

[56] Protecting personal health information in research. *med.data.edu.au Website*. URL http://med.data.edu.au/personal-health-data-and-research/. Accessed: 16-05-2016.

[57] E. McCann. Hackers exploit heartbleed to swipe data of 4.5 million. *Healthcare IT News*, 2014. URL http://www.healthcareitnews.com/news/hackers-exploit-heartbleed-swipe-data-45-million. Accessed: 26-09-2015.

[58] HIPAA Journal. Touchstone medical imaging suffers 307k patient data breach. *HIPAA Journal Website*, 2014. URL http://www.hipaajournal.com/touchstone-medical-imaging-suffers-307k-patient-data-breach/. Accessed: 26-09-2015.

[59] School Libraries in Canada. Internet privacy? *The Ethics of Information use - A Teachers' Guide*, pages 20–22, 2001.

[60] C. Donlon-Cotton. Privacy and social networking. *Law & Order*, pages 16–17, 2010.

[61] M. Hachman. New facebook phishing attack steals accounts, financial information. *PC Mag*, 2012. URL http://www.pcmag.com/article2/0,2817,2398922,00.asp. Accessed: 16-10-2012.

[62] C. Albanesius. Ramnit computer worm compromises 45k facebook logins. *PC Mag*, 2012. URL http://www.pcmag.com/article2/0,2817,2398432,00.asp. Accessed: 16-10-2012.

[63] L. Whitney. Feds investigate alleged attacks on gmail accounts. *CNet news*, 2011. URL http://news.cnet.com/8301-1009_3-20068229-83/feds-investigate-alleged-attacks-on-gmail-accounts. Accessed: 16-10-2012.

[64] C. Jim and L. Chyen Yee. Hacker attacks threaten to dampen cloud computing's prospects. *Reuters article*, 2011. URL http://www.reuters.com/article/2011/06/03/us-cloudcomputing-idUSTRE7521WQ20110603. Accessed: 16-10-2012.

[65] K. Dominguez. Trend micro researchers identify vulnerability in hotmail. *Trend Micro*, 2012. URL http://blog.trendmicro.com/trendlabs-security-intelligence/trend-micro-researchers-identify-vulnerability-in-hotmail/. Accessed: 16-10-2012.

[66] S. Choney. Hotmail, yahoo mail users also targets in attacks. *NBC News*, 2011. URL http://www.nbcnews.com/technology/technolog/hotmail-yahoo-mail-users-also-targets-attacks-123078. Accessed: 16-10-2012.

[67] A. Charles. Playstation network: hackers claim to have 2.2m credit cards. *The Guardian Technology Blog*, 2011. URL http://www.guardian.co.uk/technology/blog/2011/apr/29/playstation-network-hackers-credit-cards. Accessed: 16-10-2012.

[68] M. R. Priscilla. Privacy, government information, and technology. *Public Administration Review*, pages 629–634, 1986. URL http://www.jstor.org/stable/976229. Accessed: 15-10-2012.

[69] R. Longley. Federal privacy act. *About.com*, 2012. URL http://usgovinfo.about.com/library/weekly/aa121299a.htm. Accessed: 15-10-2012.

[70] J. Mcbeth. Governments need privacy too. *The Straits Times*, 2011.

[71] R. Verma. Confidentiality and privacy issues. *The Law Handbook. Education Law*, 2012. URL http://www.lawhandbook.org.au/handbook/ch06s03s08.php. Accessed: 16-10-2012.

[72] G. Hulme. Amazon web services ddos attack and the cloud. *InformationWeek*, 2009. URL http://www.informationweek.com/security/amazon-web-services-ddos-attack-and-the/229204417. Accessed: 16-10-2012.

[73] NIST. Nist privacy and security guidelines. *NIST Website*, . URL http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf. Accessed: 15-10-2012.

[74] P. Ramkumar D. Jayalatchumy and D. Kadhirvelu. Preserving privacy through data control in a cloud computing architecture using discretion algorithm. *2010 Third International Conference on Emerging Trends in Engineering and Technology (ICETET)*, pages 456–461, 2010.

[75] P. S. Ruotsalainen, B. Blobel, A. Seppälä, and P. Nykänen. Trust information-based privacy architecture for ubiquitous health. *JMIR Mhealth Uhealth*, 1(2):e23, 2013.

[76] C. Gentry. A fully homomorphic encryption scheme. *Dissertation*, 2009.

[77] J. Yao, S. Chen, S. Nepal, D. Levy, and J. Zic. Truststore: Making amazon s3 trustworthy with services composition. *2010 IEEE/ACM 10th International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, pages 600–605, 2010.

[78] B. M. Silva, J. J. Rodrigues, F. Canelo, I. C. Lopes, and L. Zhou. A data encryption solution for mobile health apps in cooperation environments. *J Med Internet Res*, 15(4):e66, 2013.

[79] G. Zhang, X. Liu, and Y. Yang. Time-series pattern based effective noise generation for privacy protection on cloud. *IEEE Transactions on Computers*, 2014.

[80] M. Paul, C. Collberg, and D. Bambauer. A possible solution for privacy preserving cloud data storage. *Cloud Engineering (IC2E), 2015 IEEE International Conference*, pages 397–403, 2015. doi: 10.1109/IC2E.2015.103.

[81] S. M. Rahaman and M. Farhatullah. Pccp: A model for preserving cloud computing privacy. *Data Science Engineering (ICDSE), 2012 International Conference on*, pages 166–170, 2012. doi: 10.1109/ICDSE.2012.6281900.

[82] M. Luther. Federated key management for secure cloud computing. *Voltage Security Conference Presentation*, 2010. URL http://storageconference.org/2010/Presentations/KMS/17.Martin.pdf. Accessed: 25-11-2012.

[83] S. Pate and T. Tambay. Securing the cloud - using encryption and key management to solve today's cloud security challenges. *Storage Networking Industry Association 2011*, 2011. URL http://www.snia.org/sites/default/education/tutorials/2011/spring/security/PateTambay_Securing_the_Cloud_Key_Mgt.pdf. Accessed: 25-11-2012.

[84] Cloud Security Alliance Wiki. Encryption and key management. URL https://wiki.cloudsecurityalliance.org/guidance/index.php/Encryption_and_Key_Management. Accessed: 25-11-2012.

[85] T. Mather. Key management in the cloud. *O'Reilly Community*, 2010. URL http://broadcast.oreilly.com/2010/01/key-management-in-the-cloud.html. Accessed: 25-11-2012.

[86] Oasis. Oasis key management interoperability protocol. *Oasis Open Weibiste*. URL https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=kmip#overview. Accessed: 25-11-2012.

[87] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. Recommendation for key management - part 1: General (revised) computer security. *NIST Special Publication 800-57*, 2007. URL http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf. Accessed: 25-11-2012.

[88] ISO. Iso/iec 11770-5:2011 information technology - security techniques - key management - part 5: Group key management. *ISO Standards catalogue*, . URL http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54527. Accessed: 25-11-2012.

[89] ISO. Iso 11568-2:2012 financial services - key management (retail) - part 2: Symmetric ciphers, their key management and life cycle. *ISO Standards catalogue*, . URL http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=53568. Accessed: 25-11-2012.

[90] G. Zhao, C. Rong, J. Li, F. Zhang, and Y. Tang. Trusted data sharing over untrusted cloud storage providers. *2010 IEEE Second International Conference on Cloud Computing Technology and Science(CloudCom)*, pages 97–103, 2010.

[91] S. Lei, D. Zishan, and G. Jindi. Research on key management infrastructure in cloud computing environment. *2010 9th International Conference on Grid and Cooperative Computing (GCC)*, pages 404–407, 2010.

[92] H. Fathi, S. Shin, K. Kobara, S. Chakraborty, H. Imai, and R. Prasad. Lr-ake-based aaa for network mobility (nemo) over wireless links. *IEEE J. Select. Areas Commun*, 24(9):1725–1737, 2006.

[93] S. Sanka, C. Hota, and M. Rajarajan. Secure data access in cloud computing. *2010 IEEE 4th International Conference on Internet Multimedia Services Architecture and Application(IMSAA)*, pages 1–6, 2010.

[94] N. Bennani, E. Damiani, and S. Cimato. Toward cloud-based key management for outsourced databases. *2010 IEEE 34th Annual Conference on Computer Software and Applications Conference Workshops (COMPSACW)*, pages 232–236, 2010.

[95] A. Boldyreva, V. Goyal, and V. Kumar. Identity-based encryption with efficient revocation. *15th ACM Conference on Computer and communications security (CCS '08)*, pages 417–472, 2008.

[96] A. Sahai and H. Seyalioglu. Worry-free encryption: functional encryption with public keys. *17th ACM Conference on Computer and communications security(CCS '10)*, pages 463–472, 2010.

[97] Yan Wang, Zhi Li, and Yuxia Sun. Cloud computing key management mechanism for cloud storage. *Third International Conference on Cyberspace Technology (CCT 2015)*, pages 1–4, 2015. doi: 10.1049/cp.2015.0855.

[98] N. Song and Y. Chen. Novel hyper-combined public key based cloud storage key management scheme. *China Communications*, 11(14):185–194, 2014. ISSN 1673-5447. doi: 10.1109/CC.2014.7085619.

[99] A. R. Buchade and R. Ingle. Key management for cloud data storage: Methods and comparisons. *2014 Fourth International Conference on Advanced Computing Communication Technologies*, pages 263–270, 2014. ISSN 2327-0632. doi: 10.1109/ACCT.2014.78.

[100] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *IEEE Transactions on Parallel and Distributed Systems*, pages 131–143, 2013.

[101] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. *2010 Proceedings IEEE on INFOCOM*, pages 1–9, 2010.

[102] Y. Yang and Y. Zhang. A generic scheme for secure data sharing in cloud. *2011 40th International Conference on Parallel Processing Workshops (ICPPW)*, pages 145–153, 2011.

[103] Q. Liu, G. Wang, and J. Wu. Clock-based proxy re-encryption scheme in unreliable clouds. *2012 41st International Conference on Parallel Processing Workshops (ICPPW)*, pages 304–305, 2012.

[104] S. Sundareswaran, A. C. Squicciarini, and D. Lin. Ensuring distributed accountability for data sharing in the cloud. *IEEE Transactions on Dependable and Secure Computing*, 9(4):556–568, 2012.

[105] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. *Security and Privacy, IEEE Symposium*, pages 321–334, 2007.

[106] A.V.D.M. Kayem. On monitoring information flow of outsourced data. *Information Security for South Africa (ISSA)*, pages 1–8, 2010.

[107] P. Burnap and J. Hilton. Self protecting data for de-perimeterised information sharing. *Digital Society. ICDS '09*, pages 65–70, 2009.

[108] M. S. Kirkpatrick and S. Kerr. Enforcing physically restricted access control for remote data. *ACM Conference on Data and application security and privacy (CODASPY '11)*, pages 203–212, 2011.

[109] J. Zic and S. Nepal. Implementing a portable trusted environment. *Future of Trust in Computing Conference*, pages 17–29, 2008.

[110] S. Nepal, J. Zic, H. Hwang, and D. Moreland. Trust extension device: Providing mobility and portability of trust in cooperative information systems. *On the Move to Meaningful Internet Systems, Lecture notes in computer science 4803*, pages 253–271, 2007.

[111] E. Bertino M. Nabeel. Attribute based group key management. *Transactions On Data Privacy 7*, pages 309–336, 2014.

[112] B. Qing-hai and Z. Ying. Study on the access control model. 1:830–834, July 2011. doi: 10.1109/CSQRWC.2011.6037079.

[113] W3C Information and Knowledge Domain. Extensible markup language(xml). *W3C Website*. URL https://www.w3.org/XML/. Accessed: 28-04-2016.

[114] B. Li. Research and application of soa standards in the integration on web services. *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*, 2:492–495, 2010. doi: 10.1109/ETCS.2010.199.

[115] N. Shahgholi, M. A. Seyyedi, M. Mohsenzadeh, and S. H. Qorani. A new security framework against web services' xml attacks in soa. *Next Generation Web Services Practices (NWeSP), 2011 7th International Conference on*, pages 314–319, 2011. doi: 10.1109/NWeSP.2011.6088197.

[116] C. Xin. Service-oriented architecture in it. *Information Processing, 2009. APCIP 2009. Asia-Pacific Conference on*, 2:493–496, 2009. doi: 10.1109/APCIP.2009.257.

[117] N. A. Nordbotten. Xml and web services security standards. *IEEE Communications Surveys Tutorials*, 11(3):4–21, 2009. ISSN 1553-877X. doi: 10.1109/SURV.2009.090302.

[118] J. M. Tekli, E. Damiani, R. Chbeir, and G. Gianini. Soap processing performance and enhancement. *IEEE Transactions on Services Computing*, 5(3):387–403, 2012. ISSN 1939-1374. doi: 10.1109/TSC.2011.11.

[119] Bluetooth Technology Web Site. URL http://www.bluetooth.com/Pages/Bluetooth-Home.aspx. Accessed: 12-01-2013.

[120] National Academy of Engineering. URL http://www.engineeringchallenges.org/cms/8996/8938.aspx. Accessed: 12-01-2013.

[121] W. Swan. Australia to 2050: future challenges. *Intergeneration Report*, 2010. URL http://archive.treasury.gov.au/igr/igr2010/report/pdf/IGR_2010.pdf. Accessed: 12-01-2013.

[122] O2. O2 health takes telecare mobile. *M2 Presswire*, 2012.

[123] Doro. Bosch selects doro to enrich telecare offering with mobile solutions. *M2 Presswire*, 2012.

[124] Network Weekly News. Numera; numera acquires bluelibris and expands offerings into telecare. page 625, 2012.

[125] EMC Corporation. Data sharing is the key to lower healthcare costs. *HealthTech Wire Interview*, 2012. URL http://www.healthtechwire.com/emc-corporation/data-sharing-is-the-key-to-lower-healthcare-costs-3132/. Accessed: 14-01-2013.

[126] M. Aramudhan and K. Mohan. New secure communication protocols for mobile e-health system. *International Journal of Computer Applications 8(4)*, pages 10–15, 2010.

[127] R. Marti, J. Delgado, and X. Perramon. Security specification and implementation for mobile e-health services. *2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE '04)*, pages 241–248, 2004.

[128] G. Fortino, M. Pathan, and G. Di Fatta. Bodycloud: Integration of cloud computing and body sensor networks. pages 851–856, Dec 2012.

[129] F. Bellifemine, G. Fortino, R. Giannantonio, R. Gravina, A. Guerrieri, and M. Sgroi. Spine: a domain-specific framework for rapid prototyping of wbsn applications. *Software: Practice and Experience*, 41(3):237–265, 2011. ISSN 1097-024X. doi: 10.1002/spe.998. URL http://dx.doi.org/10.1002/spe.998.

[130] G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski, and R. Jafari. Enabling effective programming and flexible management of efficient body sensor network applications. *IEEE Transactions on Human-Machine Systems*, 43(1):115–133, Jan 2013.

[131] S. Pandey, W. Voorsluys, S. Niu, A. Khandoker, and R. Buyya. An autonomic cloud environment for hosting ecg data analysis services. *Future Generation Computer Systems*, 28:147–154, 2012.

[132] BusinessWire. Alivecor(tm) mobile ecg device for heart rhythm monitoring now available by prescription. *Business Wire*, 2013. URL http://www.businesswire.com/news/home/20131218005703/en/AliveCor-Expands-eHealth-Service-Android-Users#.VgI_9t-qpBc. Accessed: 22-12-2012.

[133] BusinessWire. Cardiocomm solutions, inc. reveals a new remote mobile ecg monitoring solution at medica 2011. *Business Wire*, 2011. URL http://www.businesswire.com/news/home/20111121006670/en/CardioComm-

Solutions-Reveals-Remote-Mobile-ECG-Monitoring#.VgJAed-qpBc. Accessed: 22-12-2012.

[134] S. Gradl, P. Kugler, C. Lohmuller, and B. Eskofier. Real-time ecg monitoring and arrhythmia detection using android-based mobile devices. pages 2452–2455, Aug 2012.

[135] H. Xia, I. Asif, and X. Zhao. Cloud-ecg for real time ecg monitoring and analysis. *Computer Methods and Programs in Biomedicine*, 110:253–259, 2013.

[136] J. Saarinen. Uk health trust fined for privacy breach. *Itnews Technology News*, 2012. URL http://www.itnews.com.au/News/311079,uk-health-trust-fined-for-privacy-breach.aspx. Accessed: 14-01-2013.

[137] S. Geoghegan. The latest on data sharing & secure cloud computing. *Law & Order*, pages 24–26, 2012.

[138] Asus eee pad transformer prime tf201. . URL http://www.asus.com/Eee/Eee_Pad/Eee_Pad_Transformer_Prime_TF201/. Accessed: 22-12-2012.

[139] Alive technologies. . URL http://www.alivetec.com/index.htm. Accessed: 22-12-2012.

[140] Ambu blue sensor ecg electrodes. . URL http://www.ambu.com/corp/products/clinical_studies/ambu_blue_sensor_ecg_electrodes.aspx. Accessed: 22-12-2012.

[141] D. Thilakanathan, R. A. Calvo, S. Chen, S. Nepal, and N. Glozier. Facilitating secure sharing of personal health data in the cloud. *Journal of Medical Internet Research Medical Informatics (JMIR Medical Informatics)*, 2016.

[142] M. E. Hilliard, A. Hahn, A. K. Ridge, M. N. Eakin, and K. A. Riekert. User preferences and design recommendations for an mhealth app to promote cystic fibrosis self-management. *JMIR Mhealth Uhealth*, 2(4):e44, 2014. URL http://mhealth.jmir.org/2014/4/e44/. Accessed: 06-10-2014.

[143] F. J. Grajales III, S. Sheps, K. Ho, H. Nova-Lauscher, and G. Eysenbach. Social media: A review and tutorial of applications in medicine and health care. *J Med Internet Res*, 16(2):e13, 2014.

[144] A. Westin. Privacy and freedom. *New York: Atheneum*, 1967.

[145] Microsoft healthvault. . URL https://www.healthvault.com/au/en. Accessed: 06-10-2014.

[146] Healthvault faqs. . URL http://msdn.microsoft.com/en-au/healthvault/cc196394.aspx. Accessed: 06-10-2014.

[147] K. Sparks, B. Faragher, and C. L. Cooper. Wellbeing and occupational health in the 21st century workplace. *Journal of occupational and organizational psychology 74.4*, pages 489–509, 2011.

[148] P. Karvelas. Australia's mental health system must become more efficient. *The Australian*, 2014. URL http://www.theaustralian.com.au/national-affairs/policy/australias-mental-health-system-must-become-more-efficient/story-fn59nokw-1226850819260#. Accessed: 06-10-2014.

[149] T. Donker, K. Petrie, J. Proudfoot, J. Clarke, M. R. Birch, and H. Christensen. Smartphones for smarter delivery of mental health programs: A systematic review. *J Med Internet Res*, 15(11):e247, 2013.

[150] ProVerif. Proverif: Cryptographic protocol verifier in the formal model. URL http://prosecco.gforge.inria.fr/personal/bblanche/proverif/. Accessed: 06-10-2014.

[151] B. Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 17(4):363–434, 2013.

[152] J. Nielsen. Usability 101: Introduction to usability. *Nielsen Norman Group*, 2012. URL http://www.nngroup.com/articles/usability-101-introduction-to-usability/. Accessed: 06-10-2014.

[153] N. Regola and N. V. Chawla. Storing and using health data in a virtual private cloud. *J Med Internet Res*, 15(3):e63, 2013.

[154] A. M. Lund. Measuring usability with the use questionnaire. URL http://www.stcsig.org/usability/newsletter/0110_measuring_with_use.html. Accessed: 06-10-2014.

[155] D. Thilakanathan, S. Chen, S. Nepal, and R. A. Calvo. Secure and controlled sharing of data in distributed computing. *2nd IEEE International Conference on Big Data Science and Engineering*, pages 825–832, 2013.

[156] D. Sullivan. Reducing risks with multiple cloud service providers. *SearchCloudComputing*. URL http://searchcloudcomputing.techtarget.com/tip/Reducing-risks-with-multiple-cloud-service-providers. Accessed: 06-05-2016.

[157] J. Wang. Cp-abe java implementation using jpbc. URL http://wakemecn.com/cpabe/. Accessed: 02-08-2013.

[158] The Java Pairing Based Cryptography Library (jPBC) website. URL http://gas.dia.unisa.it/projects/jpbc/. Accessed: 02-08-2013.

[159] ProGuard. Proguard jar shrinker and obfuscator. URL http://proguard.sourceforge.net/. Accessed: 02-08-2013.

[160] D. Thilakanathan, S. Chen, S. Nepal, and R. A. Calvo. Secure and controlled sharing of data in distributed computing. *2nd IEEE International Conference on Big Data Science and Engineering (BDSE 2013)*, pages 825–832, 2013.

[161] K. W. Miller, J. Voas, and G.F. Hurlburt. Byod: Security and privacy considerations. *IT Professional*, 14(5):53–55, 2012. URL http://dx.doi.org/10.1109/MITP.2012.93.

[162] G. Thomson. Byod: enabling the chaos. *Network Security*, pages 5–8, 2012. URL http://dx.doi.org/10.1016/S1353-4858(12)70013-2.

[163] S. Nepal, J. Zic, D. Liu, and J. Jang. A mobile and portable trusted computing platform. *EURASIP J. Wireless Comm. and Networking*, page 75, 2011.

[164] J. Zic and T. Hardjono. Towards a cloud-based integrity measurement service. *Journal of Cloud Computing: Advances, Systems and Applications*, 2(1):1–9, 2013. ISSN 2192-113X. doi: 10.1186/2192-113X-2-4. URL http://dx.doi.org/10.1186/2192-113X-2-4.

[165] S. Chen, D. Thilakanathan, D. Xu, S. Nepal, and R. A. Calvo. Self protecting data sharing using generic policies. *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2015)*, pages 1197–1200, 2015.

[166] M. Wall. Big data: Are you ready for blast-off? *BBC News Business*, 2014. URL http://www.bbc.com/news/business-26383058. Accessed: 03-02-2015.

[167] TechGuru. All about digital rights management (drm). *TechGuru Website*, 2009. URL http://www.techquark.com/2009/02/all-about-digital-rights-management-drm.html. Accessed: 12-05-2016.

[168] Microsoft Office. Password protect documents, workbooks, and presentations. *Office website*. URL https://support.office.com/en-au/article/Password-protect-documents-workbooks-and-presentations-ef163677-3195-40ba-885a-d50fa2bb6b68. Accessed: 03-02-2015.

[169] A. Tchao and G. D. M. Serugendo. Smartcontent: A self-protecting and context-aware active content. *2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW)*, pages 151–156, 2012.

[170] M. Munier, V. Lalanne, and M. Ricarde. Self-protecting documents for cloud storage security. *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1231–1238, 2012.